Aurox
Podręcznik trenera

Wojciech Gabor Michał Górniak Przemysław Kasierski
Lech Ocaya Gamon

19 grudnia 2004 roku

Część I

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom : to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation : a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties : any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts : Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must

present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version :

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http ://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Część II

Licencja GNU Wolnej Dokumentacji

Uwaga. To jest nieoficjalne tłumaczenie Licencji GNU Wolnej Dokumentacji na język polski. Nie zostało opublikowane przez Free Software Foundation i pod względem prawnym nie stanowi warunków rozpowszechniania tekstów stosujących GNU FDL – ustanawia je wyłącznie oryginalny angielski tekst licencji GNU FDL. Jednak mamy nadzieję, że pomoże ono lepiej zrozumieć Licencję osobom mówiącym po polsku.

Preambuła

Celem niniejszej licencji jest zagwarantowanie wolnego dostępu do podręcznika, treści książki i wszelkiej dokumentacji w formie pisanej oraz zapewnienie każdemu użytkownikowi swobody kopiowania i rozpowszechniania wyżej wymienionych, z dokonywaniem modyfikacji lub bez, zarówno w celach komercyjnych, jak i nie komercyjnych. Ponadto Licencja ta pozwala przyznać zasługi autorowi i wydawcy przy jednoczesnym ich zwolnieniu z odpowiedzialności za modyfikacje dokonywane przez innych.

Niniejsza Licencja zastrzega też, że wszelkie prace powstałe na podstawie tego dokumentu muszą nosić cechę wolnego dostępu w tym samym sensie, co produkt oryginalny. Licencja stanowi uzupełnienie Powszechnej Licencji Publicznej GNU (GNU General Public License), która jest licencją dotyczącą wolnego oprogramowania.

Niniejsza Licencja została opracowana z zamiarem zastosowania jej do podręczników do wolnego oprogramowania, ponieważ wolne oprogramowanie wymaga wolnej dokumentacji : wolny program powinien być rozpowszechniany z podręcznikami, których dotyczą te same prawa, które wiążą się z oprogramowaniem. Licencja ta nie ogranicza się jednak do podręczników oprogramowania. Można ją stosować do różnych dokumentów tekstowych, bez względu na ich przedmiot oraz niezależnie od tego, czy zostały opublikowane w postaci książki drukowanej. Stosowanie tej Licencji zalecane jest głównie w przypadku prac, których celem jest instruktaż lub pomoc podręczna.

1. Zastosowanie i definicje

Niniejsza Licencja stosuje się do podręczników i innych prac, na których umieszczona jest pochodząca od właściciela praw autorskich informacja, że dana praca może być rozpowszechniana wyłącznie na warunkach niniejszej Licencji. Używane poniżej słowo **"Dokument"** odnosić się będzie do wszelkich tego typu publikacji. Ich odbiorcy nazywani będą licencjobiorcami.

"Zmodyfikowana wersja" Dokumentu oznacza wszelkie prace zawierające Dokument lub jego część w postaci dosłownej bądź zmodyfikowanej i/lub przełożonej na inny język.

"Sekcją drugorzędną" nazywa się dodatek opatrzony odrębnym tytułem lub sekcję początkową Dokumentu, która dotyczy wyłącznie związku wydawców lub autorów Dokumentu z ogólną tematyką Dokumentu (lub zagadnieniami z nią związanymi) i nie zawiera żadnych treści bezpośrednio związanych z ogólną tematyką (na przykład, jeżeli Dokument stanowi w części podręcznik matematyki, Sekcja drugorzędna nie może wyjaśniać zagadnień matematycznych). Wyżej wyjaśniany związek może się natomiast wyrażać w aspektach historycznym, prawnym, komercyjnym, filozoficznym, etycznym lub politycznym.

“Sekcje niezmienne” to takie Sekcje drugorzędne, których tytuły są ustalone jako tytuły Sekcji niezmiennych w nocie informującej, że Dokument został opublikowany na warunkach Licencji.

“Treść okładki” to pewne krótkie fragmenty tekstu, które w nocie informującej, że Dokument został opublikowany na warunkach Licencji, są opisywane jako “do umieszczenia na przedniej okładce” lub “do umieszczenia na tylnej okładce”.

“Jawna” kopia Dokumentu oznacza kopię czytelną dla komputera, zapisaną w formacie, którego specyfikacja jest publicznie dostępna. Zawartość tej kopii może być oglądana i edytowana bezpośrednio za pomocą typowego edytora tekstu lub (w przypadku obrazów złożonych z pikseli) za pomocą typowego programu graficznego lub (w przypadku rysunków) za pomocą ogólnie dostępnego edytora rysunków. Ponadto kopia ta stanowi odpowiednie dane wejściowe dla programów formatujących tekst lub dla programów konwertujących do różnych formatów odpowiednich dla programów formatujących tekst. Kopia spełniająca powyższe warunki, w której jednak zostały wstawione znaczniki mające na celu utrudnienie dalszych modyfikacji przez czytelników, nie jest Jawna. Kopię, która nie jest “Jawna”, nazywa się **“Niejawną”**.

Przykładowe formaty kopii Jawnych to : czysty tekst ASCII bez znaczników, format wejściowy Texinfo, format wejściowy LaTeX, SGML lub XML wykorzystujące publicznie dostępne DTD, standardowy prosty HTML przeznaczony do ręcznej modyfikacji. Formaty niejawne to na przykład PostScript, PDF, formaty własne, które mogą być odczytywane i edytowane jedynie przez własne edytory tekstu, SGML lub XML, dla których DTD i/lub narzędzia przetwarzające nie są ogólnie dostępne, oraz HTML wygenerowany maszynowo przez niektóre procesory tekstu jedynie w celu uzyskania danych wynikowych.

“Strona tytułowa” oznacza, w przypadku książki drukowanej, samą stronę tytułową oraz kolejne strony zawierające informacje, które zgodnie z tą Licencją muszą pojawić się na stronie tytułowej. W przypadku prac w formatach nieposiadających strony tytułowej “Strona tytułowa” oznacza tekst pojawiający się najbliżej tytułu pracy, poprzedzający początek tekstu głównego.

2. Kopiowanie dosłowne

Licencjobiorca może kopiować i rozprowadzać Dokument komercyjnie lub niekomercyjnie, w dowolnej postaci, pod warunkiem zamieszczenia na każdej kopii Dokumentu treści Licencji, informacji o prawie autorskim oraz noty mówiącej, że do Dokumentu ma zastosowanie niniejsza Licencja, a także pod warunkiem nie umieszczania żadnych dodatkowych ograniczeń, które nie wynikają z Licencji. Licencjobiorca nie ma prawa używać żadnych technicznych metod pomiarowych utrudniających lub kontrolujących czytanie lub dalsze kopiowanie utworzonych i rozpowszechnianych przez siebie kopii. Może jednak pobierać opłaty za udostępnianie kopii. W przypadku dystrybucji dużej liczby kopii Licencjobiorca jest zobowiązany przestrzegać warunków wymienionych w punkcie 3.

Licencjobiorca może także wypożyczać kopie na warunkach opisanych powyżej, a także wystawiać je publicznie.

3. Kopiowanie ilościowe

Jeżeli Licencjobiorca publikuje drukowane kopie Dokumentu w liczbie większej niż 100, a licencja Dokumentu wymaga umieszczenia Treści okładki, należy dołączyć kopie okładek, które zawierają całą wyraźną i czytelną Treść okładki : treść przedniej okładki, na przedniej okładce, a treść tylnej okładki, na tylnej okładce. Obie okładki muszą też jasno i czytelnie informować o Licencjobiorcy jako wydawcy tych kopii. Okładka przednia musi przedstawiać pełny tytuł; wszystkie słowa muszą być równie dobrze widoczne i czytelne. Licencjobiorca może na okładkach umieszczać także inne informacje dodatkowe. Kopiowanie ze zmianami ograniczonymi do

okładek, dopóki nie narusza tytułu Dokumentu i spełnia opisane warunki, może być traktowane pod innymi względami jako kopiowanie dosłowne.

Jeżeli napisy wymagane na którejś z okładek są zbyt obszerne, by mogły pozostać czytelne po ich umieszczeniu, Licencjobiorca powinien umieścić ich początek(taką ilość, jaka wydaje się rozsądna) na rzeczywistej okładce, a pozostałą część na sąsiednich stronach.

W przypadku publikowania lub rozpowszechniania Niejawnych kopii Dokumentu w liczbie większej niż 100, Licencjobiorca zobowiązany jest albo dołączyć do każdej z nich Jawną kopię czytelną dla komputera, albo wymienić w/lub przy każdej kopii Niejawnej publicznie dostępną w sieci komputerowej lokalizację pełnej kopii Jawnej Dokumentu, bez żadnych informacji dodanych — lokalizację, do której każdy użytkownik sieci miałby bezpłatny anonimowy dostęp za pomocą standardowych publicznych protokołów sieciowych. W przypadku drugim Licencjobiorca musi podjąć odpowiednie środki ostrożności, by wymieniona kopia Jawna pozostała dostępna we wskazanej lokalizacji przynajmniej przez rok od momentu rozpowszechnienia ostatniej kopii Niejawnej (bezpośredniego lub przez agentów albo sprzedawców) danego wydania.

Zaleca się, choć nie wymaga, aby przed rozpoczęciem rozpowszechniania dużej liczby kopii Dokumentu, Licencjobiorca skontaktował się z jego autorami celem uzyskania uaktualnionej wersji Dokumentu.

4. Modyfikacje

Licencjobiorca może kopiować i rozpowszechniać Zmodyfikowaną wersję Dokumentu na zasadach wymienionych powyżej w punkcie 2 i 3 pod warunkiem ścisłego przestrzegania niniejszej Licencji. Zmodyfikowana wersja pełni wtedy rolę Dokumentu, a więc Licencja dotycząca modyfikacji i rozpowszechniania Zmodyfikowanej wersji przenoszona jest na każdego, kto posiada jej kopię. Ponadto Licencjobiorca musi w stosunku do Zmodyfikowanej wersji spełnić następujące wymogi :

A. Użyć na Stronie tytułowej (i na okładkach, o ile istnieją) tytułu innego niż tytuł Dokumentu i innego niż tytuły poprzednich wersji (które, o ile istniały, powinny zostać wymienione w Dokumencie, w sekcji Historia). Tytułu jednej z ostatnich wersji Licencjobiorca może użyć, jeżeli jej wydawca wyrazi na to zgodę.

B. Wymienić na Stronie tytułowej, jako autorów, jedną lub kilka osób albo jednostek odpowiedzialnych za autorstwo modyfikacji Zmodyfikowanej wersji, a także przynajmniej pięciu spośród pierwotnych autorów Dokumentu (wszystkich, jeśli było ich mniej niż pięciu).

C. Umieścić na Stronie tytułowej nazwę wydawcy Zmodyfikowanej wersji.

D. Zachować wszelkie noty o prawach autorskich zawarte w Dokumencie.

E. Dodać odpowiednią notę o prawach autorskich dotyczących modyfikacji obok innych not o prawach autorskich.

F. Bezpośrednio po notach o prawach autorskich, zamieścić notę licencyjną zezwalającą na publiczne użytkowanie Zmodyfikowanej wersji na zasadach niniejszej Licencji w postaci podanej w Załączniku poniżej.

G. Zachować w nocie licencyjnej pełną listę Sekcji niezmiennych i wymaganych Treści okładki podanych w nocie licencyjnej Dokumentu.

H. Dołączyć niezmienioną kopię niniejszej Licencji.

I. Zachować sekcję zatytułowaną “Historia” oraz jej tytuł i dodać do niej informację dotyczącą przynajmniej tytułu, roku publikacji, nowych autorów i wydawcy Zmodyfikowanej wersji zgodnie z danymi zamieszczonymi na Stronie tytułowej. Jeżeli w Dokumencie nie istnieje sekcja pod tytułem “Historia”, należy ją utworzyć, podając tytuł, rok, autorów i wydawcę Dokumentu zgodnie z danymi zamieszczonymi na stronie tytułowej, a następnie dodając informację dotyczącą Zmodyfikowanej wersji, jak opisano w poprzednim zdaniu.

J. Zachować wymienioną w Dokumencie (jeśli taka istniała) informację o lokalizacji sieciowej, publicznie dostępnej Jawnej kopii Dokumentu, a także o podanych w Dokumencie lokalizacjach sieciowych poprzednich wersji, na których został on oparty. Informacje te mogą się znajdować w sekcji ”Historia”. Zezwala się na pominięcie lokalizacji sieciowej prac, które zostały wydane przynajmniej cztery lata przed samym Dokumentem, a także tych, których pierwotny wydawca wyraża na to zgodę.

K. W każdej sekcji zatytułowanej “Podziękowania” lub “Dedykacje” zachować tytuł i treść, oddając również ton każdego z podziękowań i dedykacji.

L. Zachować wszelkie Sekcje niezmienne Dokumentu w niezmienionej postaci (dotyczy zarówno treści, jak i tytułu). Numery sekcji i równoważne im oznaczenia nie są traktowane jako należące do tytułów sekcji.

M. Usunąć wszelkie sekcje zatytułowane “Adnotacje”. Nie muszą one być załączane w Zmodyfikowanej wersji.

N. Nie nadawać żadnej z istniejących sekcji tytułu “Adnotacje” ani tytułu pokrywającego się z jakąkolwiek Sekcją niezmienną.

Jeżeli Zmodyfikowana wersja zawiera nowe sekcje początkowe lub dodatki stanowiące Sekcje drugorzędne i nie zawierające materiału skopiowanego z Dokumentu, Licencjobiorca może je lub ich część oznaczyć jako sekcje niezmienne. W tym celu musi on dodać ich tytuły do listy Sekcji niezmiennych zawartej w nocie licencyjnej Zmodyfikowanej wersji. Tytuły te muszą być różne od tytułów pozostałych sekcji.

Licencjobiorca może dodać sekcję “Adnotacje”, pod warunkiem, że nie zawiera ona żadnych treści innych, niż adnotacje dotyczące Zmodyfikowanej wersji — mogą to być na przykład stwierdzenia o recenzji koleżeńskiej albo o akceptacji tekstu przez organizację jako autorytatywnej definicji standardu.

Na końcu listy Treści okładki w Zmodyfikowanej wersji, Licencjobiorca może dodać fragment “do umieszczenia na przedniej okładce” o długości nie przekraczającej pięciu słów, a także fragment o długości do 25 słów “do umieszczenia na tylnej okładce”. Przez każdą jednostkę (lub na mocy ustaleń przez nią poczynionych) może zostać dodany tylko jeden fragment z przeznaczeniem na przednią okładkę i jeden z przeznaczeniem na tylną. Jeżeli Dokument zawiera już treść okładki dla danej okładki, dodaną uprzednio przez Licencjobiorcę lub w ramach ustaleń z jednostką, w imieniu której działa Licencjobiorca, nowa treść okładki nie może zostać dodana. Dopuszcza się jednak zastąpienie poprzedniej treści okładki nową pod warunkiem wyraźnej zgody poprzedniego wydawcy, od którego stara treść pochodzi.

Niniejsza Licencja nie oznacza, iż autor (autorzy) i wydawca (wydawcy) wyrażają zgodę na publiczne używanie ich nazwisk w celu zapewnienia autorytetu jakiejkolwiek Zmodyfikowanej wersji.

5. Łączenie dokumentów

Licencjobiorca może łączyć Dokument z innymi dokumentami wydanymi na warunkach niniejszej Licencji, na warunkach podanych dla wersji zmodyfikowanych w części 4 powyżej, jednak tylko wtedy, gdy w połączeniu zostaną zawarte wszystkie Sekcje niezmienne wszystkich oryginalnych dokumentów w postaci niezmodyfikowanej i gdy będą one wymienione jako Sekcje niezmienne połączenia w jego nocie licencyjnej.

Połączenie wymaga tylko jednej kopii niniejszej Licencji, a kilka identycznych Sekcji niezmiennych może zostać zastąpionych jedną. Jeżeli istnieje kilka Sekcji niezmiennych o tym samym tytule, ale różnej zawartości, Licencjobiorca jest zobowiązany uczynić tytuł każdej z nich unikalnym poprzez dodanie na jego końcu, w nawiasach, nazwy oryginalnego autora lub wydawcy danej sekcji, o ile jest znany, lub unikalnego numeru. Podobne poprawki wymagane są w tytułach sekcji na liście Sekcji niezmiennych w nocie licencyjnej połączenia.

W połączeniu Licencjobiorca musi zawrzeć wszystkie sekcje zatytułowane "Historia" z dokumentów oryginalnych, tworząc jedną sekcję "Historia". Podobnie ma postąpić z sekcjami "Podziękowania" i "Dedykacje". Wszystkie sekcje zatytułowane "Adnotacje" należy usunąć.

6. Zbiory dokumentów

Licencjobiorca może utworzyć zbiór składający się z Dokumentu i innych dokumentów wydanych zgodnie z niniejszą Licencją i zastąpić poszczególne kopie Licencji pochodzące z tych dokumentów jedną kopią dołączoną do zbioru, pod warunkiem zachowania zasad Licencji dotyczących kopii dosłownych we wszelkich innych aspektach każdego z dokumentów.

Z takiego zbioru Licencjobiorca może wyodrębnić pojedyńczy dokument i rozpowszechniać go niezależnie na zasadach niniejszej Licencji, pod warunkiem zamieszczenia w wyodrębnionym dokumencie kopii niniejszej Licencji oraz zachowania zasad Licencji we wszystkich aspektach dotyczących dosłownej kopii tego dokumentu.

7. Zestawienia z pracami niezależnymi

Kompilacja Dokumentu lub jego pochodnych z innymi oddzielnymi i niezależnymi dokumentami lub pracami nie jest uznawana za Zmodyfikowaną wersję Dokumentu, chyba że odnoszą się do niej jako do całości prawa autorskie. Taka kompilacja jest nazywana zestawieniem, a niniejsza Licencja nie dotyczy samodzielnych prac skompilowanych z Dokumentem, jeśli nie są to pochodne Dokumentu.

Jeżeli do kopii Dokumentu odnoszą się wymagania dotyczące Treści okładki wymienione w części 3 i jeżeli Dokument stanowi mniej niż jedną czwartą całości zestawienia, Treść okładki Dokumentu może być umieszczona na okładkach zamykających Dokument w obrębie zestawienia. W przeciwnym razie Treść okładki musi się pojawić na okładkach całego zestawienia.

8. Tłumaczenie

Tłumaczenie jest uznawane za rodzaj modyfikacji, a więc Licencjobiorca może rozpowszechniać tłumaczenia Dokumentu na zasadach wymienionych w punkcie 4. Zastąpienie Sekcji niezmiennych ich tłumaczeniem wymaga specjalnej zgody właścicieli prawa autorskiego. Dopuszcza się jednak zamieszczanie tłumaczeń wybranych lub wszystkich Sekcji niezmiennych obok ich wersji oryginalnych. Podanie tłumaczenia niniejszej Licencji możliwe jest pod warunkiem zamieszczenia także jej oryginalnej wersji angielskiej. W przypadku niezgodności pomiędzy zamieszczonym tłumaczeniem a oryginalną wersją angielską niniejszej Licencji moc prawną ma oryginalna wersja angielska.

9. Wygaśnięcie

Poza przypadkami jednoznacznie dopuszczonymi na warunkach niniejszej Licencji nie zezwala się Licencjobiorcy na kopiowanie, modyfikowanie, czy rozpowszechnianie Dokumentu ani też na cedowanie praw licencyjnych. We wszystkich pozostałych wypadkach każda próba kopiowania, modyfikowania lub rozpowszechniania Dokumentu albo cedowania praw licencyjnych jest nieważna i powoduje automatyczne wygaśnięcie praw, które licencjobiorca nabył z tytułu Licencji. Niemniej jednak w odniesieniu do stron, które już otrzymały od Licencjobiorcy kopie albo prawa w ramach niniejszej Licencji, licencje nie zostaną anulowane, dopóki strony te w pełni się do nich stosują.

10. Przyszłe wersje Licencji

W miarę potrzeby Free Software Foundation może publikować nowe poprawione wersje GNU Free Documenation License. Wersje te muszą pozostawać w duchu podobnym do wersji obecnej, choć mogą się różnić w szczegółach dotyczących nowych problemów czy zagadnień. Patrz http://www.gnu.org/copyleft/.

Każdej wersji niniejszej Licencji nadaje się wyróżniający ją numer. Jeżeli w Dokumencie podaje się numer wersji Licencji, oznaczający, iż odnosi się do niego podana "lub jakakolwiek późniejsza" wersja licencji, Licencjobiorca ma do wyboru stosować się do postanowień i warunków albo tej wersji, albo którejkolwiek wersji późniejszej opublikowanej oficjalnie (nie jako propozycja) przez Free Software Foundation. Jeśli Dokument nie podaje numeru wersji niniejszej Licencji, Licencjobiorca może wybrać dowolną wersję kiedykolwiek opublikowaną (nie jako propozycja) przez Free Software Foundation.

Spis treści

I GNU Free Documentation License iii

II Licencja GNU Wolnej Dokumentacji xi

III Wstęp 1

1 Wstęp 3
- 1.1 Dlaczego Linux? . . . 3
  - 1.1.1 Linux jest za darmo . . . 3
  - 1.1.2 Linux pracuje na komputerach PC . . . 3
  - 1.1.3 Małe wymagania sprzętowe . . . 4
- 1.2 Co będzie potrzebne? . . . 4

2 Instalacja systemu 7
- 2.1 Instalacja serwera . . . 7
- 2.2 Instalacja stacji roboczej . . . 11
  - 2.2.1 Instalacja przez sieć . . . 13

IV Wiersz poleceń 15

3 Podstawy systemu Linux 17
- 3.1 Początki wielkiej przyjaźni . . . 17
- 3.2 Hasło - klucz do świata magii . . . 18
- 3.3 Tajemniczy root . . . 19

4 Shell - język zaklęć 21
- 4.1 Podstawowe zaklęcia . . . 21
- 4.2 Pliki i katalogi . . . 23
  - 4.2.1 Wędrówki po systemie . . . 23
  - 4.2.2 Tworzenie i niszczenie . . . 24
- 4.3 Co wolno wojewodzie... . . . 25
- 4.4 System plików, dyski . . . 28
- 4.5 Edytor vim . . . 29
- 4.6 Poprawianie bash'a - customizacja . . . 31
- 4.7 Zmienne środowiskowe . . . 32
- 4.8 Strumienie i potoki . . . 35

4.8.1 Strumienie . . . 35
4.8.2 Potoki . . . 37
4.9 Poszukiwany, poszukiwana . . . 37
4.10 Demony . . . 39
4.10.1 Cron Daemon . . . 39
4.10.2 At Daemon . . . 40

V Administracja 43

5 Konta użytkowników 45
5.1 Właściwości konta . . . 45
5.2 Konta pełne . . . 48
5.3 Konta bezszelowe . . . 50
5.4 Kasowanie kont . . . 51
5.5 Grupy użytkowników . . . 52

6 Konfiguracja sieci 55
6.1 Adresy w sieci IP . . . 55
6.2 Konfiguracja interfejsów . . . 59
6.2.1 Karty sieciowe . . . 59
6.2.2 Modemy . . . 63
6.2.3 Karty PCMCIA . . . 63
6.3 Konfiguracja routingu . . . 63
6.4 Resolver . . . 63
6.5 Synchronizacja czasu . . . 65
6.5.1 Synchronizacja okresowa . . . 65
6.5.2 Serwer czasu . . . 65

7 Instalacja aplikacji 69
7.1 Menadżer pakietów RPM . . . 69
7.2 Źródłowe pakiety src.rpm . . . 72
7.3 Zarządzanie pakietami rpm - apt . . . 73
7.3.1 Apt z wiersza poleceń . . . 74
7.3.2 Aktualizacja systemu . . . 77
7.3.3 Synaptic — apt w środowisku graficznym . . . 77
7.3.4 Prywatne repozytorium . . . 77
7.4 Instalacja “ze źródeł” . . . 79
7.5 Budujemy pakiet rpm . . . 82

8 Archiwizacja 89

9 Kopia bezpieczeństwa 91

10 Jądro systemu 93
10.1 Kernel z dystrybucji . . . 93
10.2 Jądro ”waniliowe” . . . 93
10.3 Kompilacja jądra . . . 93
10.4 Kernel 2.6.x w Auroksie 9.x . . . 93

11 Zapora ogniowa **95**
11.1 Zasada działania . . . 95
11.2 Prace przygotowawcze . . . 97
11.3 Polityka bezpieczeństwa . . . 98
11.4 Dostęp z sieci wewnętrznej . . . 100
11.5 Dostęp z internetu . . . 101
11.6 Prace końcowe . . . 103

VI Usługi serwerowe **105**

12 DNS - serwer nazw domenowych **107**
12.1 Konfiguracja serwera . . . 109
12.2 Pliki stref . . . 111
12.3 Pliki stref odwrotnych . . . 112
12.4 Testowanie DNS . . . 113

13 Serwer SSH **115**
13.1 Podstawowa konfiguracja . . . 115
13.2 Autoryzacja na bazie kluczy . . . 118
13.3 Tunelowanie usług . . . 119

14 Serwer pocztowy **123**
14.1 Podstawowa konfiguracja serwera sendmail . . . 123
14.2 Autoryzacja SMTP . . . 127
14.3 Czytanie poczty . . . 129
14.4 Poczta przez www . . . 130
14.5 Ochrona antywirusowa i antyspamowa . . . 131
14.5.1 Wirusy . . . 131
14.5.2 Spam . . . 132
14.5.3 Łączymy części składowe . . . 133

15 DHCP - dynamiczna konfiguracja hostów **135**
15.1 Trochę teorii . . . 135
15.2 Instalacja i konfiguracja . . . 135
15.3 Dynamiczny DNS . . . 136
15.3.1 Modyfikacja serwera DNS . . . 136
15.3.2 Modyfikacja serwera DHCP . . . 138
15.4 Konfiguracja usług pokrewnych . . . 139
15.4.1 Logi systemowe . . . 139
15.4.2 Rotacja logów . . . 139
15.5 Zaawansowane opcje konfiguracyjne . . . 139

16 Serwer ftp - vsftp **143**
16.1 Konfiguracja podstawowa . . . 143
16.2 Prawa dostępu . . . 144
16.3 Dostęp anonimowy . . . 145
16.4 Środowisko chrootowane . . . 146

17 Serwer stron www **149**
17.1 Konfiguracja podstawowa . . . 149
17.2 Strony prywatne użytkowników . . . 152
17.3 Serwery wirtualne . . . 153
17.4 Logi serwera www, Webalizer . . . 155
17.5 Moduły serwera . . . 155
17.5.1 Moduł php . . . 155
17.5.2 Moduł mod_perl . . . 155

18 Serwer plików **157**
18.1 Usługa NFS . . . 157
18.1.1 Polecenie sudo . . . 159
18.2 Klient sieci smbfs . . . 160
18.3 Serwer sieci smbfs . . . 160
18.4 Serwer w sieci NT . . . 160

19 Udostępnianie internetu **161**
19.1 Maskarada i przekazywanie pakietów . . . 161
19.2 Serwer proxy . . . 165
19.3 Podział pasma . . . 165

20 Baza danych **167**
20.1 Instalacja i konfiguracja serwera mysqld . . . 167
20.2 Podstawy administracji serwerem mysql . . . 172

VII Wizytówka pracowni **179**

21 CMS - systemy zarządzania treścią **181**
21.1 Pakiet PHPNuke . . . 181
21.2 Pakiet tikiwiki . . . 181

VIII Dodatki **183**

A Filmy w Auroksie **185**
A.1 Instalacja . . . 185
A.2 Konfiguracja . . . 185
A.3 Eksploatacja . . . 185

B Komputerowe biuro - OpenOffice **187**
B.1 Instalacja . . . 187
B.2 Konfiguracja . . . 187
B.2.1 Współpraca z MySql . . . 187
B.3 Składniki pakietu . . . 187
B.3.1 Edytor tekstu . . . 187
B.3.2 Arkusz kalkulacyjny . . . 187
B.3.3 Prezentacje . . . 187

Część III

Wstęp

Rozdział 1

Wstęp

1.1 Dlaczego Linux?

Linux został stworzony na początku lat dziewięćdziesiątych przez Linusa Torvaldsa i jest dalej rozwijany przez programistów z całego świata. Jako system operacyjny spełnia wszystkie funkcje, które wymagane są w przypadku systemów wielodostępnych oraz stacji roboczych, ale najbardziej ceniony jest za siłę, elastyczność i stabilność. Linux jest wersją systemu operacyjnego UNIX dla komputerów PC, który przez dziesiątki lat był wykorzystywany na dużych komputerach. Został tak skonstruowany, że wprawny czarodziej może z nim zrobić prawie wszystko, ma pełną kontrolę nad jego funkcjonowaniem. System Linux nie stara się być mądrzejszy od swojego administratora, choć niektóre dystrybucje nastawione na komercję hołdują hasłu "click and run".

1.1.1 Linux jest za darmo

W odróżnieniu od systemów komercyjnych, Linux jest rozpowszechniany za darmo na zasadach licencji GNU, określonej przez Free Software Foundation. Licencja ta zastrzega prawa autorskie, jednak jest tak skonstruowana, że każdy może korzystać z oprogramowania udostępnionego na tej licencji. Zastosowanie licencji GNU ma zapewnić, że Linux pozostanie darmowy i zarazem zestandaryzowany. Nie można jednak zapominać, że za darmo jest tylko kod źródłowy, natomiast wersje skompilowane i poukładane w pakiety mogą stanowić przedmiot obrotu handlowego. W związku z tym obok darmowych dystrybucji Linuksa powstają również dystrybucje komercyjne zapewniające profesjonalne wsparcie, ale to kosztuje. Istnieje tylko jeden Linux dostępny w wielu dystrybucjach. Na potrzeby naszego kursu wykorzystamy dystrybucję Aurox Linux.

1.1.2 Linux pracuje na komputerach PC

Dzięki temu, że Linux został stworzony do pracy na komputerach PC mamy możliwość zapoznania się z zasadami pracy w prawdziwych systemach operacyjnych, które do niedawna dostępne były tylko na dużych maszynach, tym samym niedostępne dla przeciętnego zjadacza chleba. Teraz możesz poznać zasady czarodziejstwa wykorzystując komputer domowy, na który Cię stać.

1.1.3 Małe wymagania sprzętowe

Linux ma wyjątkowo małe wymagania sprzętowe. Pracę na konsoli tekstowej można wykonywać na komputerze z procesorem Intel 486 i 8MB RAM, a do instalacji systemu wystarczy 300MB miejsca na twardym dysku. Oczywiście to nie jest to, co tygryski lubią najbardziej, ale na początek... Jeśli chcesz pracować w środowisku graficznym, z ikonami, okienkami itp. to będziesz potrzebował trochę lepszej maszyny, szczególnie jeśli chcesz korzystać z najnowszych wersji oprogramowania.

1.2 Co będzie potrzebne?

Aby nauczyć się podstaw czarodziejstwa będziesz potrzebował kilku rzeczy :

1. komputer — jak wcześniej pisałem, do większości prac przewidzianych w niniejszym podręczniku wystarczy stara, wysłużona maszynka; dla wydajnej pracy polecałbym jednak komputer z procesorem Intel Pentium II 360MHz i 128MB RAM lub lepszy z dyskiem twardym 4GB lub większym

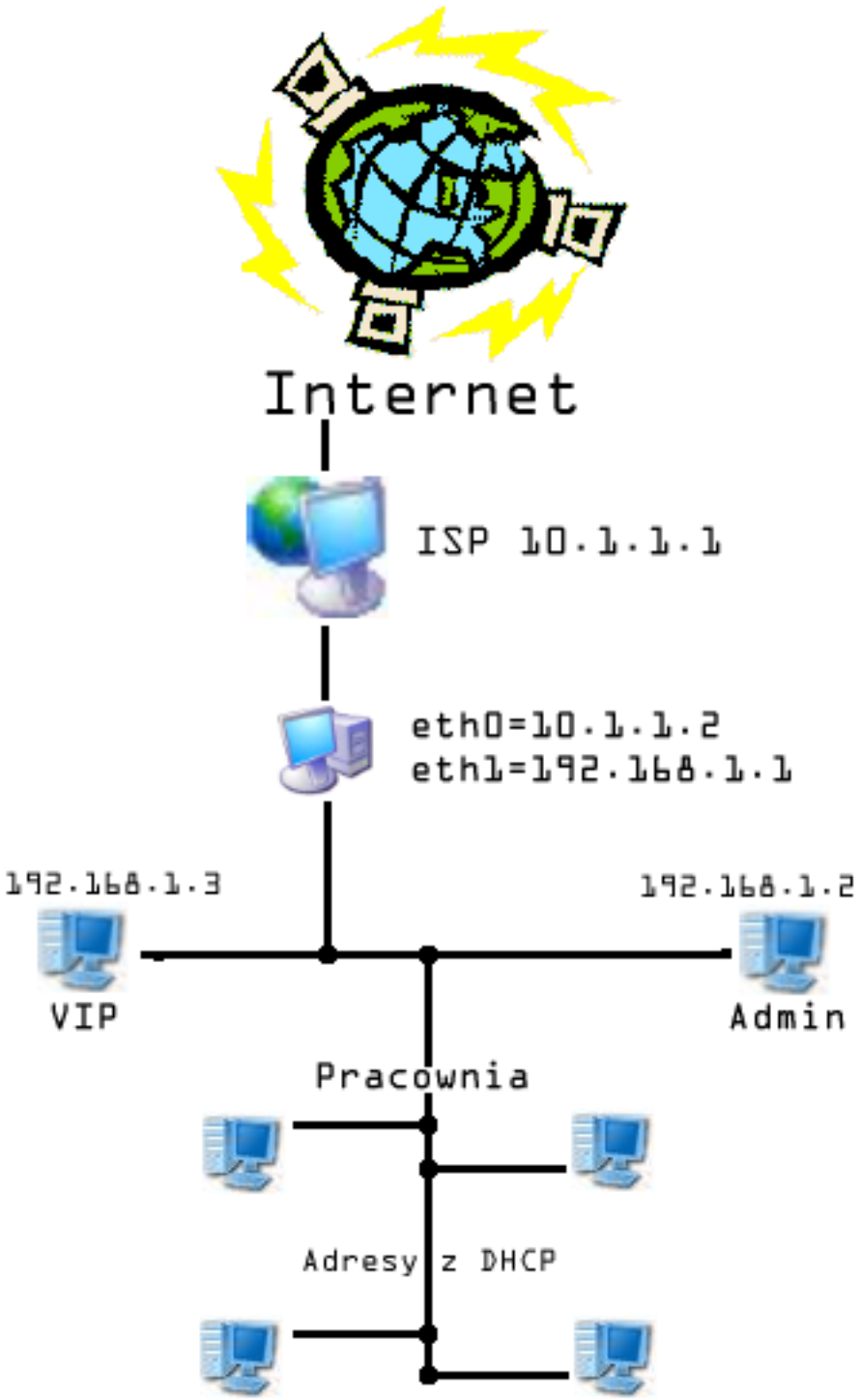
2. dyskietki, CD — do instalacji oprogramowania niezbędny będzie napęd CD-ROM, a do archiwizacji pracy oraz przenoszenia jej efektów na inne komputery potrzebne będą dyskietki (zakładam, że komputer w domu nie ma połączenia z komputerem w szkole lub pracy)

I to właściwie wszystko. Komputery w szkole lub pracowni komputerowej powinny być wyposażone w karty sieciowe tak, by można było przeprowadzić ćwiczenia dotyczące pracy w sieci komputerowej.

I co teraz?

To już wszystko. Jeśli na Twoim komputerze nie ma zainstalowanego Linuksa, to masz drobny problem, będziesz musiał zacząć od instalacji. W kolejnym rozdziale opisany jest sposób instalacji systemu, ale jeśli chcesz go zainstalować jako drugi system na stacji, na której już masz jakieś ważne dane, to radziłbym poprosić o pomoc kogoś, kto ten proces ma już za sobą. Linux pozwala Ci zrobić prawie wszystko, ale jeśli nie wiesz co robisz, to z Twoich działań może wyniknąć więcej szkód niż pożytku.
PAMIĘTAJ!!! Zanim naciśniesz `ENTER` zastanów się dwa razy.

Ponieważ najlepsze efekty w nauczaniu można uzyskać bazując na przykładach, my również zastosujemy taką zasadę. Za cel przyjmujemy uruchomienie pracowni komputerowej, której schemat przedstawia rysunek 1.1.



Rysunek 1.1: Nasza sieć

Rozdział 2

Instalacja systemu

2.1 Instalacja serwera

Budowę naszej pracowni zaczniemy od instalacji systemu na serwerze. Ponieważ na maszynie tej nie będą bezpośrednio pracować użytkownicy możemy przyjąć, że będzie to maszyna jedno-systemowa. Upraszcza nam to nieco proces instalacji, gdyż nie musimy się martwić o tworzenie wydzielonej przestrzeni dla naszego sytemu, lecz zagospodarowujemy całe dyski. Kolejnym elementem, który możemy pominąć na tym etapie jest instalacja i konfiguracja środowiska graficznego (XWindow).

Zaczynamy od sprawdzenia, czy w BIOSie komputera mamy ustawione startowanie systemu z krążków CD. Jeśli tak to startujemy komputer z krążkiem CD 1 w napędzie. W przypadku dystrybucji Aurox Linux, którą będziemy wykorzystywać, pojawi się okienko powitalne z krótkim opisem możliwych do wybrania opcji. U dołu ekranu pojawi się napis `boot:`, po którym możemy podać odpowiednie parametry pracy jądra systemu w czasie instalacji, jak też samego instalatora. Wciśnięcie klawisza `ENTER` bez podania żadnych opcji uruchamia instalator z domyślnymi ustawieniami. Prace nad jądrem systemu i sterownikami pozostają zawsze nieco w tyle za produkcją sprzętu, więc może się zdarzyć, że instalator nie wystartuje przy domyślnych ustawieniach. Najczęściej spotykanym problemem jest brak obsługi karty graficznej. Przy domyślnych ustawieniach instalator próbuje uruchomić środowisko graficzne w rozdzielczości 640x480 pikseli. Mimo tak małych wymagań zdarza się, że uruchomienie nie powiedzie się. Dlatego do instalacji serwera polecam uruchamianie instalatora w trybie tekstowym wpisując `boot: linux text`.

Na pytanie o typ instalacji odpowiadamy zaznaczając opcję "Decyduj sam".

Uwaga. W przypadku instalacji serwera niczego nie pozostawiamy automatom, zawsze wybieramy pozycje umożliwiające pełną kontrolę nad działaniem programu instalacyjnego.

Zatrzymajmy się teraz przy podziale dysku/dysków na partycje. W zasadzie można zainstalować cały system na jednej partycji obejmującej całą przestrzeń dysku. Ale tu pojawia się pierwszy problem - a jeśli mamy kilka dysków? Istnieją jeszcze inne argumenty przemawiające za podziałem. Jeśli w przyszłości będziemy chcieli wykonać aktualizację systemu, to warto by było zachować dane użytkowników, witryny, bazy danych itp. Również względy bezpieczeństwa przemawiają za podziałem. Istnieją metody ataku doprowadzające do zapchania dysku. Jeśli cały system umieścimy na jednej partycji, to tracimy w takim przypadku dostęp do systemu. Przy podziale na partycje i zapełnieniu jednej z nich pewne aplikacje mogą odmówić posłuszeń-

stwa, ale system będzie działał i będziemy mogli podjąć odpowiednie kroki dla przywrócenia stabilności.

Jak zatem podzielić dostępną przestrzeń dyskową? Oto propozycja podziału dla naszego serwera :

1. /boot – to powinna być pierwsza utworzona przez nas partycja, powinna znajdować się na początku przestrzeni dyskowej pierwszego dysku jako partycja podstawowa. Partycja ta zawiera dane programu uruchomieniowego (bootloadera), jądro systemu i jeszcze kilka innych elementów istotnych dla startu systemu plików. Wprawdzie w wielu dokumentacjach można się spotkać z twierdzeniem, że wymagania podane wcześniej nie mają obecnie zastosowania, że było to istotne na starszym sprzęcie i starszym oprogramowaniu, ale praktyka mówi co innego. Partycja `/boot` powinna mieć rozmiar 10-70MB zależnie od tego ile możesz na nią poświęcić miejsca na dysku oraz ile wersji jądra systemu chcesz przechowywać. Formatujemy ją jako `ext2` lub `ext3` (to rodzaje systemów plików tak jak `FAT` czy `NTFS`).

2. swap – to specjalny rodzaj partycji. Partycje tego typu nie są dostępne dla użytkowników systemu, Linux wykorzystuje przestrzeń dyskową swapa jako wirtualną pamięć w sytuacjach gdy brakuje fizycznej pamięci RAM. Rozmiar tej partycji trudno jednoznacznie określić. W środowisku pokutuje twierdzenie, że rozmiar partycji swap powinien wynosić 2x wielkość RAM-u. Nic bardziej błędnego. Rozmiar ten musimy określić na podstawie posiadanej pamięci RAM oraz przewidywanego obciążenia naszego systemu. Jeśli mamy mało pamięci fizycznej, a przewidujemy duże obciążenie serwera www czy serwera poczty, to warto przeznaczyć więcej dysku na swap. W przypadku, gdy systemowi brakuje pamięci RAM i skończy się swap, Linux zaczyna zamykać aplikacje, które uzna za najmniej istotne dla pracy całości. Do takiej sytuacji nie wolno nam dopuścić.

   Uwaga. Pamiętaj, że swap nie zastępuje pamięci RAM. Traktuj go jako wyjście awaryjne. Jeśli zauważysz, że system notorycznie korzysta z pamięci wirtualnej, to z całą pewnością zakup dodatkowej pamięci RAM nie będzie wyrzucaniem pieniędzy w błoto.

   Niektórzy doświadczeni administratorzy twierdzą, że swap nie może być mniejszy niż pamięć RAM. W tym twierdzeniu jest trochę racji, gdyż w przypadku nagłego i niekontrolowanego padu systemu w swapie zapisywane są informacje znajdujące się w pamięci w chwili awarii. Po podniesieniu uszkodzonego systemu można się dostać do tych danych i przeanalizować co się stało. Jednak taka funkcjonalność wymaga bardzo dużego doświadczenia i znajomości systemu, więc my możemy sobie podarować takie podejście.
   W przypadku serwera zalecałbym następującą konfigurację swapa :

   - <256MB RAM – 2x RAM
   - 256MB - 1024MB RAM – 1x RAM
   - >1GB RAM – 512MB

   Istotne jest również umieszczenie partycji swap we właściwym miejscu dysku. Ponieważ najszybciej czytane są sektory dysku znajdujące się na jego zewnętrznych cylindrach, to warto partycję swap umieszczać na początku listy partycji tak, aby znalazła się możliwie blisko początku dysku.

3. /home – to partycja, w której znajdować się będą katalogi domowe użytkowników systemu i ich dane. Rozmiar tej partycji możemy ustalić na podstawie przewidywanej liczby

użytkowników i określeniu rozmiaru katalogu domowego przeciętnego użytkownika. Tak wyliczony rozmiar należy powiększyć o ok. 15%, które wykorzysta system oraz jako rezerwa. Partycję /home formatujemy jako `ext3`.

Uwaga. W przypadku aktualizacji systemu partycji tej **nie formatujemy**, gdyż stracilibyśmy dane znajdujące się na niej. W instalatorze wybieramy edycję takiej partycji i określamy jedynie punkt montowania bez formatowania.

---

4. /var – jest to jedna z bardziej obciążonych partycji w systemie. W katalogu /var znajdować się będą logi systemowe, kolejki zadań czekających na wykonanie (np. kolejki wydruków, poczta do wysłania), skrzynki pocztowe użytkowników, strony www, serwer ftp, bazy danych itp. Stąd wniosek, że partycja ta powinna być jedną z większych partycji w systemie. Oczywiście należy wziąć pod uwagę czynniki specyficzne dla danego systemu. Strony www możemy umieścić w innym katalogu, a strony użytkowników mogą być umieszczane w ich katalogach domowych. Bazy danych także możemy umieścić w innym katalogu. Wielkość logów systemowych zależy od obciążenia systemu oraz ich szczegółowości, jak też przyjętej polityce rotacji logów.

   W przypadku aktualizacji systemu warto zachować tę partycję bez formatowania.

5. /tmp – partycja ta przeznaczona jest na przechowywanie plików tymczasowych. Wydzielamy ją głównie ze względów bezpieczeństwa.

6. /usr – partycja zawierająca aplikacje instalowane z pakietów dystrybucyjnych, aplikacje instalowane ze źródeł oraz dokumentacje do tych aplikacji. Partycji przydzielamy 4-6GB i formatujemy jako `ext3`.

7. / – partycja ta nazywana partycją root zawiera początek drzewa katalogów, katalog domowy użytkownika root oraz katalogi zawierające aplikacje najistotniejsze dla funkcjonowania systemu i pliki konfiguracyjne. Partycji przydzielamy 1-2GB i formatujemy jako `ext3`

Oczywiście powyższy opis należy traktować jedynie jako wskazówki. Podział przestrzeni dyskowej jest najtrudniejszym elementem procesu instalacji systemu, gdyż wymaga znajomości całego systemu i przewidywania jego obciążenia. Istotnym elementem jest również przeznaczenie systemu oraz ilość i wielkość dostępnych dysków.

Opis powyższy nie uwzględnia stosowania bardziej zaawansowanych technik takich jak macierze dyskowe RAID zwiększające wydajność systemu i bezpieczeństwo danych oraz stosowanie technologii LVM (Logical Volume Manager) zwiększających elastyczność systemu plików i ich rozbudowę.

Teraz kolej na konfigurację programu uruchomieniowego nazywanego bootloaderem. W przypadku Auroksa natywnym bootloaderem jest `GRUB`, więc pozycję tą pozostawiamy bez zmian. Miejsce instalacji bootloadera wydaje się być oczywiste w przypadku serwera. Instalujemy go w sektorze rozruchowym (MBR) pierwszego dysku (`/dev/hda`). Zaawansowanych opcji nie będziemy w tej chwili ustawiali, możemy to zrobić później modyfikując plik konfiguracyjny.

Kolejnym krokiem, na który należy zwrócić uwagę jest konfiguracja sieci. Dla potrzeb naszego podręcznika przyjmuję, że interfejs eth0 serwera będzie miał adres 192.168.1.1, maska sieci 255.255.255.0, gateway 192.168.1.1 (to później zmienimy), DNS 194.204.159.1 (to jest serwer kochanej TPSA, ale Ty podaj taki, jaki zaleci Ci dostawca usług internetowych nazywany ISP [Internet Service Provider]). Nazwa dla naszego serwera to *dreptak.domek.pl*, ale Ty ustaw ją

według własnych upodobań lub danych otrzymanych od dostawcy internetu.
Dla interfejsu eth1 należy podać informacje dostarczone przez dostawcę internetu (providera). Na potrzeby podręcznika przyjmuję następujące parametry :

- adres interfejsu – 10.1.1.2
- maska sieci – 255.255.255.252
- gateway – 10.1.1.1

Na pytanie o firewall odpowiadamy "Bez firewalla". Ten wizard robi masę błędów, więc ognimurek postawimy ręcznie zachowując pełną kontrolę nad jego funkcjonowaniem.

Dla poprawnego funkcjonowania serwera istotne jest właściwe ustawienie czasu. Instalator proponuje nam wybór strefy czasowej "Europa/Warszawa" i bardzo dobrze. W przypadku serwera należy jeszcze zaznaczyć jedną opcję. Otóż poważne systemy rozróżniają czas sprzętowy - pochodzący z BIOSu, oraz czas systemowy. Wybór strefy czasowej wpłynie na ustawienie zegara czasu systemowego jednak w przypadku serwera czas sprzętowy powinien być ustawiony w sposób umożliwiający komunikację z serwerami z całego świata. Takim uniwersalnym czasem jest GMT (Greenwich Mean Time). Aby dostosować naszą instalację do tych wymogów zaznaczamy opcję "Hardware to UTC".

Pozostaje nam jeszcze ustawienie hasła roota. Ponieważ root jest superużytkownikiem w naszym Auroksie i w systemie może wszystko, to dostęp do tego konta powinien być szczególnie chroniony. W tej chwili możesz podać dowolne hasło o długości sześciu lub więcej znaków. Pamiętaj jednak, aby po zakończeniu prac instalacyjnych hasło to ustawić zgodnie z wymogami bezpieczeństwa.

I wreszcie przechodzimy do instalacji oprogramowania. Pierwszym krokiem w tym procesie jest wybór grup pakietów do zainstalowania. W przypadku serwera nie będziemy instalować żadnych elementów graficznych czy multimedialnych, więc wyłączamy pozycje takie jak XFree, KDE, Gnome, Internet w środowisku graficznym itp. Warto natomiast zaznaczyć na tym etapie serwery www, ftp i poczty. Pod listą grup pakietów mamy opcję "wybierz poszczególne pakiety", zaznaczamy ją i przechodzimy do precyzyjnego określania co ma zostać zainstalowane. Jest to najbardziej czasochłonny element procesu instalacji i każdy administrator realizuje go w nieco inny sposób. W tym miejscu mogę jedynie polecić kilka pakietów, które z całą pewnością będą nam potrzebne :

- rpm, rpm-build, rpmdb – aplikacje pozwalające na zarządzanie pakietami, instalowanie i usuwanie pakietów
- apt – menadżer pakietów wykorzystujący rpm, umożliwia automatyzację procesu aktualizacji systemu, zapewnia zachowanie zależności między pakietami
- httpd, mod_ssl, vsftpd – serwer www i moduł umożliwiający połączenia szyfrowane oraz serwer ftp
- php – język programowania dynamicznych stron www, warto również wybrać jego rozszerzenia php-imap i php-mysql
- mysql, mysql-server – serwer baz danych
- perl, bash, mc, vim – bez tych aplikacji raczej trudno sobie wyobrazić pracę na serwerze
- gcc – kompilator języka C, z całą pewnością będziesz z niego korzystał

Po wybraniu przycisku "Dalej" zobaczymy listę pakietów do instalacji zmodyfikowaną przez instalator w taki sposób, aby zachować zależności między pakietami. Warto tę listę przejrzeć. Jeśli zauważymy pakiety, których nie chcemy mieć na swoim serwerze, to możemy cofnąć się i zmodyfikować wybór. Należy jednak pamiętać, że niektóre pakiety muszą być zainstalowane, aby inne, te wybrane przez nas, mogły poprawnie funkcjonować.

Po dłuższym czasie, kiedy jedynym zajęciem będzie wymienianie krążków CD, dojdziemy do pytania o utworzenie dyskietki startowej. Warto ją wykonać, chociaż można to również zrobić już po uruchomieniu systemu. Konfigurację X Window pomijamy i zaznaczamy, aby system startował w trybie tekstowym.

Teraz następuje to, na co czekaliśmy. Instalator zostaje zamknięty, krążek wysunięty i następuje restart maszyny.

2.2 Instalacja stacji roboczej (Michał Górniak)

Kiedy już na serwerze mamy zainstalowany system (albo właśnie się instaluje i chcemy zaoszczędzić czasu nastawiając kolejne instalacje) powinniśmy zająć się zainstalowaniem systemu na stacji roboczej. Zakładając, że na komputerze mamy inny system i nie chcemy go usuwać należy "zabrać" trochę miejsca z jego partycji. Uruchamiamy tamten system i robimy miejsce dla Linuksa, tak minimum 4-5 GB. Polecam więcej, aby była możliwość zainstalowania potrzebnych programów (wszystko zależy od tego jak duży mamy dysk twardy). Ponieważ Linux nie instaluje się na partycjach FAT (można to zrobić, ale wymaga to nieco więcej wiedzy niż posiadamy w tej chwili) musimy zmniejszyć ich rozmiar zostawiając na dyskach wolną przestrzeń. Zanim jednak do tego kroku przejdziemy należy wykonać defragmentację dysków tak, aby wszystkie dane znajdowały się na początku partycji, którą będziemy modyfikować. Teraz restartujemy komputer i sprawdzamy w BIOS-ie czy mamy ustawione startowanie systemu z płyt CD. Jeśli tak, jeszcze raz restartujemy komputer i wkładamy krążek CD 5 instalacji Auroksa (jest to System Rescue CD). Pokaże się nam menu wyboru opcji startowych. Naciskamy `ENTER` , wybieramy "Keymap selectin:" i wpisujemy liczbę 28 (polska klawiatura). Jesteśmy zalogowani. Wydajemy polecenie `run_qtparted`, aby uruchomić program `qtparted` umożliwiający zarządzanie partycjami na dysku. Po lewej stronie u góry jest napis urządzenia i są wypisane dyski podłączone do naszego komputera. Klikamy w ten, z którego chcemy "zabrać" trochę miejsca. Teraz po prawej stronie pokaże się partycja z obecnym systemem i różnymi informacjami takimi jak np. ilość wolnego miejsca. Klikamy prawym przyciskiem myszki i wybieramy `[Zmień rozmiar]`. Wpisujemy nowy rozmiar partycji i klikamy `[Ok]`. Gotowe, przygotowaliśmy sobie miejsce na dysku dla naszego Auroksa.

Teraz rozpoczniemy prawdziwą instalację. Restartujemy komputer i wkładamy krążek CD 1 instalacji Auroksa. Pojawi się znany nam już napis "boot:", wciskamy `ENTER` , aby uruchomić instalację z domyślnymi ustawieniami lub podajemy specjalne ustawienia jądra systemu.

Uwaga. Instalator Linux Aurox - Anaconda jest bardzo wygodny dla początkującego użytkownika, ponieważ jest w pełni spolszczony. Dodatkowo po lewej stronie znajduje się bardzo przejrzysta pomoc, która podpowie nam, co oznaczają poszczególne opcje i jakie są skutki ich wybrania.

Na pytanie o rodzaj instalacji wybieramy "Decyduj Sam", dzięki czemu będziemy mogli wszystko sami dokładnie określić nie pozostawiając nic automatom. Teraz należy dokonać podziału dysku na partycje. Wcześniej przygotowaliśmy sobie wolne miejsce nie tracąc danych na innym systemie i teraz nie kasujemy partycji, tylko dodajemy nowe (patrz instalacja serwera).

Jeśli chcemy, aby partycje innego systemu dostępne były z poziomu naszego Linuksa, to partycję taką zaznaczamy i wybieramy `[Edytuj]`. W tym miejscu określamy punkt montowania takiej partycji (np. `/mnt/dysk_c`) i sprawdzamy, czy przypadkiem nie jest zaznaczona opcja formatowania. Partycje FAT pozostawiamy bez zmian, **NIE FORMATUJEMY**.

Jeśli chodzi o bootloader, to polecam GRUB-a. Jest ładniejszy i wygodniejszy bez konieczności jakiejkolwiek dodatkowej konfiguracji. Instalujemy go oczywiście w MBR. Który system ma się pierwszy ładować jako domyślny, to już wybieramy dla wygody w zależności od tego, z którego systemu zamierzamy częściej korzystać. Pozycje na liście systemów nazywamy również według własnego upodobania (te dwie opcje jest bardzo łatwo zmienić już po zainstalowaniu systemu, więc nie przejmujmy się zbytnio tym wyborem).

Ustawienia sieci nie stwarzają większego problemu. Jeśli na serwerze ustawimy dynamiczne przydzielanie adresów IP (patrz DHCP - dynamiczna konfiguracja hostów) po prostu klikamy dalej. Jeśli jednak korzystamy z serwera jakiejś firmy wpisujemy konfiguracje podaną przez dostawcę internetu.

Na pytanie o firewalla odpowiemy "Bez firewalla", ponieważ lepiej ustawiać wszystko ręcznie, a jeśli ustawimy go dobrze na serwerze, to na stacji będzie zbędny. Następnie należy ustawić zegarek, co nie jest trudne, bo instalator praktycznie sam za nas już wszystko ustawił. Wystarczy potwierdzić.

Następnym krokiem jest ustawienie hasła root'a (jest to super-użytkownik, ale o nim później, na razie tylko tyle, że jego hasło musi być dość skomplikowane, ale bez przesady, żebyśmy mogli je bez trudu zapamiętać. Tylko nie zapisujcie go na karteczce !).

Teraz najciekawszy i najważniejszy krok instalacji — instalacja oprogramowania (pamiętajmy jednak, iż później możemy wszystko doinstalować lub odinstalować). Tu postąpimy inaczej niż w przypadku instalacji serwera - zainstalujemy system graficzny oraz multimedia... Ale po kolei. Do wyboru mamy kilka środowisk graficznych : KDE (najbardziej rozbudowany, można bardzo ładnie go skonfigurować, lecz jest najbardziej wymagający), GNOME (jest rozbudowany, nie aż tak wymagający, a przy odrobinie wysiłku można z niego zrobić bardzo ładny system) oraz fluxbox (najszybszy, nie mający ikonek, lecz mający swoich zwolenników nawet wśród użytkowników lepszych komputerów). Wybór ten zależy tylko i wyłącznie od naszych upodobań. Proponuję zainstalować 2 lub 3 i zdecydować, z którym najlepiej nam się pracuje lub wybrać ten, który będzie przedmiotem prowadzonych zajęć.

Aurox oferuje nam wiele programów. Nie będę opisywał każdego z osobna, tylko napiszę, jakie grupy pakietów należy przejrzeć. Ważniejsze grupy to :

- edytory — z linuksowych edytorów tekstu najpopularniejsze to vim oraz emacs,
- internet — tleenx, xchat (graficzny klient IRC'a), klient poczty: mozilla-mail,
- programy biurowe — najciekawszym programem w tej grupie jest Openoffice, alternatywa dla MS Office oraz przeglądarka XPDF pozwalająca przeglądać pliki z rozszerzeniem .pdf,
- dźwięk i wideo — odtwarzacze filmów : xine, mplayer, alternatywa winampa : xmms, kodeki oraz narzędzia do nagrywania płyt CD,
- grafika — obróbka grafiki : gimp, imagemagick,
- narzędzia dla programistów — narzędzia dla programistów c++, c, python, perl i inne : gcc, automake, biblioteki i pliki nagłówkowe,
- narzędzia administratora — graficzne narzędzia do zarządzania systemem : redhat-config-date, users itd.,

- narzędzia systemowe — narzędzia pomagające pracować na Linuksie : nmap (skaner portów), samba-client(klient do wymiany plików między Linuksem a Windowsem)
- popularne narzędzia — grupa popularnych narzędzi : mc (program pomagający pracować w konsoli), wine (emulator środowiska Windows dla programów Windowsowych).

We wszystkich wersjach Auroksa poza 9.3 i 9.4 mamy możliwość wybrania poszczególnych pakietów (wcześniej opisywałem wybór grup pakietów). Myślę, iż na początek wystarczy. Radzę nie przejmować się aż tak wyborem pakietów, ponieważ w każdej chwili można doinstalować kolejne. Po wybraniu wszystkich potrzebnych pakietów klikamy "Dalej" i tylko zmieniamy płyty CD. Po zakończeniu instalacji instalator będzie chciał zrobić dyskietkę startową Auroksa, która może się nam bardzo przydać w przyszłości. Ze względów bezpieczeństwa doradzam wykonanie takiej dyskietki. Wprawdzie dyskietkę taką można wykonać z poziomu pracującego systemu, ale skoro instalator proponuje nam pomoc ... Restartujemy komputer i już mamy nowy system operacyjny zainstalowany na stacji roboczej.

2.2.1 Instalacja przez sieć

Część IV

Wiersz poleceń

Rozdział 3

Podstawy systemu Linux

3.1 Początki wielkiej przyjaźni

Po włączeniu komputer trochę pomruczy, pokręci dyskami, wyświetli sporo niezrozumiałych komunikatów i wreszcie obudzi się wyświetlając komunikat podobny do poniższego :

```
Aurox Linux release 9.3 (Wind)
Kernel 2.4.20-20.9 on an i686

dreptak login:
Password:
```

Dwie pierwsze linie nie mają w tej chwili dla nas większego znaczenia, zresztą mogą wyglądać zupełnie inaczej (root decyduje o ich wyglądzie). Kolejna linia to prośba o przedstawienie się. Komputer o nazwie *dreptak* chce się dowiedzieć, kto go zbudził z błogiego snu. Wpisz swój login (mój to tuptus, Twój poda Ci administrator systemu, na którym chcesz czarować) i zatwierdź klawiszem ENTER . W odpowiedzi pojawi się kolejna linia. To pytanie o hasło. Twój login może podać każdy i dlatego komputer chce się upewnić, że Ty to rzeczywiście Ty. Pamiętaj, że czarnoksiężnicy byli, są i będą. Zatem strzeż swego hasła jak oka w głowie. Jeśli ktoś niepowołany je pozna może narobić Ci kłopotów, jeśli zapomnisz hasło, to komputer nie będzie chciał z Tobą rozmawiać. Teraz wpisz swoje hasło i zatwierdź klawiszem ENTER .

Uwaga. Dla bezpieczeństwa wpisywane hasło nie pojawi się na ekranie. Uważaj czy nikt Ci nie patrzy na ręce. Ty też tego nigdy nie rób, to objaw **złego wychowania**.

Jeśli poprawnie wprowadziłeś powyższe dane, to wejdziesz do systemu i otrzymasz prompt (znak zachęty). Standardowo, w systemach UNIX pojawia się niewiele mówiący `$` lub `#`, ale w naszym systemie jest trochę lepiej :

```
Last login: Tue Jan 7 20:00:07 on tty1
[tuptus@dreptak tuptus]$
```

Otrzymałeś informację o ostatnim logowaniu (czy to Ty się logowałeś, czy ktoś włamał się na Twoje konto?) oraz zachętę do wpisania zaklęcia. W prompcie podano Twój login, nazwę komputera oraz katalog, w którym aktualnie się znajdujesz. Oczywiście dane podane wyżej dotyczą mojego konta, a Twoje będą wyglądać nieco inaczej. Wygląd promptu można zmienić, ale do tego trzeba troszkę więcej wiedzy, więc zostawmy ten temat do omówienia na przyszłość.

Więc jesteś w systemie, komputer czeka na zaklęcia, zacznijmy czary. Sprawdźmy, czy komputer dobrze Cię rozpoznał :

```
[tuptus@dreptak tuptus]$who am i
tuptus   tty1         Jan  7 22:06
[tuptus@dreptak tuptus]$
```

Wygląda na to, że wszystko jest OK, więc sprawdźmy, co komputer wie o Tobie :

```
[tuptus@dreptak tuptus]$finger -l
Login: tuptus                           Name: (null)
Directory: /home/tuptus                 Shell: /bin/bash
On since Tue Jan  7 22:04 (CET) on tty1
No mail.
No Plan.
[tuptus@dreptak tuptus]$
```

Całkiem sporo jak na początek. `Login` - to już wiesz, `Name` - to pełna nazwa użytkownika, z reguły imię i nazwisko, `(null)` oznacza, że tych danych nie wprowadzono, `Directory` - to Twój katalog domowy, Twoje prywatne dane, do tego jeszcze wrócimy, `Shell` - to język w jakim możemy przekazywać zaklęcia komputerowi, o tych językach za moment, dalej jest informacja od kiedy jesteś zalogowany do systemu, czy masz w skrzynce nieodebraną pocztę e-mail. Polecenie `finger` ma kilka opcji. Aby dowiedzieć się więcej o tym zaklęciu wpisz `man finger`. Jeśli masz szczęście, to system odpowie Ci po polsku.

3.2 Hasło - klucz do świata magii

Aby wejść do systemu operacyjnego (ale nie tylko) musisz podać login (przedstawić się) oraz hasło, które pozwala systemowi stwierdzić, że Ty to naprawdę Ty. Po zidentyfikowaniu, system przydziela użytkownikowi odpowiednie uprawnienia, ustala jego środowisko oraz przydziela odpowiednie zasoby. Dlatego identyfikacja jest tak istotna. Jeżeli ktoś inny pozna Twoje hasło, to będzie mógł podszyć się pod Ciebie i w Twoim imieniu rzucać czary. Kiedy po raz pierwszy logowałeś się do systemu wpisałeś hasło podane Ci przez administratora. Pierwszą rzeczą, którą powinieneś wtedy zrobić jest zmiana hasła. Zatem jak tego dokonać? Wpisz zaklęcie `passwd`. System kolejno pyta o stare hasło (aby upewnić się, ze ktoś nie robi Ci głupiego dowcipu), nowe hasło oraz prosi o potwierdzenie go. Ponieważ w trakcie wprowadzania hasła nie są wyświetlane żadne znaki, to o pomyłkę bardzo łatwo i stąd żądanie potwierdzenia hasła.

Wybór właściwego hasła nie jest rzeczą łatwą. Hasło musi być łatwe do zapamiętania, ale jednocześnie trudne do odgadnięcia. W niektórych szczególnie strzeżonych systemach administratorzy stawiają wymóg okresowej zmiany haseł, co jeszcze bardziej komplikuje wybór właściwego, gdyż hasła nie mogą się powtarzać (w każdym razie nie nazbyt często). Podam zatem kilka rad dotyczących doboru hasła :

1. hasło powinno mieć co najmniej sześć znaków (wielu administratorów ustawia takie ograniczenie w konfiguracji systemu i system nie przyjmie krótszego)

2. hasło nie powinno zawierać Twojego loginu (system zwróci na to uwagę i odrzuci taką propozycję)

3. hasło nie powinno zawierać kolejnych liter alfabetu czy też kolejnych cyfr (wielu mugoli ustawia hasło `12345678`, to najprostsza droga do problemów)

4. hasło nie powinno zawierać słów ze słownika (jedną z metod łamania haseł jest wyszukiwanie słów ze słownika i porównanie wyniku szyfrowania z oryginałem)

5. hasło nie powinno zawierać słów łatwo powiązanych z Twoją osobą (imię kota, psa, marka wymarzonego samochodu itp.)

6. hasło powinno zawierać duże i małe litery, cyfry i inne znaki np. `T_b;T#03` (to nie jest moje hasło, zbyt proste)

Warto zwrócić uwagę na to, że hasło może być bardzo długie, ale w wielu, zwłaszcza starszych systemach rozpoznawane jest tylko pierwsze osiem znaków. Linux to ten nowszy system i interpretuje 255 znaków.

Skoro hasło jest tak ważne, to gdzie system przechowuje te hasła i w jaki sposób? Oczywiście przechowuje je w pliku. We wszystkich systemach UNIX-owych w katalogu `/etc` istnieje plik o nazwie `passwd`. Plik ten jest dostępny do odczytu dla wszystkich, więc możesz go obejrzeć (np. zaklęciem `less`). Skoro każdy może obejrzeć, to co to za tajemnica? Przyjrzyj się jeszcze raz zawartości tego pliku. Czy znalazłeś tam swoje hasło? Oczywiście, że nie. Linux stara się być systemem bezpiecznym i ukrywa hasło. Teraz trochę bardziej szczegółowo. Pierwsze pole (pola rozdzielone są dwukropkiem) to login użytkownika, drugie to jego hasło, trzecie - numer id (komputerowi łatwiej posługiwać się liczbami niż ciągami znaków), czwarte - numer gid, czyli id grupy użytkowników, do której należy dany użytkownik, piąte - opisowa nazwa użytkownika np. imię i nazwisko, szóste - katalog domowy użytkownika i ostatnie to domyślny shell użytkownika. Jak widzisz w polu hasła wszyscy użytkownicy mają x. Dla systemu to sygnał, że hasła ma szukać w innym miejscu (w starych systemach rzeczywiście było tu hasło). Tym innym miejscem w Linuksie jest plik shadow (cień). Nie szukaj go. Jeśli nawet go znajdziesz, to zdziwiłbym się mocno, gdybyś mógł go obejrzeć. W pliku tym zapisane jest hasło (niezupełnie, ale o tym za moment), data ważności hasła i jeszcze kilka pożytecznych informacji, ale raczej dla administratora. Co zatem z hasłem? Otóż w pliku shadow nie jest przechowywane samo hasło, lecz tak zwany odcisk hasła. Kiedy ustawiasz swoje hasło system wykonuje algorytm MD5 i wynik jego działania zapisuje do pliku shadow. Nie będę tłumaczył na czym polega ten algorytm, bo to już bardzo zaawansowana magia. Dla naszych potrzeb wystarczy informacja, że MD5 jest algorytmem jednokierunkowym, czyli mając odcisk hasła nie można na jego podstawie odtworzyć samego hasła. Nawet root nie jest w stanie stwierdzić jakie hasło ustawiłeś. W takim razie jak system może stwierdzić, że przy logowaniu podałeś właściwe hasło? Prosto. Bierze podane przez Ciebie hasło, przepuszcza przez algorytm MD5 i sprawdza czy wynik jest identyczny z tym w pliku shadow.

Teraz jeszcze jedna informacja o hasłach. Wprawdzie na obecnym etapie kursu nie będzie to zbyt przydatna wiedza, ale żeby już był komplet. Hasło użytkownika może być :

- nieustawione – na takie konto nie można się zalogować, konta tego typu wykorzystywane są przez demony (trochę o ich oswajaniu powiemy w dalszej części kursu, w systemie taką hodowlę prowadzi root)
- puste – na pytanie o nowe hasło wciśnięto `ENTER` bez wpisywania jakichkolwiek znaków (taką sztuczkę może wykonać tylko root), logowanie wymaga jedynie wciśnięcia `ENTER`
- ustawione – czyli normalne, ustawione przez użytkownika zgodnie ze wszystkimi omawianymi regułami

3.3 Tajemniczy root

W czasie tego krótkiego kursu już kilkakrotnie pojawiał się użytkownik root. Kto to taki? Root to taki mistrz ceremonii w Twoim systemie. Rootowi wolno prawie wszystko. To root decyduje

o starcie i zamknięciu systemu, on zakłada konta użytkownikom, przydziela im zasoby i uprawnienia. Root hoduje demony, root otwiera i zamyka wrota do świata zewnętrznego lub rozdaje klucze do nich, root dba o porządek w systemie, root decyduje o możliwości użycia określonych zaklęć i dodaje nowe do zasobów systemu. Jak widzisz root to ważna figura. Dlatego jednym z głównych celów adeptów ciemnej strony mocy jest przejęcie roli roota w Twoim systemie. Root nie jest konkretnym użytkownikiem. Użytkownik o dużym doświadczeniu lub właściciel danego systemu w pewnych okolicznościach staje się rootem, ale poza tym jest zwykłym użytkownikiem, takim jak Ty. Użytkownik, który może przemieniać się w roota nazywany jest administratorem systemu lub adminem. I to tyle tajemnicy. Pamiętaj, jeśli mówimy o jakimś zaklęciu, że może je wykonać tylko root, to najprawdopodobniej jest to bardzo mocne zaklęcie i bardzo często nieodwracalne, a już na pewno jego efekty odczują wszyscy pozostali użytkownicy systemu.

Jak stać się rootem? Pierwszą sprawą jest znajomość hasła dla konta root. Jeśli jesteś jedynym użytkownikiem hosta lub właśnie instalujesz linuksa na nowej maszynie, to zapewne znasz to hasło. Znając hasło możesz zalogować się na konto użytkownika root tak samo jak na każde inne (nie zawsze tak jest, ale o tym w dalszej części). Odradzam jednak takie postępowanie, gdyż wyrabia zły nawyk pracy na tym koncie.
Lepszym rozwiązaniem jest logowanie na swoje konto, a następnie, na czas wykonania określonej czynności, zmieniamy tożsamość. Możemy dokonać tego wykorzystując zaklęcie `su`. Wydanie polecenia `su` bez żadnych parametrów zmienia naszą tożsamość na roota i nic więcej. Dlatego zalecałbym stosowanie zaklęcia w postaci `su -`. Dzięki temu minusowi na końcu zmieniamy nie tylko tożsamość, ale przejmujemy wszystkie cechy roota, całą jego wiedzę o systemie. Bardziej profesjonalnie należałoby powiedzieć, że następuje uruchomienie powłoki użytkownika root w trybie logowania. O powłoce i jej cechach będziemy jeszcze mówić szerzej. W tym miejscu jedynie krótka informacja, że jednym z elementów wiedzy o systemie jest wiedza o umiejscowieniu w nim różnych zaklęć i umiejętność ich wyszukiwania (root wie na ten temat więcej niż inni użytkownicy).

Rozdział 4

Shell - język zaklęć

Shell to program UNIX-a, który oczekuje na wpisanie zaklęcia i stara się je wykonać. Istnieją różne shelle tworzone przez ambitnych czarodziejów o wysokich kwalifikacjach. Główne to Bourne shell, Korn shell i C shell. W systemie nie znajdziesz programów o takich nazwach. Znaleźć możesz natomiast bash (Bourne Again shell), csh, tcsh (to z rodziny C shelli), ksh (Korn shell) i jeszcze kilka innych. My będziemy pracować w bash-u.

4.1 Podstawowe zaklęcia

Poznałeś już kilka zaklęć. Nie były to zaklęcia specjalnie przydatne w życiu codziennym. Teraz wypełnimy tę lukę. Zacznijmy od rozglądnięcia się po Twoim "domku". Tak, zaraz po zalogowaniu system przenosi Cię do Twojego katalogu domowego. Pamiętasz pole `Directory`? Sprawdźmy to :

```
[tuptus@dreptak tuptus]$ pwd
/home/tuptus
[tuptus@dreptak tuptus]$
```

Zaklęcie (przez system nazywane command - komenda) `pwd` to tyle, co pytanie : "Gdzie ja jestem?" No cóż, jesteś w swoim katalogu domowym. Zatem sprawdźmy co w tym "domku" się znajduje. Wykorzystamy do tego celu komendę `ls` (od angielskiego list) :

```
[tuptus@dreptak tuptus]$ ls
Desktop
[tuptus@dreptak tuptus]$
```

Raczej nie wiele. Jeśli nie masz zainstalowanego środowiska graficznego, to nawet tego nie zobaczysz. Spróbujmy więc trochę inaczej :

```
[tuptus@dreptak tuptus]$ ls -la
drwx------   11 tuptus   tuptus       4096 sty  7 22:43 .
drwxr-xr-x    5 root     root         4096 sty  4 13:02 ..
-rw-------    1 tuptus   tuptus       4359 sty  7 22:00 .bash_history
-rw-r--r--    1 tuptus   tuptus         24 sty  4 00:06 .bash_logout
-rw-r--r--    1 tuptus   tuptus        191 sty  4 00:06 .bash_profile
-rw-r--r--    1 tuptus   tuptus        124 sty  4 00:06 .bashrc
drwx------    3 tuptus   tuptus       4096 sty  7 21:16 Desktop
[tuptus@dreptak tuptus]$
```

Teraz mamy trochę więcej informacji. Czas wyjaśnić, co to za tajemnicze zaklęcia. Komenda `ls` w języku shell-a oznacza tyle, co "podaj mi listę plików". Opcja `-l` żąda podania informacji w

długim formacie (od angielskiego long), opcja `-a` żąda podania informacji o wszystkich plikach (od angielskiego all), również tych ukrytych. Czego właściwie się dowiedzieliśmy po wydaniu tego polecenia?

1. Tajemnicze znaczki w pierwszej kolumnie informują o rodzaju pliku oraz uprawnieniach do niego. Pierwszy znaczek to rodzaj pliku :
   - \- – zwykły plik
   - d – katalog
   - l – link; to taki plik, który w rzeczywistości znajduje się w innym miejscu

   Kolejne znaczki (po trzy) informują o prawach do pliku dla właściciela, grupy i innych :
   - \- – brak prawa
   - r – prawo czytania
   - w – prawo pisania
   - x – prawo wykonania

   Więcej o prawach dowiesz się z dalszej części kursu.
2. Liczby w drugiej kolumnie na tym etapie nauki nas nie interesują, mówią one o ilości linków do danego pliku
3. Trzecia i czwarta kolumna to odpowiednio właściciel pliku i grupa; zauważ, że dla pliku `..` właścicielem jest root
4. kolejna kolumna to rozmiar pliku w bajtach
5. kolejne trzy kolumny to data i godzina utworzenia/modyfikacji pliku
6. ostatnia kolumna to nazwa pliku; jeśli nazwa zaczyna się od kropki, to jest to plik ukryty

Zaklęcie `ls` ma jeszcze wiele innych wariantów, których tu nie omówiliśmy.

Uwaga. Za każdym razem kiedy omawiamy jakąś nową komendę wskazane jest abyś zapoznał się z jej opisem w manualu systemowym. Wywołuje się go poleceniem `man` podając jako parametr nazwę zaklęcia, o którym chcemy poczytać. Istnieje również drugi, nowszy podręcznik, wywoływany poleceniem `info`. Tutaj również jako parametr wywołania należy podać nazwę polecenia, o którym chcemy zasięgnąć informacji.

A teraz kolejne doświadczenie. Wpisz poniższą komendę i zobacz, co się stanie :

```
[tuptus@dreptak tuptus]$ ls -l /usr/bin
```

Wyniku nie przedstawiam bo ... Zapewne zauważyłeś, że przez ekran przeleciało bardzo dużo danych i nie zdążyłeś ich przeczytać, uciekły bezpowrotnie. Nie przejmuj się, jest na to sposób, a nawet kilka sposobów. Moje ulubione zaklęcie w takich sytuacjach to `less`. Wpisz ponownie ... Nie, może nie wpisuj. Jest pewna sztuczka, która ułatwi Ci życie. Wciskając klawisze strzałek (w górę i w dół) możesz ponownie przywołać wcześniej wykonane zaklęcia. Zapamiętaj ten trik i wykorzystuj go. W dalszej części poznasz jeszcze kilka takich trików. Zatem wróć do poprzedniego zaklęcia i dopisz `| less` :

```
[tuptus@dreptak tuptus]$ ls -l /usr/bin | less
```

Teraz widzisz początek listy plików, a u dołu ekranu dwukropek. Wykorzystując klawisze nawigacyjne (strzałki, Page Down i Page Up) możesz poruszać się po tej liście. Jeśli chcesz znaleźć jakieś słowo, które powinno być na wyświetlanej liście wpisz /słowo np. /cpp ENTER Lista przesunęła się tak, aby linia z poszukiwanym słowem była na górze ekranu, a znalezione słowo zostaje podświetlone. Jeśli chcesz znaleźć kolejne wystąpienie tego słowa wpisz / ENTER i lista zostanie przesunięta do kolejnej pozycji. Przeszukiwanie odbywa się od początku do końca wyświetlonej listy. Jeśli chciałbyś wykonać przeszukiwanie w odwrotnym kierunku, to zamiast / użyj ?. Aby zakończyć wyświetlanie listy wciśnij q. Zaklęcie, które przed chwilą poznałeś, to proces potokowania poleceń. Więcej o potokach i strumieniach będziemy jeszcze mówić w dalszej części.

4.2 Pliki i katalogi

W naszych dotychczasowych zaklęciach pojawiły się zapisy, których do tej pory nie wyjaśniłem. Co oznacza /home/tuptus? Tak, to jest katalog domowy, ale co to takiego ten katalog? Zacznijmy od początku. Każda informacja w systemie musi być w jakiś sposób reprezentowana. Tym sposobem są pliki. Plik to obszar na dysku, gdzie zapisane są jakieś informacje. Nie do końca to prawda, ale w tej chwili musi Ci taka informacja wystarczyć. Plików w systemie jest bardzo dużo (u Ciebie prawdopodobnie ponad 100 tyś.). Trzeba je jakoś uporządkować. Przecież taką ilością nikt nie umiałby zarządzać. Dlatego wymyślono katalogi. Katalog to wydzielony obszar zawierający pliki i inne katalogi (podkatalogi). Katalogi i podkatalogi tworzą strukturę podobną do drzewa. Na początku jest root czyli korzeń i zapisujemy go znakiem /. Root zawiera pliki i katalogi np. /home, /usr. Te katalogi zawierają pliki i kolejne katalogi nazywane podkatalogami np. Twój katalog domowy /home/tuptus itd. itd. Dobrze. Teraz już wiesz jak są poukładane dane i zaklęcia (tak tak, zaklęcia też są plikami), z poprzedniej części wiesz jak sprawdzić zawartość katalogu, jeśli wiesz jak on się nazywa. Ale jak zarządzać tymi katalogami? Już wyjaśniam.

4.2.1 Wędrówki po systemie

Zaczniemy od spaceru po systemie. Umiejętność poruszania się po systemie będzie Ci niezbędna do dalszego czarowania, więc zapoznaj się z tą częścią bardzo dokładnie.

Zaklęcie, które umożliwia poruszanie się po drzewie katalogów to cd (od angielskiego change directory).

```
[tuptus@dreptak tuptus]$ cd /usr/bin
[tuptus@dreptak bin]$ ls -l | less
```

Pierwsze polecenie przenosi Cię do katalogu /usr/bin. Drugie polecenie już znasz. Jest jednak pewna różnica. Zauważ, że zmienił się prompt, pokazuje bin jako katalog bieżący (czyli ten, w którym się aktualnie znajdujesz). I tu ujawniła się pewna tajemnica zaklęcia ls. Otóż jeśli nie podasz katalogu, który chcesz wylistować, to zaklęcie pokaże Ci zawartość katalogu bieżącego, czyli tego, w którym się aktualnie znajdujesz. Proponuję abyś pospacerował po systemie plików. Nie zdziw się, jeśli do jakiegoś katalogu nie zostaniesz wpuszczony. O prawach w systemie decyduje root (taki czarodziej, a nie początek drzewa katalogów) i możliwe, że nie życzy sobie Twojej wizyty w tym miejscu. Teraz kilka podpowiedzi :

- jeśli jesteś w katalogu np. /home i chcesz wejść do podkatalogu to wystarczy, że wpiszesz cd tuptus, nie musisz pisać całej ścieżki zaczynając od root-a

- jeśli wpiszesz `cd` bez parametrów to zostaniesz natychmiast przeniesiony do katalogu domowego, niezależnie od tego gdzie się obecnie znajdujesz
- jeśli wpiszesz `cd -` zostaniesz przeniesiony z powrotem do katalogu, z którego przyszedłeś

4.2.2 Tworzenie i niszczenie

Skoro umiesz już poruszać się po katalogach, to warto nauczyć się jak katalogi zakładać i kasować. Przejdź do swojego katalogu domowego i zaczynamy czary. Zaklęcie zakładające katalog to `mkdir`. Załóżmy kilka katalogów.

```
[tuptus@dreptak tuptus]$ mkdir test
[tuptus@dreptak tuptus]$ cd test
[tuptus@dreptak test]$ mkdir test_s1
[tuptus@dreptak test]$ mkdir test_s2
[tuptus@dreptak test]$
```

OK. Teraz sprawdź gdzie jesteś i co znajduje się w bieżącym katalogu.

Nazwy katalogów, które utworzyłeś to `test`, `test_s1` i `test_s2`. Może chciałbyś założyć jeszcze jakieś katalogi. Bardzo dobrze, ale zaczekaj chwilę. Czy już wiesz jak nazywać katalogi? Masz dużą swobodę w nadawaniu nazw, ale jest też kilka ograniczeń :

1. litery duże i małe to różne znaki zatem `test` i `Test` to różne katalogi
2. w nazwach katalogów nie używaj polskich znaków ąćęłńóśźż - wprawdzie nie są to znaki zakazane, ale w pewnych sytuacjach mogą stwarzać problemy
3. w nazwach nie używaj spacji (odstępu), `mkdir nowy katalog` nie założy katalogu `nowy katalog`, lecz dwa oddzielne katalogi (można to obejść, ale lepiej nie ucz się tego, to prosta droga do kłopotów), zamiast spacji użyj podkreślenia
4. nie używaj znaków zastrzeżonych `/\&$#@*{}[]'"‘~` (to chyba wszystkie)

Teraz możesz poeksperymentować, ale radzę nie wychodzić poza katalog domowy.

A jak usunąć niepotrzebne katalogi? Można to zrobić na dwa sposoby :

1. `rmdir` – to zaklęcie usuwa katalog, ale pod warunkiem, że jest on pusty, np. jeśli jesteś w katalogu domowym i spróbujesz usunąć tym sposobem katalog test to shell zbuntuje się podając komunikat
   `rmdir: ‘test’: Katalog nie jest pusty`, musisz najpierw wejść do tego katalogu i go wyczyścić, jeśli w podkatalogach też coś jest, to czynność musisz powtórzyć również dla podkatalogów
2. `rm -R` – to zaklęcie służy w zasadzie do kasowania plików, ale jak wcześniej mówiłem w systemie wszystko jest plikiem, opcja `-R` nakazuje wykonać polecenie rekursyjnie (po naszemu, przeleć również podkatalogi), więc ostatecznie polecenie powinno wyglądać tak `rm -R test`

Uwaga. Zaklęcie `rm` **jest nieodwracalne**, używaj z rozwagą. Wprawdzie istnieją sposoby na odzyskanie skasowanych danych, ale to już wyższa szkoła jazdy, tylko dla najlepszych.

Teraz jeszcze kilka słów o specjalnych katalogach. Zapewne zauważyłeś, że w wyniku wykonania każdego polecenia `ls -a` na początku listy plików pojawiają się dwa tajemnicze katalogi, . (jedna kropka) i .. (dwie kropki). To są te specjalne katalogi. Jedna kropka oznacza katalog bieżący. Zatem jeśli podasz zaklęcie `less ./plik`, to shell zrozumie, że chcesz obejrzeć zawartość pliku `plik` znajdującego się w bieżącym katalogu. Dwie kropki oznaczają katalog nadrzędny. Zatem jeśli podasz zaklęcie `less ../plik`, to shell zrozumie, że chcesz obejrzeć zawartość pliku `plik` znajdującego się w katalogu nadrzędnym np. jeśli jesteś w `/home/tuptus/test`, to shell będzie próbował wyświetlić plik `/home/tuptus/plik`. Jest jeszcze jeden znaczek specjalny, który będzie Ci przydatny. Ten znaczek to `~` (tylda). Znaczek ten rozumiany jest przez shell jako skrót ścieżki do Twojego katalogu domowego. Zatem jeśli podasz zaklęcie `less ~/plik`, to shell zrozumie, że chcesz obejrzeć zawartość pliku `plik` znajdującego się w Twoim katalogu domowym i będzie próbował wyświetlić plik `/home/tuptus/plik`. Możesz też wpisać `less ~user/plik`, co oznacza, że chcesz obejrzeć plik `plik` z katalogu domowego użytkownika `user` (ale czy masz takie uprawnienia, to inna sprawa). Zauważ, że to shell wstawia odpowiednią ścieżkę na podstawie informacji o danym użytkowniku takich, jakie posiada. Nie jest istotne czy katalog domowy to `/home/user`, czy jakiś inny (przypomnij sobie polecenie `finger`).

Skoro powiedziałem już jak kasować pliki, to powinienem jeszcze powiedzieć jak takie pliki zakładać. Na początek polecam zaklęcie `touch`. Słowo to oznacza dotyk lub dotknięcie. Co się dzieje kiedy wydaję polecenie `touch plik`? Jeśli plik `plik` istnieje, to zostanie zmieniona data jego ostatniej modyfikacji, jeśli nie istnieje, to zostanie utworzony pusty plik (zawiera 0 bajtów). Zasady nazywania plików są takie same jak dla katalogów, przecież katalogi to też pliki tylko trochę inne.

Proponuję abyś teraz trochę poeksperymentował z plikami i katalogami. Na koniec posprzątaj swój katalog domowy z tych eksperymentów.

4.3 Co wolno wojewodzie...

Kiedy listowałeś system plików widziałeś po lewej stronie literki rwx. Powiedziałem wówczas, że to opis uprawnień do plików i katalogów. Teraz zajmiemy się dokładniej tym tematem. Pierwszy znaczek informuje o typie pliku. Jeśli jest kreska, to jest to zwykły plik, literka `d` oznacza katalog, literka `l` oznacza link (zawartość pliku wskazuje na położenia prawdziwego pliku). W systemie mogą wystąpić jeszcze inne typy plików, ale to raczej informacje potrzebne dla roota, więc nie będziemy sobie nimi zawracali głowy.

Kolejne dziewięć znaczków opisuje uprawnienia do pliku/katalogu. Tak na prawdę to są to trzy grupy po trzy bity (bit to 0 lub 1 i nic więcej; dla komputera - jest sygnał lub go nie ma). Pierwsza grupa dotyczy uprawnień właściciela pliku, druga - grupy użytkowników, a trzecia - pozostałych użytkowników. Tabela 4.1 przedstawia opis tych uprawnień. Aby łatwiej zrozumieć

Uprawnienie	**Dla pliku**	**Dla katalogu**
r	prawo do czytania pliku	prawo do czytania zawartości katalogu (listowania)
w	prawo do pisania do pliku już istniejącego	prawo do tworzenia plików w katalogu oraz ich kasowania
x	prawo wykonania pliku - ustawiane dla programów i skryptów	prawo do wejścia do katalogu, ale nie do czytania jego zawartości

Tabela 4.1: Uprawnienia do plików/katalogów

znaczenie uprawnień do katalogów traktuj katalog jako plik zawierający informacje o zwykłych

plikach znajdujących się w katalogu : położenie w systemie plików, nazwa, właściciel, uprawnienia itd. Zwróć uwagę na zależności między uprawnieniami. Jeśli chcesz umożliwić wykonanie pliku (programu lub skryptu), to uprawnienie `x` nie wystarczy, gdyż przed wykonaniem plik musi być wcześniej odczytany, więc potrzebne jest uprawnienie `r`. Jeśli chcesz umożliwić odczytanie pliku, ale nie jego skasowanie, to do pliku nadaj uprawnienia `r`, a dla katalogu zabierz uprawnienia `w`. Takie ustawienie nie zabezpieczy przed zmianą zawartości pliku jeśli będzie ustawione uprawnienie `w` dla pliku.

A teraz praktyczne wykorzystanie powyższej wiedzy. Wszystkie doświadczenia wykonuj wewnątrz katalogu domowego. Do zmiany uprawnień służy polecenie `chmod`. Parametrami tego zaklęcia są :

1. komu i jakie uprawnienie zmieniamy
2. jakiego pliku dotyczy zmiana
3. opcje modyfikujące sposób wykonania polecenia

Pierwszy element to litery u-użytkownik, g-grupa, o-inni; + lub - określające nadanie lub zabranie uprawnień; rwx - opisane wyżej uprawnienia. Oznaczenia te możemy łączyć. Jako ćwiczenie wykonaj następujące zadanie : w katalogu `~/test` załóż plik `plik.test` i nadaj pełne uprawnienia dla właściciela, prawo do wykonania dla grupy i zabierz wszystkie uprawnienia pozostałym użytkownikom systemu. Dla przypomnienia :

1. mkdir – zakładanie katalogów
2. cd – przechodzenie miedzy katalogami
3. touch – tworzenie pustego pliku
4. ls – listowanie katalogu (opcja -l wyświetla długą informację)

Ostatecznie powinieneś otrzymać :

```
[tuptus@dreptak test]$ ls -l
-rwxr-x---    1 tuptus   tuptus          0 sty 15 12:32 plik.test
```

Aby wykonać powyższe zadanie musisz wydać następującą komendę :

```
[tuptus@dreptak test]$ chmod u+rwx,g+rx,g-w,o-rwx plik.test
```

Oczywiście można to samo osiągnąć podając zaklęcie `chmod` wielokrotnie, za każdym razem jako opcję podając kolejną część opcji rozdzielonych przecinkami w powyższym zaklęciu. Ale jest jeszcze "prostsza" metoda stosowana przez prawdziwych czarodziejów. Jak powiedziałem wcześniej uprawnienia to bity. Bit ustawiony (=1) uprawnienie jest, skasowany (=0) uprawnienia nie ma. Aby tę informację wykorzystać musisz przypomnieć sobie zajęcia matematyki i przeliczanie liczb pomiędzy układem dwójkowym i dziesiętnym. Tutaj podam bez tłumaczenia, że :

1. ustawiony x – wartość 1
2. ustawione w – wartość 2
3. ustawione r – wartość 4

Tak więc wcześniejsze zadanie można wykonać zaklęciem :

```
[tuptus@dreptak test]$ chmod 750 plik.test
[tuptus@dreptak test]$
```

czyli dla właściciela ustawione wszystkie bity(4+2+1=7), dla grupy ustawione bity r i x (4+1=5) a dla pozostałych wyłączamy wszystkie bity. Dużo prostsze, a już na pewno wymaga mniej pisania i jest szybsze. Przeanalizuj dokładnie tą składnię i zapamiętaj ją, gdyż są sytuacje, gdy tylko w ten sposób można zmienić uprawnienia na takie, jakie są pożądane.

Aby informacje były pełne, to podam jeszcze kilka informacji, które przydatne są głównie dla roota (w przyszłości Tobie) :

1. uprawnienia możesz zmieniać tylko dla plików, których jesteś właścicielem, ale nie dotyczy to roota; root może wszystko
2. tylko root może zmienić właściciela i grupę dla pliku (polecenia `chown` i `chgrp`)
3. istnieje jeszcze czwarta grupa bitów tzw. bity s - modyfikujące uprawnienia w momencie wykonania polecenia

I właśnie tą czwartą grupą bitów się teraz zajmiemy. Zestawienie bitów i ich znaczenie zostało przedstawione w tabeli 4.2. Podobnie jak dla uprawnień podstawowych ustawienia bitów

Pozycja	**Dla pliku**	**Dla katalogu**
użytkownik (suid) oznaczenie – s	podczas wykonania ustawia efektywny identyfikator użytkownika procesu na taki, jaki ma właściciel pliku wykonywanego	ignorowany
grupa (sgid) oznaczenie – s	podczas wykonania ustawia efektywny identyfikator grupy procesu na taki, jaki ma grupa pliku wykonywanego	ustawia dla plików w nim tworzonych grupę danego katalogu, bez względu na grupę użytkownika, który je tworzy
inni (sticky) oznaczenie – t	zachowuje kod programu na urządzeniu buforującym (swap) tak, aby przy następnym wywołaniu szybciej się uruchamiał	zabrania użytkownikom usuwania w nim innych plików poza własnymi

Tabela 4.2: Bity modyfikujące do plików/katalogów

modyfikujących można wykonywać poprzez symbole jak również liczbowo. Wartości przypisane odpowiednim bitom :

1. ustawiony bit sticky – wartość 1
2. ustawiony bit sgid – wartość 2
3. ustawiony bit suid – wartość 4

Zatem, jeśli mamy w systemie plik o nazwie `program` i chcemy, aby wszyscy mogli ten plik wykonywać tak, jak właściciel i dodatkowo, aby właściciel mógł ten plik modyfikować uprawnienia nadajemy komendą :

```
[root@dreptak test]# chmod 4755 program
```

Zwróć uwagę na pewien szczegół. Otóż bity modyfikujące ustawiać może jedynie root, mimo że właścicielem jest inny użytkownik i może on zmieniać prawa do pliku. Na zakończenie jeszcze dwie uwagi. Bitów modyfikujących nie można ustawiać dla skryptów, można jedynie dla plików binarnych. I druga uwaga — w przypadku braku ustawionych odpowiednich bitów `x` bity `s` są

ignorowane.

Uwaga. Bity modyfikujące stanowią niezwykle istotne narzędzie w ręku administratora i w wielu wypadkach ich użycie jest absolutnie niezbędne. Jednocześnie stanowią poważne zagrożenie bezpieczeństwa systemu i utrudniają śledzenie uprawnień. Używaj ich ze szczególną ostrożnością.

4.4 System plików, dyski

Wiemy już jak korzystać z plików i katalogów. Zajmijmy się teraz tym jak ten system jest zbudowany, jak dodać nowe dyski, jak korzystać z CD-ROM'u i dyskietek.

Jak zapewne już zauważyłeś system katalogów Linuksa stanowi jeden ciągły obszar. Powstaje zatem pytanie gdzie w tym systemie są dyski? Otóż Linux oddziela użytkownika od sprzętu. Każde urządzenie pamięci masowej (dysk, partycja itd.) musi zostać zamontowane zanim zostanie użyte. Montowanie polega na podłączeniu filesystemu znajdującego się na nośniku do istniejącego katalogu. Funkcję tą wykonuje polecenie `mount`.

Zanim jednak zajmiemy się tym poleceniem musimy poznać nazwy urządzeń. Zaczniemy od dysków twardych typu IDE. We współczesnych komputerach dostępne są dwa kontrolery IDE (Primary i Secondary) i do każdego z nich możemy podłączyć dwa urządzenia. Urządzenia te należy skonfigurować tak, aby jedno z nich było urządzeniem głównym (master), a drugie podrzędnym (slave). Urządzenia w systemie widoczne są jako :

- /dev/hda - Primary Master
- /dev/hdb - Primary Slave
- /dev/hdc - Secondary Master
- /dev/hdd - Secondary Slave

Urządzeń IDE może być więcej, w najnowszych maszynach spotyka się nawet cztery kontrolery IDE na płycie głównej.

Każdy z dysków musi posiadać przynajmniej jedną partycję, ale może ich mieć do 32. Partycje są numerowane od 1. I tu pojawia się kolejna uwaga. Na dyskach możemy mieć maksymalnie cztery partycje podstawowe. Jeśli chcemy mieć więcej partycji, to musimy jedną z partycji podstawowych zaznaczyć jako rozszerzoną i w niej tworzyć partycje logiczne. Partycje logiczne numerowane są od 5 niezależnie od liczby partycji podstawowych. Możemy zatem spotkać się z sytuacją gdy na dysku mamy partycje : 1 - podstawowa, 2 - rozszerzona, 5 i 6 - partycje logiczne. Jeśli sytuacja taka występuje na pierwszym dysku, to w systemie dostępne będą następujące urządzenia : /dev/hda1, /dev/hda5 i /dev/hda6.

Napędy CD-ROM to często także urządzenia IDE. Jeśli taki napęd podłączony jest jako Primary Slave, to w systemie widoczny będzie jako /dev/hdb (CD nie mają podziału na partycje, stanowią ciągły obszar danych nawet w przypadku krążków wielosesyjnych). Dla wygody zarządzania, dla napędów CD-ROM tworzone jest dowiązanie symboliczne o nazwie /dev/cdrom. W przypadku opisanym wyżej wykonujemy to poleceniem `ln -s /dev/hdb /dev/cdrom` (polecenie zarezerwowane dla roota). Należy jednak pamiętać, że /dev/cdrom to tylko dowiązanie, a nie faktyczne urządzenie.

Identycznie wygląda sprawa z urządzeniami SCSI z tą różnicą, że nazwy urządzeń zaczynają się od sd (/dev/sda, /dev/sdb itd.). Jeden kontroler SCSI pozwala na podłączenie do ośmiu

urządzeń. Występują jednak również kontrolery pozwalające na podłączenie do szesnastu urządzeń. Można również wykorzystać kilka odrębnych kontrolerów. Istnieją także napędy CD-ROM SCSI lub nagrywarki tego typu. W takim przypadku utworzenie dowiązania może mieć postać `ln -s /dev/sdb /dev/cdrom` (jeśli CD-ROM jest drugim urządzeniem SCSI). W systemach serwerowych występują także streamery — napędy taśm magnetycznych wykorzystywane do archiwizacji danych oraz wykonywania kopii bezpieczeństwa. Urządzenia te również podłączane są do kontrolerów SCSI.

Jak widać możemy mieć w systemie bardzo dużo różnych urządzeń i trzeba je w jakiś sposób zamontować do systemu plików. Na potrzeby naszych rozważań przyjmuję następujące założenia:

- Primary Master podzielony jest na partycje :
  - hda1 partycja "startowa" Linuksa
  - hda2 partycja z Sam-Wiesz-Czym sformatowana jako VFAT
  - hda3 partycja pamięci wirtualnej - swap
  - hda5 partycja główna Linuksa
- Primary Slave - napęd CD-ROM
- Secondary Master podzielony na partycje (dodany w wyniku rozbudowy sprzętu) :
  - hdc1 partycja danych użytkowników
  - hdc2 partycja danych systemu (baza danych, poczta, logi)

Aby system mógł wystartować należy zamontować co najmniej partycje hda5, hda1 i hda3. Czynność tą wykonuje za nas bootloader - proces odpowiedzialny za start systemu. Tą tematyką w tej chwili nie będziemy się zajmować.

Pozostałe partycje również mogą być zamontowane automatycznie przy starcie systemu, ale o tym za moment. Przyjmijmy w tej chwili, że mamy tylko jeden dysk i pracując w Linuksie chcemy uzyskać dostęp do danych na dysku hda2. Bierzemy się zatem do dzieła. Logujemy się jako root i przechodzimy do katalogu `/mnt`. Jest to katalog przewidziany do tworzenia podkatalogów nazywanych punktami montowania. Można oczywiście montowanie wykonać w dowolnym miejscu systemu plików, ale o tym za moment. Tworzymy zatem katalog win_c i montujemy do niego partycję hda2 :

```
[root@dreptak mnt]$ mkdir win_c
[root@dreptak mnt]$ mount /dev/hda2 /mnt/win_c -t vfat
```

(**TODO**...)

Poznałeś już podstawowe zaklęcia, wiesz jak poruszać się po drzewie katalogów. Czas zacząć przystosowywać ten świat do własnych potrzeb. Zanim jednak do tego przejdziemy musisz poznać kilka pożytecznych i niezbędnych narzędzi. Zaczniemy od edytora tekstu.

4.5 Edytor vim

Chociaż w każdym systemie UNIX dostępnych jest wiele edytorów tekstu, wszystkie one zawierają z pewnością edytor `vi`. W naszym systemie występuje `vi` - to taki standard, `vim` - rozszerzona wersja vi oraz `gvim` - vim działający w środowisku okienkowym. Tego ostatniego może nie być jeśli nie masz zainstalowanego środowiska X-Window. Dlaczego właśnie ten edytor? Po pierwsze występuje we wszystkich systemach UNIX, po drugie pozwala na edycję tekstów i

w pliku wynikowym nie zostawia żadnych znaków kontrolnych (pracuje na "czystym" tekście), a po trzecie wiele zaklęć jest ściśle powiązanych z tym edytorem i bez jego znajomości nie mógłbyś ich wykorzystać.

Edytor pracuje w dwóch trybach: tryb poleceń oraz tryb wprowadzania i edycji tekstu. W pierwszym przypadku niektóre klawisze lub ich zestawienia pozwalają na poruszanie się po tekście, usuwanie znaków i linii tekstu, zmiany tekstu oraz wiele innych. W drugim przypadku klawisze reprezentują poszczególne znaki. W naszym edytorze są również klawisze strzałek, oraz klawisze specjalne : Insert , Backspace , Delete , Tab i inne. Pamiętaj, że działanie tych klawiszy zależne jest od rodzaju terminala na jakim pracujesz oraz od wersji programu vi (w starszych wersjach klawisze te mogą nie działać). Aby rozpocząć pracę należy wpisać :

```
[tuptus@dreptak test]$ vim plik.test
```

Oczywiście `plik.test` to przykład. Możesz wpisać inna nazwę pliku, ale ten plik już mamy w systemie, więc warto go wykorzystać. Otworzyło się okno edytora i wygląda trochę tajemniczo, więc śpieszę wyjaśnić co jest co. Ponieważ nasz plik był pusty w chwili otwarcia edytora, to na ekranie, na początku każdej linii pojawiła się tylda (`~`). Oznacza ona, że w tej linii nie ma żadnych znaków. W ostatniej linii mamy informację o nazwie otwartego pliku, ilości linii i znaków w pliku, bieżące położenie kursora (linia, znak) oraz słowny opis położenia (w tym wypadku `Wszystko` gdyż cały plik został wyświetlony na ekranie). W momencie otwarcia pliku edytor przechodzi do trybu poleceń. Aby przejść do trybu wprowadzania wciskamy a lub i . Ponieważ nasz plik jest pusty, to nie ma znaczenia, który klawisz wybierzesz. Gdyby w Twoim pliku były już jakieś dane, to sprawa wyglądałaby trochę inaczej. Otóż a oznacza dodawanie tekstu (od angielskiego add) – nowy tekst będzie wprowadzany za bieżącym położeniem kursora. Jeśli wybierzesz i (od angielskiego insert), to dla edytora jest sygnał, że chcesz coś wstawić przed bieżącym położeniem kursora. Zwróć uwagę na te różnice, gdyż w standardowej wersji edytora rozróżnienie tych dwu komend jest bardzo ściśle przestrzegane. Możesz teraz wprowadzić kilka linii tekstu. Zauważ, że edytor "zawija" wiersze, których długość przekracza szerokość ekranu. Aby przejść do nowej linii tekstu należy wcisnąć ENTER . Aby powrócić do trybu poleceń wciśnij ESC . Ponieważ na ekranie nie pojawia się żaden sygnał pokazujący nam, że jesteśmy w trybie poleceń, proponuję wciśnięcie ESC kilka razy aż komputer zacznie piszczeć. Jak wcześniej wspomniałem, w trybie poleceń klawisze nabierają szczególnego znaczenia i teraz postaram się przedstawić podstawowe polecenia.

Do poruszania się po tekście można wykorzystać klawisze strzałek, ale czasami (zwłaszcza przy pracy w sieci) klawisze strzałek nie działają lub działają nieprawidłowo. Wówczas do poruszania się po tekście wykorzystujemy klawisze h , j , k , l oznaczające odpowiednio ← , ↓ , ↑ , → . Klawisze 0 i $ oznaczają przejście na początek/koniec bieżącego wiersza. Teraz kilka komend zmieniających tekst. Klawisz r oznacza zmianę bieżącego znaku. Stajesz kursorem na znaku, który chcesz zmienić, wciskasz r , a następnie znak jaki ma być w tym miejscu. Kolejne polecenia dotyczą kasowania. x kasuje bieżący znak (w naszym edytorze możesz użyć Delete , ale ...), d w kasuje słowo, d d kasuje bieżącą linię, D kasuje tekst od bieżącej pozycji kursora do końca linii. Skasowany tekst nie ginie bezpowrotnie. Edytor zapamiętuje go w buforze i możesz go ponownie wstawić do tekstu klawiszem p . Dzięki temu mechanizmowi możesz przenosić fragmenty tekstu. Jeśli jakiś fragment tekstu należy napisać kilka razy, to niekoniecznie musisz kilka razy wystukiwać go na klawiaturze. Wystarczy, że napiszesz go raz, skopiujesz do bufora i potem wkleisz klawiszem p . Kopiowanie słowa wykonujemy przez y w , a kopiowanie całej linii - y y . Jeśli przez pomyłkę popsułeś coś, to nie przejmuj się. Klawisz u pozwala odwołać ostatnie polecenie przywracając stan sprzed jego wykonania. Dokonujemy już operacji na znakach, słowach i liniach, ale to trochę

mało. Co zrobić gdy chcemy skopiować kilka linii? Nic prostszego. Stań kursorem w pierwszej linii, którą chcesz skopiować, napisz ile linii ma być skopiowane, wciśnij y y , przejdź do miejsca gdzie chcesz wkleić skopiowany tekst i wciśnij p . W ten sam sposób możesz kasować kilka linii lub słów. Podanie na początku komendy jakiejś liczby jest możliwe w bardzo wielu komendach i będziemy bardzo często tę właściwość wykorzystywać. Raczej proste, ale ... żeby tak jeszcze można zaznaczyć jakiś fragment i operować na zaznaczonym bloku. Program `vim` umożliwia taką operację. Polecenie v przenosi nas do trybu `VISUAL` i teraz klawiszami kursora zaznaczamy żądany blok tekstu. Na koniec wykonujemy polecenie np. y . Tym samym edytor wychodzi z trybu `VISUAL` i możemy skopiowany tekst wstawić w odpowiednie miejsce.

Napracowaliśmy się, stworzyliśmy plik z tekstem i co dalej? Należało by go zapisać. Wykonujemy to wpisując `:w` (ten napis pojawi się u dołu ekranu). Jeśli wpiszemy `:wq`, to edytor zapisze naszą pracę i zakończy działanie. Jeszcze jedno ważne polecenie : `:q!` nakazuje edytorowi natychmiastowe zakończenie pracy bez zapisywania.

Oczywiście nasz edytor posiada jeszcze wiele innych poleceń (np. wyszukiwanie tekstu, praca z kilkoma plikami jednocześnie), ale te podstawowe powinny Ci pomóc na początku znajomości. Jeśli chcesz się dowiedzieć więcej o możliwościach vim'a, zawsze możesz wcisnąć F1 , ale to będzie pomoc w języku czarodziejów (choć ostatnio pojawił się plik pomocy w języku polskim, ale jeszcze nie jest dołączany do standardowych dystrybucji). Jest kilka książek o vi, a w większości podręczników o UNIX'ie znajdziesz skrócone opisy. I oczywiście jest Internet, ale to nie na tym etapie nauki, do tego też dojdziemy.

Skoro mamy już narzędzie do edycji plików, to możemy wrócić do systemu.

4.6 Poprawianie bash'a - customizacja

Zaczniemy od usprawnień naszego środowiska. Pierwszym elementem, który będziemy poprawiać jest znak zachęty (prompt). Opis znaku zachęty znajduje się w zmiennej środowiskowej `PS1`. Aby zmienić wartość zmiennej środowiskowej wystarczy wpisać :

```
[tuptus@dreptak tuptus]$ PS1='Czekam=>'
Czekam=>
```

Kilka słów wyjaśnienia. Pomiędzy nazwą zmiennej, znakiem = i wartością zmiennej nie wolno wstawiać spacji. A jak wrócić do poprzedniej postaci? Trzeba zastosować specjalne znaki :

```
Czekam=>PS1='[\u@\h \W]$'
[tuptus@dreptak tuptus]$
```

Konstrukcja taka jest poprawna w bash'u. W innych shell'ach nie będzie działać. Poszczególne znaki mają następujące znaczenie :

\u nazwa użytkownika

\h nazwa hosta

\W nazwa katalogu bieżącego

\w ścieżka do katalogu bieżącego

Takich zaklęć jest więcej. Informacje o nich znajdziesz w manualu do bash'a.

Kolejnym elementem do poprawy będzie polecenie `rm`. Jak zapewne pamiętasz działanie tego polecenia jest nieodwracalne. Możemy jednak w pewnym sensie zmniejszyć ryzyko skasowania plików, które jednak będą nam potrzebne uruchamiając polecenie z parametrem `-i`. Parametr ten spowoduje interaktywne działanie polecenia – przed skasowaniem pliku system zapyta, czy

rzeczywiście chcesz ten plik skasować. Aby nie zaprzątać sobie głowy pamiętaniem o tym parametrze stworzymy alias do polecenia `rm` :

```
[tuptus@dreptak tuptus]$ alias rm='rm -i'
```

W podobny sposób możemy postąpić z innymi poleceniami np. :

```
[tuptus@dreptak tuptus]$ alias dir='ls -al --color'
```

Od tej pory polecenie `dir` wyświetli listę plików w pełnym formacie i w kolorach. Aliasy możemy usuwać poleceniem `unalias`. Ponieważ polecenia `alias` i `unalias` są poleceniami bash'a, po więcej informacji ponownie odsyłam Cię do manuala tej powłoki (`help alias`).

Powyższe zaklęcia są przydatne, ale przy następnym logowaniu nie będzie po nich żadnego śladu. Aby nie trzeba ich ponownie wpisywać ręcznie warto je umieścić w jakimś pliku, który będzie przetwarzany w momencie zalogowania. W swoim katalogu domowym znajdziesz dwa interesujące nas pliki : `.bashrc` i `.bash_profile`. Wymieniam dwa pliki, gdyż w zależności od sposobu wywołania powłoki przetwarzany jest jeden lub oba pliki. Nasze zaklęcia wpisujemy do jednego z plików w zależności od przeznaczenia :

```
[tuptus@dreptak tuptus]$ vim .bashrc
```

Polecenia basha takie jak `alias` oraz polecenia konfigurujące samego basha.

```
[tuptus@dreptak tuptus]$ vim .bash_profile
```

Polecenia konfigurujące środowisko wykonywane w momencie logowania. Szczególnie ważne jest ustawienie zmiennej środowiskowej `$PATH`. Zmienna ta przechowuje ścieżki przeszukiwania, czyli informacje o katalogach, w których shell ma poszukiwać zaklęć.

4.7 Zmienne środowiskowe

Skoro już wspomniałem o zmiennej `PATH` zajmijmy się dokładniej zmiennymi środowiskowymi. Pierwszą sprawą jest zrozumienie czym są zmienne środowiskowe. Otóż są to obszary pamięci przechowujące dane. Obszary te identyfikowane są przez nazwy. Przyjęło się, że nazwy zmiennych środowiskowych zapisujemy wielkimi literami w odróżnieniu od zmiennych wykorzystywanych lokalnie w skryptach. Wartości przechowywane w zmiennych środowiskowych to informacje istotne dla funkcjonowania systemu lub poszczególnych aplikacji. Część tych zmiennych ustawiana jest przy starcie systemu w trakcie wykonywania skryptów startowych z katalogu `/etc/rc.d`. Większość zmiennych ustawiana jest w sposób specyficzny dla użytkownika w trakcie procesu logowania. Po poprawnej autoryzacji następuje uruchomienie powłoki w trybie logowania. W trakcie tego uruchomienia wykonywane są pliki `/etc/profile` i `/etc/bashrc` oraz pliki z katalogu domowego użytkownika `.bash_profile` i `.bashrc`. Wymienione pliki są tak naprawdę skryptami powłoki zawierającymi polecenia ustawiające odpowiednie wartości zmiennych środowiskowych. Również niektóre aplikacje powodują ustawienie zmiennych środowiskowych. Zaklęciem takim jest polecenie `su` wykorzystywane do chwilowej zmiany tożsamości (np. przelogowanie się na roota). Aplikacja ta ustawia zmienne `USER`, `LOGNAME`, `SHELL` i `HOME` według informacji odpowiednich dla danego użytkownika odczytanych z pliku `/etc/passwd`.

Wiemy już czym są zmienne środowiskowe i gdzie są ustawiane. Zajmiemy się teraz wykorzystaniem tych informacji. Jak zwykle zaczynamy od zapoznania się z tym, co już jest dostępne. Listę zmiennych środowiskowych wraz z ich wartościami możemy obejrzeć wykorzystując dwa polecenia `env` oraz `set`. Oba polecenia podane bez parametrów wyświetlają listę zmiennych.

Zapewne zauważyłeś, że lista zmiennych podawanych przez polecenie `set` jest dłuższa. Wynika to z faktu, że polecenie `set` pokazuje wszystkie zmienne obowiązujące w aktualnym środowisku, natomiast polecenie `env` tylko te, które są dziedziczone przez środowisko potomne (są przekazywane do środowiska potomnego). Co z tego wynika? Całkiem sporo. Jeśli zmienna jest dziedziczona przez środowisko potomne, to znaczy, że wszystkie skrypty i aplikacje będą znały wartość tej zmiennej i będą z niej mogły korzystać. Korzystać to nie znaczy zmieniać. Zmiana wartości zmiennej środowiskowej w procesie potomnym nie ma wpływu na jej wartość na poziomie środowiska rodzica. Za moment prześledzimy te właściwości na przykładach, ale wcześniej jeszcze jedno wyjaśnienie. Zapewne zastanawiasz się czym różnią się zmienne dziedziczone od tych niedziedziczonych. Już wyjaśniam. Aby dana zmienna mogła być dziedziczona musi zostać wyeksportowana. Wykonujemy to poleceniem basha `export ZMIENNA`. Od momentu wyeksportowania zmienna będzie dostępna we wszystkich procesach potomnych naszego środowiska, ale nie w procesach nadrzędnych.

Zajmijmy się zatem praktycznym wykorzystaniem zdobytej wiedzy. Częstym problemem, z którym borykają się początkujący adepci wiedzy tajemnej jest brak "dostępu" do zaklęcia. Oto przykład. Zgłosiłeś na forum dyskusyjnym pewien problem i zostałeś poproszony o podanie wyniku polecenia `ifconfig -a` (wyświetla informację o konfiguracji interfejsów sieciowych), więc wykonujesz i . . . :

```
[tuptus@dreptak tuptus]$ ifconfig -a
bash: ifconfig: command not found
[tuptus@dreptak tuptus]$
```

Czyżby w naszym systemie nie było tak podstawowego zaklęcia? Sprawdźmy to wykorzystując polecenie `whereis` :

```
[tuptus@dreptak tuptus]$ whereis ifconfig
ifconfig: /sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
[tuptus@dreptak tuptus]$
```

Okazuje się, że jest i jest nawet manual do niego. Dlaczego zatem shell poinformował, że nie może go znaleźć? Przyczyna tkwi w ustawieniu zmiennej środowiskowej `PATH`. Zmienna ta jest ustawiana w procesie logowania w ramach wykonania pliku `/etc/profile` i przechowuje listę katalogów, które shell będzie przeszukiwał w celu znalezienia podanego przez nas zaklęcia. Sprawdźmy jak jest ona aktualnie ustawiona :

```
[tuptus@dreptak tuptus]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/tuptus/bin
```

Faktycznie, katalogu `/sbin` nie ma w zmiennej `PATH`. W wiekszości przypadków zwykli użytkownicy nie potrzebują dostępu do katalogów `sbin` (w systemie jest kilka takich), ale ty nie jesteś zwykłym użytkownikiem tylko adeptem wiedzy wszelakiej.

Uwaga. Zwróć uwagę, że w celu odczytania wartości zmiennej przed jej nazwą stawiamy znak $. W shellu jest jeszcze kilka innych możliwości odwoływania się do wartości zmiennej. Jeśli w przykładowych skryptach będziemy je wykorzystywać, to wyjaśnię ich znaczenie. W tej chwili dociekliwych czytelników odsyłam do dokumentacji basha.

Wniosek z powyższych rozważań nasuwa się sam. Zmienną `PATH` należy zmodyfikować. Zaczniemy modyfikacje środowiska od drobnych kroczków :

```
[tuptus@dreptak tuptus]$ PATH=/sbin:$PATH
[tuptus@dreptak tuptus]$ ifconfig -a
```

```
eth0      Link encap:Ethernet  HWaddr 00:50:04:67:CC:93
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
(...)
[tuptus@dreptak tuptus]$
```

Jak widzisz polecenie zadziałało. Możesz jeszcze sprawdzić jaką obecnie zawartość ma zmienna `PATH`. Oczywiście nowa wartość tej zmiennej będzie widziana również w procesach potomnych, gdyż już wcześniej uzyskała ona właściwość dziedziczenia. Możemy to sprawdzić pisząc polecenie `bash -c 'echo $PATH'`.

Pozostaje jeszcze jeden problem do rozwiązania. Po ponownym zalogowaniu zmienna będzie miała ponownie wartość początkową. Oczywiście do modyfikacji pliku `/etc/profile` nie mamy uprawnień, a jeśli nawet znasz hasło roota, to odradzam modyfikacje wspomnianego pliku, gdyż będzie to miało wpływ na wszystkich użytkowników systemu. Tego typu modyfikacje należy wprowadzać do pliku `.bash_profile` znajdującego się w katalogu domowym. W pliku tym już jest linia modyfikująca ścieżki dostępu — dopisz na końcu potrzebne Ci katalogi.

Teraz sprawdźmy co się będzie działo gdy zmienną tą ustawimy w procesie potomnym. Przejdź do drugiego terminala (Alt | Ctrl | F2 w środowisku tekstowym lub otwórz nowe okno terminala w środowisku graficznym) i sprawdź aktualną wartość zmiennej `PATH`. Teraz uruchom powłokę potomną, zmień wartość zmiennej, sprawdź jej wartość i wróć do poprzedniej powłoki.

```
[tuptus@dreptak tuptus]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/tuptus/bin
[tuptus@dreptak tuptus]$ bash
[tuptus@dreptak tuptus]$ PATH=/sbin:$PATH
[tuptus@dreptak tuptus]$ echo $PATH
/sbin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/tuptus/bin
[tuptus@dreptak tuptus]$ exit
[tuptus@dreptak tuptus]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/tuptus/bin
```

Polecenie `bash` uruchamia powłokę potomną. Jak widać wartość zmiennej `PATH` została zmieniona w powłoce potomnej, ale po powrocie do powłoki rodzimej jej wartość ponownie jest taka, jak przed zmianą. Ten przykład obrazuje to, o czym wcześniej wspominałem, **zmiany eksportowanej zmiennej dotyczą tylko procesów potomnych i nie mają wpływu na powłokę "rodzica"**.

Zajmiemy się teraz tematem eksportowania zmiennych środowiskowych. Omówimy go na przykładzie zmiennej `EDITOR`. Zmienna ta odczytywana jest przez wiele aplikacji w celu określenia domyślnego edytora wykorzystywanego przez te aplikacje. Doskonałym przykładem jej wykorzystania są demony zegarowe (crontab i at) omówione w dalszej części kursu. I znowu zaczniemy od sprawdzenia co nam oferuje system :

```
[tuptus@dreptak tuptus]$ echo $EDITOR

[tuptus@dreptak tuptus]$
```

Wygląda na to, że nie oferuje nam niczego ciekawego. Wypełnijmy zatem tę lukę :

```
[tuptus@dreptak tuptus]$ EDITOR=/usr/bin/vim
[tuptus@dreptak tuptus]$ echo $EDITOR
/usr/bin/vim
[tuptus@dreptak tuptus]$
```

Wygląda to całkiem obiecująco. Mamy ustawioną zmienną środowiskową, tylko czy aplikacje będą umiały z niej skorzystać? Sprawdźmy co zobaczy proces potomny :

```
[tuptus@dreptak tuptus]$ bash -c 'echo $EDITOR'

[tuptus@dreptak tuptus]$
```

I mamy problem. Okazuje się bowiem, że proces potomny (w tym przypadku polecenie `echo`) nie zobaczy nic ciekawego. Przyczyną jest brak możliwości dziedziczenia zmiennej, która nie została wyeksportowana. Zobaczmy zatem jak się zachowa `echo` po dokonaniu exportu :

```
[tuptus@dreptak tuptus]$ export EDITOR
[tuptus@dreptak tuptus]$ bash -c 'echo $EDITOR'
/usr/bin/vim
[tuptus@dreptak tuptus]$
```

Pięknie. O to właśnie nam chodziło. Wyeksportowanie zmiennej pozwala procesowi potomnemu na odziedziczenie wartości tej zmiennej i może on z niej skorzystać. Teraz pozostaje jedynie dopisać polecenia do odpowiedniego pliku i już zawsze powłoka będzie wiedziała, że naszym ulubionym edytorem jest `vim`.

Tutaj zatrzymaj się na chwilkę. Wiemy już całkiem sporo o zmiennych środowiskowych, ale mamy kolejny dylemat. Skoro proces potomny nie może zmieniać wartości zmiennych obowiązujących w środowisku rodzicielskim, to w jaki sposób skrypty "startowe" ustawiają te zmienne? W większości przypadków skrypty shella rozpoczynają się w poniższy sposób :

```
#!/bin/sh
# Tutaj jakiś komentarz

polecenie1
polecenia2
(...)
```

i mają ustawione bity `x` umożliwiające ich wykonanie. Wywołanie takiego skryptu powoduje uruchomienie interpretera `/bin/sh` jako procesu potomnego powłoki i wykonanie kolejnych poleceń zawartych w pliku. Zauważ jednak, że pliki `.bash_profile` i `.bashrc` nie mają tej pierwszej linii i nie mają ustawionych bitów `x`. Jak zatem są wykonywane? Oto rozwiązanie :

```
[tuptus@dreptak tuptus]$ . .bashrc
[tuptus@dreptak tuptus]$
```

Rozwiązaniem jest ta początkowa kropka. Wyobraź sobie, że to również jest zaklęcie. Należy je rozumieć jako polecenie *"w ramach bieżącej powłoki wykonaj polecenia znajdujące się w pliku .bashrc"*. Tak więc nie jest tworzony żaden proces potomny, a plik nie musi mieć ustawionych bitów `x`, gdyż zawiera on jedynie listę poleceń do wykonania, a faktycznym procesem wykonującym te polecenia jest bieżący shell. W ten właśnie sposób działa proces logowania i tak samo Ty możesz uruchamiać swoje skrypty modyfikujące zmienne środowiskowe. Oczywiście modyfikacje zmiennych wcześniej wyeksportowanych nie wymagają takich zabiegów, co pokazałem wcześniej.

4.8 Strumienie i potoki

4.8.1 Strumienie

Czym byłby świat bez strumieni i potoków szumiącej wody? W naszym magicznym świecie również istnieją te cuda natury. Wszystko co wprowadzamy do systemu płynie strumieniem wejściowym, to co system wysyła do nas płynie strumieniem wyjściowym, a komunikaty błędów przepływają strumieniem błędów. Ponieważ komputer to inteligentniejsze liczydło, więc łatwiej mu posługiwać się liczbami niż nazwami. W przypadku strumieni nie jest inaczej. Standardowy

strumień wejściowy (najczęściej klawiatura) to strumień o numerze 0, standardowe wyjście (z reguły monitor) to strumień o numerze 1, a strumień błędów ma numer 2. Skoro już wiemy jakie strumienie występują w naszym świecie i znamy ich nazwy, to możemy nimi kierować (i nie będzie to zawracanie kijem wody). Załóżmy taką sytuację : chcemy zapisać w pliku informacje o tym, jakie pliki znajdują się w naszym katalogu domowym wraz z informacjami o rozmiarze, uprawnieniach itd. Do wyświetlenia tych informacji posłuży nam znane już zaklęcie `ls`. Możemy oczywiście wszystkie dane spisać z ekranu na kartkę, a potem wykorzystując `vim'a` przepisać te dane do pliku. Tylko po co tak się męczyć? Jak prawdziwy czarodziej zmieńmy kierunek strumienia wyjścia. Niech wynik polecenia `ls` wyląduje w pliku zamiast na ekranie :

```
[tuptus@dreptak tuptus]$ ls -la > my.dir 2>&1
```

Cała tajemnica tkwi w znaczku `>`. Należy go rozumieć jako pewien ciąg poleceń :

1. otwórz plik (w naszym przypadku my.dir), jeśli nie istnieje, to utwórz nowy, jeśli istnieje, to wyczyść jego zawartość
2. zapisz standardowe wyjście do podanego pliku.
3. zamknij plik

Jeśli teraz otworzysz ten plik, to zauważysz, że na liście figuruje nasz plik z rozmiarem 0. Wyjaśnienia wymaga jeszcze tajemniczy zapis `2>&1`. Pamiętasz numery strumieni? Standardowy strumień błędów przekierowujemy na standardowe wyjście, a więc tak naprawdę do naszego pliku. W tym wypadku nie ma to większego sensu, bo żaden błąd nie pojawi się, ale ...

```
[tuptus@dreptak tuptus]$ ls -aR / >/dev/null 2>brak.dir
```

już ma sens. Jeśli wiesz co robi powyższe zaklęcie, to znaczy, że jesteś naprawdę dobry. Dodam tylko, że `/dev/null` to taka czarna dziura w systemie. Możesz do tego pliku wrzucać cokolwiek, a dane tam wpadną i znikną. Pomyśl, poczytaj manuale i na końcu wypróbuj zaklęcie (działanie zabierze sporo czasu więc bądź cierpliwy). Jeśli nadal nie rozumiesz tego zaklęcia, to wróć do manuali. To nie jest trudne zaklęcie i z całą pewnością możesz je samodzielnie rozszyfrować. To dobre ćwiczenie. Przyłóż się.

Konstrukcje pokazane powyżej mają jedną ogromną wadę. Każde wykonanie komendy powoduje zamazanie zawartości podanego pliku (u nas my.dir). Ale prawdziwi czarodzieje na wszystko mają sposób.

```
[tuptus@dreptak tuptus]$ ls -la >> my.dir
```

Zmieniliśmy znak przekierowania strumienia na `>>`. Należy go rozumieć jako ciąg następujących poleceń :

1. otwórz plik (w naszym przypadku my.dir), jeśli nie istnieje, to utwórz nowy, jeśli istnieje, to ustaw się na jego końcu
2. zapisz standardowe wyjście do podanego pliku.
3. zamknij plik

I o to nam chodziło. Teraz każde wywołanie zaklęcia dopisze nowe dane do pliku my.dir nie niszcząc wcześniej uzyskanych danych.

Do tej pory przekierowywaliśmy standardowe wyjście, ale możemy również przekierowywać standardowe wejście. Zamiast znaku `>` wstawiamy `<` np.

```
[tuptus@dreptak tuptus]$ polecenie < dane > wynik
```

To tylko przykład!!!
Do zaklęcia `polecenie` (nie ma go w systemie, to przykład) wysyłamy dane z pliku `dane` zamiast z klawiatury, a wynik działania zapisujemy w pliku `wynik`. Może to głupi przykład, ale nic mądrzejszego nie przychodzi mi do głowy.

4.8.2 Potoki

W naturalnym świecie strumienie łączą się tworząc potoki. W naszym świecie jest podobnie. Najczęstszym zjawiskiem jest połączenie standardowego wyjścia pierwszego zaklęcia ze standardowym wejściem kolejnego zaklęcia. Ale jak to wykorzystać? Możesz dla przykładu wyświetlić listę plików z Twojego katalogu domowego według wielkości.

```
[tuptus@dreptak tuptus]$ ls -l | sort -n -k 5
```

Tajemnicą jest znak `|`. Dzięki temu znaczkowi standardowe wyjście polecenia `ls` zostaje połączone ze standardowym wejściem zaklęcia `sort`, które sortuje otrzymane dane numerycznie/liczbowo (`-n`) według piątej kolumny danych (`-k 5`). Oczywiście prawdziwy czarodziej nigdy nie zastosuje takiego zaklęcia, bo jest zbyt długie i działa raczej wolno. Podałem je tylko w celu pokazania działania potoków. Prawdziwy czarodziej wpisze tak :

```
[tuptus@dreptak tuptus]$ ls -lSr
```

i osiągnie tan sam efekt znacznie szybciej. Nie oznacza to wcale, że potoki są zbędne :

```
[tuptus@dreptak tuptus]$ du -sk * | sort -n
(...)
[tuptus@dreptak tuptus]$
```

Ten zestaw zaklęć podaje zajętość systemu plików przez pliki i katalogi w kilobajtach (`du -sk *`) w kolejności rosnącej (`sort -n`). Polecenie `du` nie ma opcji sortowania, więc nie można inaczej uzyskać takiego efektu.

4.9 Poszukiwany, poszukiwana

Znasz już polecenia `cd` i `ls`. Można za ich pomocą przeglądać zawartość poszczególnych katalogów i dzięki temu przeszukiwać cały system plików. Jeśli jednak chciałbyś poszukać jakiegoś konkretnego pliku i do tego nie znasz jego pełnej nazwy, to byłaby to straszna męczarnia. Jest jednak kilka zaklęć, które ułatwiają czarodziejom odnajdowanie potrzebnych danych.

Pierwszym poleceniem, któremu przyglądniemy się dokładniej jest polecenie `whereis`. Zaklęcie to ułatwia odnalezienie innych zaklęć oraz dokumentacji do nich. Nie przeszukuje ono całego systemu plików, a jedynie miejsca, w których standardowo pliki takie się znajdują. Niedogodnością jest konieczność znania pełnej nazwy polecenia. Komendę tą wykorzystywaliśmy już szukając polecenia `ifconfig`. Podając polecenie bez żadnych parametrów, poza nazwą interesującego nas zaklęcia, nakazujemy `whereis` przeszukanie całego systemu plików i podanie wszystkiego, co znajdzie na temat szukanego zaklęcia. Możemy jednak ograniczyć zakres poszukiwań oraz wyświetlane informacje. Sprawdźmy praktyczny przykład. W katalogu `/usr/bin` znajduje się zaklęcie `apt-get` (w dalszej części będziemy o nim mówić dokładniej). Dowiedzmy się czy ma jakąś dokumentację, a jeśli tak to gdzie.

```
[tuptus@dreptak tuptus]$ whereis -m -B /usr/bin -f apt-get
apt-get: /usr/share/man/man8/apt-get.8.gz
[tuptus@dreptak tuptus]$
```

Okazuje się, że dokumentacja znajduje się w ósmym rozdziale manuala systemowego. Co oznaczają poszczególne parametry? `-m` ogranicza wyświetlane informacje wyłącznie do manuali. W tym miejscu może wystąpić jeszcze `-b` — binaria i `-s` — źródła. `-B katalog` ogranicza zakres poszukiwań, w tym wypadku poszukiwań binariów. Podobnie jak poprzednio można ograniczyć zakres poszukiwań manuali wykorzystując parametr `-M`. Jeśli w poleceniu występują parametry "wielkich liter" to wystąpić musi również parametr `-f`, a po nim nazwa poszukiwanego pliku. Parametr ten oddziela listę parametrów poszukiwań od nazwy poszukiwanego pliku.

Oczywiście jest to bardzo ciekawe polecenie, ale to jeszcze nie jest to. Nie możemy z jego pomocą szukać dowolnego pliku. Istnieje jednak zaklęcie dużo potężniejsze od `whereis` — `find`. Śmiało można powiedzieć, że `find` to cały system poszukiwań. Wiemy już, że w `/usr/bin` jest `apt-get`, sprawdźmy zatem czy są tam inne zaklęcia związane z apt :

```
[tuptus@dreptak tuptus]$ find /usr/bin -name "apt*"
/usr/bin/apt-cache
/usr/bin/apt-cdrom
/usr/bin/apt-config
/usr/bin/apt-get
/usr/bin/apt-shell
[tuptus@dreptak tuptus]$
```

Jak widać jest ich kilka. Teraz kilka słów wyjaśnienia. Pierwszym parametrem polecenia `find` zawsze jest katalog stanowiący początek drzewa katalogów przeznaczonych do przeszukiwania. Dalej następują parametry określające czego właściwie szukamy. W naszym przypadku, parametr `-name` określa, że szukamy plików o nazwach zgodnych z podanym, jako następny parametr, wzorcem. Wzorzec ten należy podawać w cudzysłowiu, aby uniknąć rozwinięcia go przez powłokę. Na końcu należy podać polecenie do wykonania na wynikach wyszukiwania. Ponieważ nas interesuje lista plików pasujących do wzorca, to należałoby podać parametr `-print`. Ponieważ jednak parametr ten jest domyślnym parametrem, to w przykładzie nie podawałem go.

To było w miarę proste. Zajmiemy się teraz nieco trudniejszym przykładem. W rozdziale o uprawnieniach poznałeś bity modyfikujące. Ponieważ ich stosowanie może prowadzić do poważnych zagrożeń systemu jednym z zadań administratora jest sprawdzanie, czy nic w tej dziedzinie się nie zmieniło. Wprawdzie zwykły użytkownik nie może tych bitów ustawiać, ale istnieją programy omijające to zabezpieczenie i tu mogą pojawić się problemy. Pierwszą sprawą jaką powinieneś wykonać po zainstalowaniu systemu jest stworzenie listy plików z ustawionymi przez system bitami suid. Nasze polecenie `find` musi sprawdzać czy dany obiekt jest plikiem oraz jak ma ustawione uprawnienia, a wynik zapiszemy do pliku. Oczywiście zadanie to wykonać należy z konta root.

```
[root@dreptak root]# find / -type f -a -perm +4000 >bity_s 2>/dev/null
[root@dreptak root]#
```

Parametr `-type f` nakazuje sprawdzanie tylko plików, parametr `-perm +4000` poleca wyszukać pliki, które mają ustawiony co najmniej bit suid a parametr `-a` to logiczne `and`. Wyniki zapytania zostaną zapisane do pliku bity_s, a ewentualne błędy zostaną zignorowane. Co pewien czas powinieneś wykonywać podobne polecenie i wyniki porównywać z powstałym przed chwilą plikiem.

Niektórzy administratorzy zabraniają użytkownikom uruchamiania aplikacji spoza dystrybucji, chyba że zostaną wcześniej zaakceptowane przez administratora. Aby mieć kontrolę nad katalogami użytkowników można okresowo wykonywać polecenie `find` i jego wyniki porównywać z listą plików wykonywalnych "dopuszczonych" do użytkowania.

```
[root@dreptak root]# find /home  -perm +111 -a -type f -a ! -path "*tuptus*"\
> -print > exeki
[root@dreptak root]#
```

W powyższym przykładzie szukamy plików z ustawionym co najmniej jednym bitem `x`. Przeszukujemy wszystkie podkatalogi katalogu `/home` z wyjątkiem katalogu `tuptus` (siebie nie musimy sprawdzać). Przy bardzo restrykcyjnych założeniach możemy natychmiast kasować znalezione pliki wykorzystując polecenie `xargs`.

```
[root@dreptak root]# find /home  -perm +111 -a -type f -a ! -path "*tuptus*"\
> | xargs rm -f
[root@dreptak root]#
```

To bardzo restrykcyjna polityka i przed jej zastosowaniem poważnie rozważ wszystkie za i przeciw.

4.10 Demony

Prawdziwy czarodziej nie brudzi sobie rąk ciężką i mozolną pracą. Do takich zadań najlepiej zatrudnić demona. W naszym systemie żyją demony tak samo jak w każdym magicznym świecie. Czym są demony w systemie UNIX? Demony to procesy uruchamiane przy starcie systemu (z reguły) i pracujące niezależnie od działań użytkowników. Nauczymy się teraz jak oswoić dwa z tych demonów.

4.10.1 Cron Daemon

Pierwszym omawianym demonem będzie cron. Ten demon czyta pliki konfiguracyjne przeznaczone dla niego i na podstawie zawartych w nich wpisach wykonuje zadania w określonych odstępach czasu. Do plików tych nie należy sięgać bezpośrednio, lecz przez polecenie `crontab`. Aby zobaczyć jakie zadania ma do wykonania cron wpisujemy `crontab -l`. Oczywiście w tej chwili nie zobaczymy nic ciekawego, bo jeszcze nic nie zlecaliśmy cronowi, a widzimy tylko zlecenia własne. Aby zlecić cronowi jakieś zadania, to należy je wpisać do pliku konfiguracyjnego. Wykonujemy to wydając polecenie `crontab -e`. Zanim jednak zaczniemy oswajać corna musimy poznać kilka szczegółów.

Po pierwsze edytor. Do edycji pliku konfiguracyjnego wykorzystywany jest vi. Jeśli chcesz wykorzystywać inny edytor, to musisz ustawić i wyeksportować zmienną środowiskową `EDITOR`. Proponuję zrobić odpowiednie wpisy w pliku startowym .bash_profile.

Kolejna sprawa to format pliku konfiguracyjnego. Każde zlecenie zapisujemy w jednej linii. Linia zlecenia składa się z sześciu pól oddzielonych spacją :

1. pole minut – podajemy minutę, w której powinno być wykonane zlecenie lub interwał (np. zapis */5 oznacza co pięć minut)
2. pole godzin – podajemy godzinę, w której powinno być wykonane zlecenie lub interwał
3. dzień miesiąca – podajemy dzień miesiąca, w którym powinno być wykonane zlecenie
4. pole miesięcy – podajemy miesiąc, w którym powinno być wykonane zlecenie
5. dzień tygodnia – podajemy dzień tygodnia, w którym powinno być wykonane zlecenie
6. pole komendy – wszystko aż do końca linii jest traktowane jako polecenie do wykonania

Omawiając polecenie `find` przedstawiłem technikę przeszukiwania katalogów domowych pod kątem występowania plików wykonywalnych. Stosowanie tej techniki jest mocno dyskusyjne. Jednak w analogiczny sposób możemy przeszukiwać dowolną część systemu plików. Przykładem

katalogu, do którego warto zastosować bardzo restrykcyjne podejście jest katalog `/tmp`. W katalogu tym przechowywane są pliki tymczasowe i nie ma żadnego uzasadnienia umieszczania w nim plików wykonywalnych. Jednocześnie wiele technik przełamywania zabezpieczeń polega na wykorzystaniu właśnie tego katalogu. Zatem stwórzmy sobie strażnika w oparciu o `crona`. Jako root wpisujemy polecenie `crontab -e` i tworzymy strażnika :

```
*/30 * * * * /usr/bin/find /tmp -type f -perm +111 -print | xargs
 rm -f >/dev/null 2>&1
```

Oczywiście powyższy wpis powinien znaleźć się w całości w jednej linii i tutaj jest przedstawiony w dwóch liniach tylko ze względów edycyjnych.

4.10.2 At Daemon

Kolejnym demonem zegarowym jest `atd`. Demon ten wykonuje jednokrotnie zadanie, które mu zlecimy. Aby przekazać demonowi `atd` czego od niego oczekujemy należy użyć polecenia `at` podając jako parametr czas, w którym ma być wykonane zadanie. W tym momencie otworzy nam się okno pozwalające na wpisanie listy poleceń do wykonania. Pracę możemy sobie ułatwić tworząc plik z odpowiednią listą poleceń, a następnie wywołać plecenie `at` z dodatkowym parametrem `-f plik`.

Czas możemy podawać na wiele sposobów. Podstawowym jest format hh:mm (godziny:minuty). Jeśli podany czas już minął to traktowany będzie jako czas następnego dnia. Możemy podać dokładnie datę wyknania w formacie dd.mm.yy hh:mm (dzien.miesiąc.rok godzina:minuta). Takie formaty wyglądają całkiem naturalnie, ale `at` rozumie znacznie więcej. Jak podoba Ci się zapis *"midnight +10minutes"*? Oznacza on godzinę 00:10. A co myślisz o takim zapisie *"teatime tomorrow"*? Ten zapis oznacza godzinę szesnastą następnego dnia.

Uwaga. Jeśli do określenia czasu wykorzystujemy nazwy takie jak `midnight`, to demon ustawia czas na odpowiednią godzinę w bieżącym dniu niezależnie od tego czy czas ten już minął. Zatem jeśli chcesz, aby zadanie wykonało się 10 minut po najbliższej północy, to prawidłowe określenie czasu powinno wyglądać tak *"midnight tomorrow +10minutes"*.

A teraz kolej na wykorzystanie zdobytej wiedzy. Wyobraź sobie, że masz w swoim systemie niepokornego użytkownika korzystającego z konta hakier (to nie jest błąd, takie konto już widziałem). Użytkownik ten, "pracując" po nocach, zachowuje się co najmniej niegrzecznie uważając, że skoro Ciebie nie ma, to nic mu nie grozi. Zróbmy mu zatem niespodziankę. Zaczniemy od stworzenia pliku z listą poleceń do wykonania. Wejdz na konto root, utwórz plik np. test i wpisz poniższy tekst :

```
if [ `w | grep hakier | wc -l` -ge 1 ]
then
  echo "Admin czuwa" | mail -s "Ostrzezenie" hakier
fi
```

Może na pierwszy rzut oka wygląda to bardzo tajemniczo, ale jest to dziecinnie prosty skrypcik. Polecenie `w` wyświetla listę aktualnie zalogowanych użytkowników. Przepuszczamy jego wynik przez filtr `grep`, który w potoku pozostawi tylko linie zawierające słowo `hakier`. Wynik filtrowania trafia do polecenia `wc`, ktore w tej formie poda ilość linii otrzymanych na wejściu. Teraz następuje sprawdzenie (`-ge`) czy wynik jest większy lub równy jeden. Inaczej mówiąc sprawdzamy czy hakier pracuje na jakiejś konsoli. Jeśli test jest prawdziwy, wówczas wynik polecenia

`echo` przesyłamy do polecenia `mail`, a to polecenie wysyła otrzymany tekst do użytkownika hakier jako list o tytule zawartym w parametrze następującym po `-s`. Wprawdzie wykorzystaliśmy kilka wcześniej nie omówionych poleceń, ale konstrukcja nie powinna stanowić dla Ciebie żadnej zagadki. O pisaniu skryptów jeszcze porozmawiamy dokładniej, a teraz wracamy do naszego demona. Wpisz polecenie :

```
[root@dreptak root]# at -f test midnight tomorrow +7minutes
job 3 at 2004-05-24 00:07
[root@dreptak root]#
```

W odpowiedzi na Twoje polecenie `at` informuje jaki numer przydzielił zadaniu oraz podaje godzinę wykonania. Jeśli siedem minut po północy nasz hakier będzie znowu zalogowany, to otrzyma naszą wiadomość. Oczywiście prawdziwy czarnoksiężnik jedynie uśmiechnie się pod nosem, ale hakier ...

Część V

Administracja

Rozdział 5

Konta użytkowników

Do tej pory posługiwaliśmy się pojęciem konta użytkownika przemilczając wiele istotnych informacji dotyczących tego zagadnienia. W rozdziale niniejszym zajmiemy się dokładniej problematyką zarządzania kontami użytkowników systemu.

5.1 Właściwości konta

Każdy obiekt ma swoje właściwości. Również konto użytkownika ma swoje właściwości :

nazwa nazywana również login lub username, nazwa konta identyfikuje użytkownika w systemie i jest znana publicznie

identyfikator to numer konta w systemie, komputer łatwiej sobie radzi z liczbami niż nazwami i z tego powodu na potrzeby systemu operacyjnego, do identyfikacji użytkownika wykorzystywany jest identyfikator numeryczny nazywany również numerem ID

hasło jest informacją znaną jedynie użytkownikowi, w systemie przechowywana jest informacja o haśle, ale nie samo hasło, hasło niezbędne jest do dokonania autoryzacji dostępu użytkownika do systemu

grupa każdy użytkownik systemu należy do co najmniej jednej grupy użytkowników

katalog domowy każdemu użytkownikowi przydzielany jest pewien obszar systemu plików nazywany katalogiem domowym, w którym przechowywane są pliki konfigurujące środowisko pracy użytkownika oraz jego zasoby prywatne

shell jest to domyślny interpreter poleceń przydzielony użytkownikowi, w przypadku systemu Linux jest to najczęściej bash

czasy to grupa właściwości konta określających czas ważności konta, datę zmiany hasła itp.

Przedstawione cechy to jedynie te najbardziej podstawowe. Część z nich już omawialiśmy, więc nie będziemy do nich wracać. Pozostałe omówimy na przykładzie. Zanim jednak przejdziemy do przykładu przyjrzyjmy się systemowi plików i jego powiązaniom z kontami użytkowników. Podstawą informacji o koncie są informacje zawarte w pliku `/etc/passwd`. Dla każdego konta tworzona jest jedna linia w powyższym pliku w następującym formacie :

```
nazwa:haslo:id:gid:opis:katalog domowy:shell
```

Elementy nazwa, id, katalog domowy i shell już znasz z wcześniejszych informacji. Pozostałe pola wymagają pewnego omówienia.

Hasło nie jest przechowywane w pliku `passwd`, chociaż nazwa pliku i opis podany wyżej na to wskazuje. Jest to jednak pozostałość po starych systemach uniksowych, w których faktycznie hasło występowało w tym miejscu. W polu hasła widnieje znak `x` informujący o tym, że hasło znajduje się w innym pliku, a konkretnie `/etc/shadow`.

Kolejne pole wymagające wyjaśnienia to `gid`. Pole to zawiera identyfikator grupy głównej użytkownika nazywanej również grupą początkową. Grupa główna to grupa domyślna wykorzystywana w momencie tworzenia przez użytkownika nowych plików. Jeśli użytkownik należy również do innych grup, to grupy te są grupami dodatkowymi danego użytkownika.

Pozostało jeszcze pole `opis`. Pole to zawiera informację opisową dotyczącą konta i nie ma znaczenia dla systemu operacyjnego. Przykładowa linia opisująca konto `test` może wyglądać następująco :

```
test:x:504:504:User testowy:/home/test:/bin/bash
```

Skoro już wspomniałem o pliku `shadow`, to przyglądnijmy się bliżej temu plikowi. Podobnie jak w pliku `passwd` dla każdego konta tworzona jest jedna linia, a jej format wygląda następująco:

```
nazwa:haslo:data zmiany hasla:min:max:warn:inact:expire:flag
```

Pole nazwa jest identyczne z polem w pliku `passwd` i nie wymaga komentarza.

Pole hasło wymaga jednak omówienia. W polu tym przechowywana jest informacja o haśle, a nie samo hasło. Wcześniej już omawiałem sposób korzystania z tej informacji. Opiera się on o wykorzystanie algorytmu szyfrowania jednokierunkowego MD5. Jednokierunkowość oznacza, że raz zaszyfrowany ciąg znaków nie może zostać w żaden sposób odszyfrowany. Przykładowe "hasło" może wyglądać następująco : `$1$xsH6r1yC$XqTkhvRGG9rX/Q3NeRjtR/`. Początkowe dwanaście znaków tego ciągu to klucz szyfrujący, a reszta to wynik szyfrowania hasła. Klucz szyfrujący składa się z `$1$`, ośmiu losowych znaków i kończy się znakiem `$`. Ten losowy ciąg znaków generowany jest w momencie tworzenia lub zmiany hasła, co ma na celu dodatkowe utrudnienie przy próbie odgadnięcia hasła. Jednocześnie zapisanie klucza w pliku jest niezbędne do późniejszego wykorzystania przy autoryzacji użytkownika. Otóż w czasie logowania, wpisane przez użytkownika hasło jest szyfrowane z wykorzystaniem klucza znajdującego się w pliku `shadow`, a wynik porównywany jest z ciągiem znaków począwszy od trzynastego. Przykład podany powyżej charakterystyczny jest dla konta, dla którego zostało ustawione hasło. Ale pole to może zawierać również inne znaki. Przykładowo `!!` oznacza, że dla takiego konta nie ustalono hasła i nie można się na takie konto zalogować. Wyjątkiem jest przejście na to konto użytkownika root wykorzystującego polecenie `su`. Znak `*` w polu hasła całkowicie blokuje możliwość logowania również użytkownikowi root (jeden z nielicznych przypadków ograniczenia władzy roota). Zdarzyć się również może, choć bardzo rzadko, że pole hasła jest puste. W takiej sytuacji po wpisaniu nazwy użytkownika system już nie pyta o hasło tylko natychmiast udostępnia użytkownikowi powłokę. Domyślna konfiguracja systemu nie dopuszcza do takich sytuacji jednak plik `shadow` jest zwykłym plikiem tekstowym, więc wykorzystując edytor tekstu możemy ręcznie "wyczyścić" pole hasła. Stanowczo odradzam takie postępowanie z oczywistych względów, a informację powyższą podaje dlatego, że może Ci ona uratować skórę w sytuacjach awaryjnych.

Pole "data zmiany hasla" zwiera faktycznie tę datę, ale w formie mało zrozumiałej dla człowieka. Otóż w systemach uniksowych za datę początkową przyjęto 1 styczeń 1970 roku i wszystkie daty podawane są jako okres czasu od tego momentu. Dokładnie tak samo jest w tym przypadku. Pole to zawiera liczbę dni pomiędzy datą początkową a datą zmiany hasła.

Pole "min" zawiera informację o ilości dni, jaka musi upłynąć pomiędzy kolejnymi zmianami hasła. Z reguły wartość tego pola wynosi 0, ale w systemach o podwyższonych wymogach bezpieczeństwa administratorzy często ustawiają tu pewną wartość. Chodzi o to, aby wymusić na

użytkownikach okresową zmianę haseł, a jednocześnie uniknąć sytuacji, gdy użytkownik próbując ominąć te wymogi zmienia hasło na nowe i natychmiast dokonuje ponownej zmiany wracając do poprzedniego hasła.

Pole "max", analogicznie do pola "min", zawiera informację o ilości dni, po których hasło musi zostać zmienione. Na podstawie tego pola oraz pola daty zmiany hasła system wylicza datę wygaśnięcia ważności hasła. W oparciu o tak wyliczoną datę, datę aktualną oraz wartość z pola "warn" użytkownik jest odpowiednio wcześniej informowany o zbliżającym się terminie utraty ważności obecnego hasła. W przypadku braku ograniczenia czasu ważności hasła pole to zawiera wartość 99999.

Pole "warn" określa na ile dni przed datą wygaśnięcia hasła użytkownik ma być informowany o tym fakcie. Domyślnie pole to zawiera wartość 7.

Pole "inact" wprowadza dodatkowe restrykcje związane z wygasaniem hasła. Zdarzyć się może, że hasło wygasło w czasie, gdy użytkownik nie miał dostępu do systemu. W sytuacji takiej konto nie jest jeszcze blokowane, ale bezpośrednio po zalogowaniu wymuszona zostaje zmiana hasła. W polu "inact" podaje się ilość dni od wygaśnięcia hasła, po których następuje definitywne zablokowanie konta w przypadku braku jego zmiany. Konto takie odblokować może jedynie root.

Pole "expire" zawiera datę ważności konta zapisaną w formacie "uniksowym". Zwróć uwagę, że jest to cecha związana z kontem, a nie hasłem. Po przekroczeniu tej daty konto zostaje zablokowane niezależnie od ważności hasła.

Ostatnie pole nie jest aktualnie wykorzystywane.

Do kompletu brakuje nam jeszcze plików `/etc/group` i `/etc/gshadow`. Plikiem `gshadow` nie będziemy się zajmować. Wykorzystywany on jest sporadycznie w zaawansowanych instalacjach. Obiektem naszych zainteresowań jest natomiast plik `group`. Podobnie jak w przypadku kont, plik ten zawiera opis grup po jednej linii na grupę. Format takiej linii jest następujący :

```
nazwa:haslo:gid:lista kont
```

Nazwa i gid nie wymagają tłumaczenia. Pole hasła zawiera znak `x` informujący, że hasło znajduje się w innym pliku (właśnie `gshadow`) Pole "lista kont" zawiera konta użytkowników należących do tej grupy oddzielone przecinkami. I tu drobna uwaga. W domyślnej instalacji Auroksa konfiguracja jest tak ustawiona, że w momencie dodawania nowego użytkownika, jeśli nie podamy grupy początkowej, zostanie utworzona grupa o nazwie identycznej z nazwą konta. Na liście kont takiej grupy nie widnieje konto, dla którego ta grupa została utworzona.

I to tyle na temat przechowywania danych o kontach i grupach. Zajmiemy się teraz plikami konfiguracyjnymi związanymi z kontami. W momencie zakładania nowego konta istotne są dla nas :

- plik `/etc/default/useradd`; plik ten zawiera wartości domyślne przyjmowane przez program `useradd`. Wprawdzie jest to zwykły plik tekstowy i można go edytować, jednak nie polecam takiego rozwiązania. Znacznie lepszym podejściem jest wywołanie programu `useradd` z parametrem `-D` oraz parametrami, które chcemy zmienić.

- katalog `/etc/skel`; katalog ten zawiera wzorzec katalogu domowego dla nowo tworzonego konta. Zawartość tego katalogu możemy swobodnie modyfikować tak, aby ułatwić sobie zakładanie nowych kont. Szczególnie ciekawe efekty możemy uzyskać modyfikując plik `.bash_profile`. W katalogu tym znajdują się podkatalogi `.kde` oraz `Desktop`. Jeśli nie przewidujemy udostępniania użytkownikom środowiska graficznego (tak będzie w przypadku serwera), to katalogi te możemy z "wzorca" usunąć.

5.2 Konta pełne

Uzbrojeni w te podstawowe informacje możemy zacząć zarządzanie kontami. Zaczniemy od sprawdzenia ustawień domyślnych dla nowo zakładanych kont :

```
[root@dreptak root]# useradd -D
GRUPA=100
KAT_DOM=/home
NIEAKTYWNE=-1
WYGAŚNIĘCIE=
POWŁOKA=/bin/bash
SKEL=/etc/skel
```

Pierwsza pozycja to domyślna grupa początkowa dla nowo tworzonego konta. *Jest ignorowana, wyjaśnić co jest przyczyną* ***TODO***

Kolejną pozycją jest domyślny katalog domowy. W procesie zakładania nowego konta we wskazanym katalogu tworzony jest katalog o nazwie identycznej z nazwą konta.

Pozycja nieaktywne opowiada polu "inact" z pliku `shadow`, czyli oznacza ilość dni pomiędzy wygaśnięciem hasła a zablokowaniem konta. Wartość -1 deaktywuje tę funkcjonalność.

Pozycja czwarta zawiera informacje o dacie wygaśnięcia ważności konta. Z ustawieniem tej pozycji należy być ostrożnym. Raz ustawionej pozycji nie można wyzerować. Jedynym wyjściem z sytuacji jest ręczna edycja pliku.

Kolejne dwie pozycje nie wymagają szerszego omówienia, więc przechodzimy do dalszych kroków. Możemy sobie zażyczyć, aby konta naszych kursantów znajdowały się w katalogu `/home/students`, a katalogi instruktorów tradycyjnie w `/home`. Ponieważ instruktorów mamy znacznie mniej niż "studentów", to przy zakładaniu dla nich kont możemy podać jawnie katalog domowy, a dla "studentów" ustawimy katalog domyślny.

```
[root@dreptak root]# mkdir /home/students
[root@dreptak root]# useradd -D -b/home/students
[root@dreptak root]# useradd test
[root@dreptak root]# finger test
Login: test                             Name: (null)
Directory: /home/students/test          Shell: /bin/bash
Never logged in.
No mail.
No Plan.
[root@dreptak root]#
```

Wygląda na to, że plan się powiódł. Ale jak teraz zakładać konta instruktorom? Można to zrobić podając katalog domowy w sposób jawny jako parametr polecenia `useradd` :

```
[root@dreptak root]# useradd -d /home/test2 -m test2
[root@dreptak root]# finger test2
Login: test2                            Name: (null)
Directory: /home/test2                  Shell: /bin/bash
Never logged in.
No mail.
No Plan.
[root@dreptak root]#
```

Parametr `-d` pozwala na podanie katalogu domowego dla nowego konta przy czym należy podać konkretny katalog, a nie miejsce, w którym ma się on znaleźć. Parametr `-m` powoduje, że katalog domowy zostanie utworzony, jeśli nie istnieje i skopiowane do niego będą pliki i katalogi z katalogu `/etc/skel`.

Jak widzisz w obu wypadkach uzyskaliśmy zamierzony efekt. Jednak użytkownicy nie mogą jeszcze korzystać ze swoich kont, gdyż nie mają ustawionych haseł. Pierwsze ustawienie hasła

wykonuje administrator systemu. Zadanie to jest wyjątkowo proste. Musisz wywołać polecenie `passwd` podając jako parametr nazwę konta, dla którego chcesz ustawić hasło. Teraz już możesz poinformować użytkowników o założeniu konta i podać im ustawione przez Ciebie hasło. Użytkownik powinien się zalogować i zmienić to hasło na sobie tylko znane. Niestety wielu mugoli ignoruje zasady bezpieczeństwa i zostawia hasło podane przez admina, szczególnie wtedy, gdy jest ono łatwe do zapamiętania. Możesz oczywiście ustawiać jakieś wymyślne hasła, ale wówczas musisz się liczyć z faktem, że użytkownicy będą takie hasła zapisywać na kartkach i również ich nie zmienią uznając, ze to dobre hasła.

Niestety nie ma prostej metody na wymuszenie zmiany hasła. Mamy jednak narzędzie umożliwiające zarządzanie hasłami. Narzędziem tym jest zaklęcie `chage`. Wykorzystując to polecenie możemy sprawdzić kiedy użytkownik zmieniał ostatnio hasło :

```
[root@dreptak root]# chage -l test
Minimum:                0
Maksimim:               99999
Ostrzeżenie:            7
Inactive:       -1
Ostatnia zmiana:        maj 25, 2004
Hasło traci ważność:    Nigdy
Hasło nieaktywne:       Nigdy
Konto traci ważność:    Nigdy
[root@dreptak root]# date
sob maj 29 20:23:56 CEST 2004
[root@dreptak root]#
```

Ostatnia zmian hasła miała miejsce cztery dni temu. Albo użytkownik zmienił hasło zaraz po założeniu konta albo nie zmienił go wcale. Pewnych informacji może nam dostarczyć polecenie `finger`.

```
[root@dreptak root]# finger test
Login: test                             Name: User
Directory: /home/students/test          Shell: /bin/bash
Office: Testy
Never logged in.
No mail.
No Plan.
[root@dreptak root]#
```

Wygląda na to, że jeszcze nie logował się ani razu. Oczywiście te informacje to zbyt mało, by stwierdzić, że nie korzysta z systemu bo udostępniamy jeszcze pocztę i ftp, a wykorzystanie tych usług nie jest widoczne przez polecenie `finger`. Mamy jednak pewność, że hasła nie zmieniał. Skoro nie korzysta z konta, to widocznie nie jest mu ono potrzebne. Konto, które nie jest używane to potencjalne zagrożenie dla bezpieczeństwa systemu. Konto blokujemy :

```
[root@dreptak root]# chage -E 2004-05-28 test
[root@dreptak root]# chage -l test
Minimum:                0
Maksimim:               99999
Ostrzeżenie:            7
Inactive:       -1
Ostatnia zmiana:        maj 25, 2004
Hasło traci ważność:    Nigdy
Hasło nieaktywne:       Nigdy
Konto traci ważność:    maj 28, 2004
[root@dreptak root]#
```

Konto zostało zablokowane. Jeśli teraz użytkownik przypomni sobie o koncie, to będzie musiał ponownie skontaktować się z administratorem i . . . bez batonika się nie obejdzie, a i do pouczenia jest okazja.

W niektórych systemach o podwyższonych wymaganiach bezpieczeństwa stosuje się politykę wymuszającą okresową zmianę hasła. Temat ten już poruszałem omawiając domyślne wartości dla polecenia `useradd` oraz strukturę pliku `shadow`. Wartości z `/etc/default/useradd` są wartościami domyślnymi, a niektórych użytkowników musimy traktować indywidualnie. Tutaj ponownie z pomocą przychodzi zaklęcie `chage`.

Załóżmy, że użytkownik test okazał skruchę i należy mu umożliwić pracę. Musimy zatem zdjąć blokadę konta. Ze względu na wcześniejsze przewiny ustawimy mu wymóg zmiany hasła na 30 dni, powiadomienie o konieczności zmiany na 7 dni wcześniej (ta wartość jest już ustawiona), minimalny czas pomiędzy zmianami na 20 dni i karencja po upływie ważności hasła na 7 dni :

```
[root@dreptak root]# chage -E -1 test
[root@dreptak root]# chage -m 20 -M 30 -I 7 test
[root@dreptak root]# chage -l test
Minimum:                20
Maksimim:               30
Ostrzeżenie:            7
Inactive:       7
Ostatnia zmiana:        maj 25, 2004
Hasło traci ważność:    cze 24, 2004
Hasło nieaktywne:       lip 01, 2004
Konto traci ważność:    Nigdy
[root@dreptak root]#
```

Powyższy opis nie wyczerpuje wszystkich możliwości opisanych poleceń i jak zwykle czytelnika odsyłam do manuala systemowego po szczegółowe informacje.

5.3 Konta bezszelowe

Wcześniejszy opis pokazuje w jaki sposób tworzyć konta "pełne", czyli takie, których użytkownicy mają dostęp do wszystkich usług świadczonych na serwerze. Jednak często zachodzi potrzeba tworzenia kont o ograniczonych prawach. Przykładem takich kont są konta udostępniające jedynie pocztę, konta tworzone na potrzeby uruchomienia serwisu www. Użytkownicy takich kont nie potrzebują dostępu do wiersza poleceń, a skoro nie będą się logować, to lepiej już przy zakładaniu konta zadbać o to, aby ktoś nie wykorzystał takiego konta w sposób niewłaściwy. Najprostszym sposobem utworzenia takiego konta jest podanie odpowiedniego parametru do polecenia `useradd`. W literaturze często spotyka się w tym miejscu następujący przykład :

```
[root@dreptak root]# useradd -s /sbin/nologin test3
[root@dreptak root]#
```

Polecenie to tworzy konto test3 ustawiając jako powłokę domyślną `/sbin/nologin`. I faktycznie próba logowania na takie konto kończy się niepowodzeniem :

```
[tuptus@dreptak tuptus]$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Linux pod opieką Tuptusia
Nieautoryzowany dostęp zabroniony
Wszelkie zdarzenia są logowane!!!
================================

login: test3
Password:
This account is currently not available.
Connection closed by foreign host.
[tuptus@dreptak tuptus]$
```

Rozwiązanie to ma jednak jedną poważną wadę. Skoro użytkownik nie może się zalogować, to w jaki sposób ma sobie zmienić hasło? Nie ma przecież dostępu do polecenia `passwd`. Ustawienie hasła przez administratora tak raz na zawsze to nie jest najlepszy pomysł. Dlatego w takich przypadkach proponuje zastosowanie nieco innego rozwiązania. Jako shell podstawiam `passwd`. Aby można było stosować takie rozwiązanie należy wcześniej "douczyć" system. Aby dana aplikacja mogła być wykorzystywana jako shell, to musi się znajdować na liście powłok w pliku `/etc/shells`. Dopisujemy zatem `/usr/sbin/passwd` do tego pliku i teraz możemy już zakładać odpowiednie konta :

```
[root@dreptak root]# useradd -s /usr/sbin/passwd test3
[root@dreptak root]#
```

Jak działa ten mechanizm? Otóż użytkownik chcąc zmienić hasło loguje się do systemu tak jakby miał konto pełne. Po dokonaniu autoryzacji powinien zostać uruchomiony shell, ale w tym wypadku jest to `passwd`. W związku z tym natychmiast pojawiają się komunikaty wymuszające zmianę hasła. Po dokonaniu takiej zmiany `passwd` kończy działanie tym samym zamykając sesję użytkownika, tak samo jak ma to miejsce przy wylogowaniu z pełnego konta. Zatem otrzymaliśmy konto bez dostępu do powłoki nie zabierając użytkownikowi możliwości samodzielnej zmiany hasła.

Uwaga. W podobny sposób możemy podstawić za shell dowolny program. System jednak nie dopuści do przyjęcia za shell skryptów. Musi to być plik binarny.

Tak utworzone konto ma jednak dostęp do wielu innych usług na serwerze takich jak choćby ftp. Zablokowaniem dostępu do poszczególnych serwerów zajmiemy się jednak przy omawianiu konkretnych usług.

5.4 Kasowanie kont

Temat kasowania kont jest tematem dość dyskusyjnym. Większość doświadczonych czarodziejów uważa, że raz założone konto powinno istnieć aż do "śmierci" systemu, a w przypadku rezygnacji użytkownika z usług danego serwera konto należy jedynie blokować. Jest w tym dużo racji. Istnieją jednak sytuacje kiedy potrzebujemy takiej funkcjonalności, więc przyglądniemy się temu zagadnieniu.

Poleceniem służącym do kasowania kont jest polecenie `userdel`. Wydanie tego polecenia z parametrem określającym nazwę konta do usunięcia powoduje jedynie skasowanie konta, ale na dysku pozostają pliki należące do konta. Lepszym rozwiązaniem jest dodanie parametru `-r`. Parametr ten informuje zaklęcie, że razem z kontem mają zostać skasowane pliki użytkownika. Należy jednak pamiętać, że nie są kasowane wszystkie pliki, a jedynie te znajdujące się w katalogu domowym. Ale w systemie istnieją jeszcze inne miejsca, w których mogą znajdować się pliki związane z kasowanym kontem. Mogą to być pliki witryny www, skrzynka pocztowa i pliki tymczasowe, ale również kilka innych mniej typowych przypadków. Do rozwiązania tego problemu posłuży nam znane już zaklęcie `find`.

```
[root@dreptak root]# id test3
uid=507(test3) gid=507(test3) grupy=507(test3)
[root@dreptak root]# userdel -r test3
[root@dreptak root]# find / -nouser 2>/dev/null
[root@dreptak root]#
```

W tym przypadku w systemie nie było żadnych plików bez właściciela. Test `nouser` wyszukuje

pliki, dla których id właściciela nie ma odpowiednika w `/etc/passwd`. Warto taki test wykonywać okresowo niezależnie od zarządzania kontami. Oczywiście możemy przeprowadzić bardziej selektywny test pytając o konkretne id i od razu usunąć je z systemu :

```
[root@dreptak root]# find / -uid 507 | xargs rm -f
[root@dreptak root]#
```

Po takich zabiegach mamy pewność, że zatarliśmy wszelkie ślady niechcianego konta. Wprawdzie w logach mogą się jeszcze pojawiać próby dotarcia do naszego systemu przez to konto, ale na to już nic nie poradzimy. Jeśli ktoś będzie próbował wysłać pocztę na konto test3, to nasz serwer ją odrzuci, ale w logach zapisze taką informację.

5.5 Grupy użytkowników

Omawiając zarządzanie kontami nie sposób pominąć tematu grup użytkowników. Zaklęcia związane z zakładaniem i kasowaniem grup są zbliżone do poleceń zarządzających kontami, więc nie będę się nimi bliżej zajmował, a czytelnika odsyłam do manuala systemowego. Polecenia te to `groupadd` oraz `groupdel`. Omówimy natomiast przykład zastosowania grup.

Załóżmy, że nasi użytkownicy test i test2 mają prowadzić wspólny projekt i nikt poza nimi nie ma mieć dostępu do zasobów projektu. Aby umożliwić realizację projektu musimy :

- utworzyć odpowiednią grupę, tutaj będzie to *projekt*
- dodać do grupy członków projektu
- utworzyć katalog na zasoby projektu
- ustawić odpowiednio prawa dostępu do katalogu
- ustawić środowisko kont należących do projektu

Poniżej przedstawię przebieg procesu uruchomienia projektu oraz omówię trudniejsze elementy, szczegółową analizę pozostawiam czytelnikowi.

```
[root@dreptak root]# cd /home
[root@dreptak home]# groupadd projekt
[root@dreptak home]# usermod -G projekt test
[root@dreptak home]# usermod -G projekt test2
[root@dreptak home]# mkdir projekt
[root@dreptak home]# chgrp projekt projekt
[root@dreptak home]# chmod 2770 projekt
[root@dreptak home]# ln -s /home/projekt /home/test2/projekt
[root@dreptak home]# ln -s /home/projekt /home/students/test/projekt
[root@dreptak home]# echo "umask 007" >> /home/test2/.bashrc
[root@dreptak home]# echo "umask 007" >> /home/students/test/.bashrc
[root@dreptak home]#
```

Wyjaśnienia wymaga polecenie `chgrp`. Polecenia to zmienia grupę pliku podanego jako parametr. Podobnym poleceniem jest polecenie `chown`, przy czym służy ono do zmiany właściciela pliku. Możemy jednak zmienić jednocześnie właściciela i grupę pliku podając je jako parametr polecenia `chown`, przy czym musimy je oddzielić kropką. Oba polecenia mają możliwość działania rekursywnego, czyli przechodzenia do podkatalogów. W tym celu należy podawać parametr `-R`.

Wyjaśnienia wymagają również dwie ostatnie linie. Konstrukcje te dopisują polecenie `umask` do odpowiednich plików konfigurujących środowisko użytkowników projektu. Należy jednak pamiętać, że użytkownicy mogli samodzielnie modyfikować pliki konfiguracyjne i wówczas nasza ingerencja może spowodować niepożądane efekty. Ale czym jest zaklęcie `umask`? Otóż polecenie

to ustawia maskę uprawnień do plików wykorzystywaną w chwili tworzenia nowych plików (choć nie tylko). Kiedy zachodzi potrzeba założenia nowego pliku lub katalogu system musi nadać jakieś wartości początkowe uprawnień. W tym celu od uprawnień "pełnych" odejmuje wartości ustawione poleceniem `umask` i wynik przyjmuje jako wartość początkową. W przypadku plików bity `x` nie są ustawiane nawet wtedy gdy wynik ustalenia uprawnień początkowych na to zezwala. W naszym przypadku nowo tworzone pliki i katalogi będą miały ustawione pełne prawa dostępu dla właściciela i grupy oraz całkowity brak możliwości dostępu dla innych użytkowników. Ustawienie bitu `sgid` dla katalogu daje nam pewność, że dla wszystkich plików i katalogów grupa będzie ustawiona na *projekt*, co umożliwi członkom grupy dostęp do nich.

Poleceniem, którego do tej pory nie używaliśmy jest `ln`. Zaklęcie to służy do tworzenia linków znanych w innych systemach jako skróty. Wprawdzie odpowiedniość linków i skrótów nie jest tak jednoznaczna, ale dla naszych potrzeb możemy przyjąć, że spełniają tę samą rolę. W naszym przypadku utworzone linki mają znaczenie raczej kosmetyczne. Tworzymy je w celu ułatwienia użytkownikom dotarcia do zasobów projektu bez konieczności przeszukiwania drzewa katalogów. I na tym zakończę omawianie tego problemu. Jestem pewien, że na pojawiające się pytania potrafisz samodzielnie znaleźć odpowiedzi korzystając z dokumentacji dostępnej w systemie.

Zarządzając kontami i grupami kont pamiętaj, że pod ich nazwami kryją się rzeczywiste osoby i Twoje postępowanie będzie rzutować na Twoje stosunki z nimi. Postępuj zawsze tak, jakbyś miał właśnie przed sobą rzeczywistego petenta proszącego o pomoc w sprawie, której nie rozumie. Zawsze staraj się takiemu mugolowi pomóc, nauczyć go, "wcisnąć' mu do głowy choć odrobinę wiedzy tajemnej. Będziesz miał mniej pracy, a Twój system będzie bezpieczniejszy.

Rozdział 6

Konfiguracja sieci

Jeszcze do niedawna komputery wykorzystywane były głównie do obliczeń i jako inteligentne maszyny do pisania. Obecnie sytuacja się znacznie zmieniła. Komputery stały się urządzeniami multimedialnymi, ale także oknem na świat. Łatwość dostępu do Internetu sprawiła, że komputer, w wielu wypadkach traktowany jest jako "rozwinięcie" technologiczne telefonu. Aby jednak wykorzystać go do takich celów, należy go włączyć do sieci i odpowiednio skonfigurować. W tej części podręcznika zajmiemy się właśnie przystosowaniem naszego systemu do pracy w sieci.

6.1 Adresy w sieci IP

Znajomość z siecią zaczniemy od poznania podstaw najpopularniejszej obecnie sieci — sieci IP. Pierwszym punktem będzie poznanie budowy sieci. Z wielu względów przyjęto, że protokół TCP/IP należy przedstawiać jako schemat warstwowy. I tak :

warstwa fizyczna - to część związana ściśle ze sprzętem i jego funkcjonowaniem

interfejs - to już warstwa związana z obsługą urządzeń fizycznych i udostępnienia ich systemowi operacyjnemu

warstwa IP - jest to warstwa zawierająca mechanizmy komunikacji pomiędzy systemami

warstwa TCP - to już warstwa transportowa umożliwiająca przesyłanie danych

warstwa aplikacji - a to warstwa najbliższa człowiekowi, tutaj znajdziemy aplikacje serwerowe oraz klientów korzystających z usług serwerów.

Nie będziemy się szczegółowo zagłębiać w specyfikę poszczególnych warstw. Zajmiemy się tylko tymi elementami, które mają istotne znaczenie dla użytkownika. Z warstwy fizycznej interesuje nas fizyczny adres karty sieciowej nazywany często MAC adresem. Adres ten składa się z 48 bitów i najczęściej przedstawiany jest w notacji szesnastkowej jako 6 grup ośmiobitowych. Przykładem takiego adresu może być taki zapis `02:60:8C:2E:9B:CA`. Adres Twojej karty możesz poznać wpisując polecenie `ifconfig` :

```
[tuptus@dreptak tuptus]$ /sbin/ifconfig -a
eth0  Link encap:Ethernet  HWaddr 00:50:04:67:CC:93
      inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
(...)
[tuptus@dreptak tuptus]$
```

Pole `HWaddr` to właśnie adres fizyczny karty sieciowej. Teoria mówi, że adresy te są niepowtarzalne i nie mogą być zmienione (są zapisane na stałe na karcie sieciowej). Nie jest to do końca prawdą, gdyż zdarza się, że firmy produkujące karty przydzielają identyczne adresy kartom kierowanym na różne rynki zbytu. Stąd adres fizyczny Twojej karty może być identyczny z adresem karty sprzedawanej na Tajwanie. Sytuacja taka nie powoduje konfliktów, dopóki nie sprowadzamy "na własną rękę" urządzeń z odległych stron świata. Niezmienność adresu też do końca nie jest prawdą. Rzeczywiście nie mamy możliwości zmiany danych zapisanych na karcie, ale istnieją możliwości programowe zmuszenia karty do przedstawiania się w sieci innym adresem, niż faktycznie zapisany w jej pamięci. Właściwość ta jest często wykorzystywana przez czarnoksiężników w celu podszycia się pod serwery lub stacje klienckie. Ale istnieją też przykłady całkiem legalnego wykorzystania maskowania MAC adresów np. w projektach High Avability Cluster. Dla naszych potrzeb przyjmujemy jednak, zgodnie z teorią, że MAC adres jest unikalny i niezmienny.

Kolejną warstwą wymagającą omówienia jest warstwa interfejsu. Przez interfejs rozumieć będziemy reprezentację urządzenia fizycznego w systemie operacyjnym. Interfejsów obsługiwanych przez system Linux jest wiele, jednak w praktyce spotkasz się z trzema z nich (inne spotykane są sporadycznie w zaawansowanych zastosowaniach). Kiedy wykonałeś polecenie `ifconfig` otrzymałeś informację co najmniej o jednym interfejsie — `lo`. Jest to tzw. loopback. W literaturze polskojęzycznej nazywany jest często interfejsem pętli zwrotnej. W dużym uproszczeniu jest to programowa imitacja interfejsu sieciowego wykorzystywana przez system do odwoływania się do samego siebie. Dwa pozostałe to `eth` i `ppp`. Za nazwą interfejsu następuje jego numer w systemie, przy czym numerowanie zaczyna się od 0. Interfejsy `eth` to urządzenia wykorzystujące sieć ethernet. Znajdziemy tu karty sieciowe i DSL. Interfejsy `ppp` to urządzenia wykorzystujące połączenia typu "point to point" i będą to różnego rodzaju modemy. Jeśli w Twoim komputerze zainstalowana jest jedna karta sieciowa ethernet i jeden modem, to po wydaniu polecenia `ifconfig` zobaczysz interfejsy `eth0` i `ppp0`.

Następną warstwą jest warstwa IP. Tutaj zatrzymamy się na dłużej. Warstwa IP odpowiedzialna jest za transport pakietów pomiędzy poszczególnymi interfejsami sieciowymi. Internet Protocol (IP) ukrywa przed systemem fizyczną strukturę sieci udostępniając mechanizmy umożliwiające dostęp do struktury logicznej. Dzięki takiemu podejściu wyższe warstwy mogą korzystać z sieci niezależnie od tego jakie urządzenia w tej sieci funkcjonują i jak fizycznie zbudowana jest struktura sieci. W celu stworzenia takiej wirtualnej sieci każdemu interfejsowi nadawany jest unikalny adres IP składający się z 32 bitów. Oczywiście to tylko teoria. Już w tej chwili w sieci funkcjonuje więcej komputerów niż jest dostępnych adresów IP. Jak to jest możliwe? Otóż unikalność adresów musi być zapewniona w ramach danej sieci logicznej, natomiast sieci logiczne mogą być maskowane. Temat maskowania adresów sieciowych zostanie omówiony w dalszej części podręcznika. W tej chwili przyjmujemy, że adres IP interfejsu jest unikalny.
Kolejnym elementem warstwy IP jest zapewnienie prawidłowego określenia trasy przesyłania pakietów pomiędzy interfejsami. Element ten nazywany jest routingiem. Każdy system przechowuje w pamięci tabelę routingu, którą możemy zobaczyć wydając polecenie `route`. Tabele tą system wykorzystuje na własne potrzeby, jednak można system tak skonfigurować, aby umiał przekazywać pakiety pochodzące z innych komputerów i kierowane również do innych komputerów. Do tego celu wymagane jest, aby komputer wyposażony był w co najmniej dwa fizyczne interfejsy sieciowe połączone do różnych sieci logicznych i był odpowiednio skonfigurowany. System udostępniający taką funkcjonalność nazywamy routerem. Warstwa IP odpowiedzialna jest również za przesyłanie sygnałów sterujących wykorzystaniem sieci. Do tego celu wykorzystywane są datagramy ICMP. Zapewne zetknąłeś się już z poleceniem `ping` wykorzystywanym do sprawdzenia czy "żyje" interfejs o podanym adresie. Polecenie to wykorzystuje właśnie pakiety ICMP. Przykład nie bardziej zaawansowany : kiedy próbujesz obejrzeć jakąś stronę www musisz podać adres serwera udostępniającego taką usługę. Jeśli pomylisz się i wpiszesz adres hosta, któ-

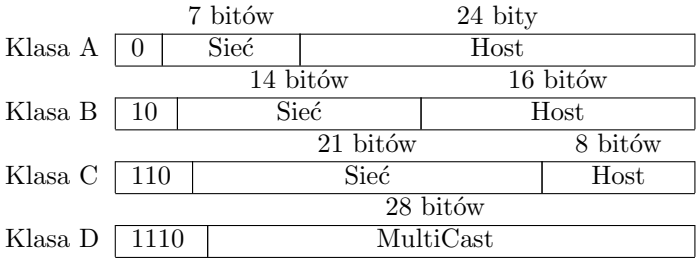
ry nie udostępnia stron www, to system zainstalowany na tym hoście odpowie pakietem ICMP informującym o braku żądanej usługi.

Zajmiemy się teraz najważniejszym tematem w tej części, czyli adresami IP interfejsów, przy czym omówimy jedynie standard IPv4 jako najbardziej rozpowszechniony (IPv6 jest jeszcze rzadko spotykany). Jak już wcześniej wspomniałem adres IP składa się z 32 bitów. Dla ułatwienia korzystania z niego przyjęło się przedstawiać adres w formie dziesiętnej jako cztery oktety (cztery grupy po osiem bitów). Maksymalną liczbę jaką możemy zapisać na 8 bitach jest liczba 255. Zatem adres 11000000101010000000000100000001 przedstawiamy jako 192.168.1.1

binarnie	11000000	10101000	00000001	00000001
dziesiętnie	192	168	1	1

Przedstawiony adres tak naprawdę nie jest adresem komputera. Każdy adres IP składa się z dwóch części : adresu sieci i adresu hosta w ramach tej sieci.

Adresy IP podzielono na tzw. klasy adresowe. Poniższy schemat przedstawia podział przestrzeni adresowej IP w zależności od klasy :



Z powyższego schematu można wyciągnąć kilka ciekawych wniosków :

Klasa A w pierwszym oktecie może zawierać liczby 1 – 127, w klasie tej może być 128 (7 bitów) różnych sieci po 16 777 214 hostów

Klasa B w pierwszym oktecie może zawierać liczby 128 – 191, w klasie mogą być 16 384 (14 bitów) rożne sieci po 65 534 hosty

Klasa C w pierwszym oktecie może zawierać liczby 192 – 223, w klasie może być utworzone 2 097 152 (21 bitów) różne sieci po 254 hosty

Klasa D jest klasą specjalną wykorzystywaną w zastosowaniach multimedialnych i rzadko spotykaną. Istnieje jeszcze klasa E, ale jest ona zarezerwowana dla wojska, więc nie będziemy się nią zajmowali.

Zapewne zauważyłeś, że liczby hostów w poszczególnych sieciach są o dwa mniejsze niż by to wynikało z liczby bitów przeznaczonych na adres hosta. Nie jest to błąd. Otóż w każdej sieci istnieją dwa adresy specjalne. Pierwszy adres, ten o najniższej wartości, to adres samej sieci i wykorzystywany jest w procesie routingu. Drugi adres specjalny to tzw. adres rozgłoszeniowy (ang. broadcast) i zajmuje on najwyższą wartość dostępną w danej sieci. Zapytanie wysłane na adres rozgłoszeniowy dociera do wszystkich hostów w danej sieci. Ze względu na działania różnych czarnoksiężników większość administratorów wyłącza obsługę broadcastów, gdyż odpowiadanie na zapytania tego typu może prowadzić do zatkania sieci.

W ramach adresów wyróżniono kilka adresów do zastosowań specjalnych. Przykładem takiego adresu jest adres 127.0.0.1 nadawany interfejsowi `lo`. Zapamiętaj ten adres, żebyś w przyszłości nie popełniał błędu mugoli wołających na grupach dyskusyjnych o pomoc : “Atakuje mnie komputer o adresie 127.0.0.1. Jak mam go namierzyć?”. Również pewne adresy sieci mają specjalne

znaczenie. Otóż wydzielono trzy adresy sieci i przyjęto, że są to sieci prywatne, niewidoczne z Internetu. Sieci te nie są routowane bezpośrednio do sieci Internetu. Adresy te to : w klasie A — 10.x.x.x, w klasie B — 172.16.x.x i w klasie C — 192.168.x.x

Zanim przejdziemy do praktycznego wykorzystania zdobytej wiedzy musimy poznać jeszcze jedno zagadnienie. Zagadnieniem tym jest maska sieci oraz związany z tym temat wydzielania podsieci. Maska sieci to ciąg 32 bitów zaczynających się jedynkami i zakończony zerami. Początkowe jedynki oznaczają bity, które w adresie IP oznaczają adres sieci. Na tej podstawie system może określić w jaki sposób przekazywać pakiety. W ramach jednej sieci pakiety przekazywane są bezpośrednio pod wskazany adres, w przypadku adresu spoza sieci pakiety kierowane są do routera, a ten przekazuje je dalej. Skoro odpowiednie ustawienie maski określa sieć, to co z klasami? Klasy to pojęcie umowne i dla komputera nie ma większego znaczenia poza tym, że granicą adresu sieci jest oktet. Zauważ, że adres sieci dla klasy A zajmuje 8 bitów, dla klasy B 16 bitów i dla klasy C 24 bity. Stosowanie się ściśle do zakresów klas byłoby bardzo niewygodne i w znacznym stopniu ograniczało możliwości konfiguracji sieci. Załóżmy, że w naszej sieci mamy 500 komputerów, jakiej klasy użyć? Klasa C jest zbyt mała, a stosowanie klasy B byłoby zwykłym marnotrastwem. A jeśli mamy fizycznie wydzieloną sieć zawierającą tylko dwa urządzenia (np. abonent DSL)? Potrzebujemy 4 adresów (sieć, 2 interfejsy i broadcast), a co z resztą adresów z klasy np. C? Z tego względu postanowiono stosować podsieci uzyskując je dzięki stosowaniu masek sieci. Zerknijmy zatem na schemat :

binarnie	11000000	10101000	00000001	00000001
dziesiętnie	192	168	1	1
maska C	11111111	11111111	11111111	00000000
dziesiętnie	255	255	255	0
Adres sieci	192	168	1	0

Powyższy przykład przedstawia przykład dla naszej sieci LAN przyjmując, że wykorzystujemy prywatną klasę C. W tym wypadku maska sieci zawiera 24 jedynki informując system, że adresy o początkowych 24 bitach identycznych z zawartymi w adresie naszej karty należą do tej samej sieci i należy je traktować jako adresy lokalne.

Nieco inaczej wyglądałby powyższy schemat dla naszego interfejsu zewnętrznego. Na potrzeby naszego kursu przyjęliśmy, że jest to adres 10.1.1.2. Zobaczmy jak będzie wyglądał schemat dla tego interfejsu zakładając, że znajduje się on w podsieci zawierającej 4 adresy :

binarnie	000001010	00000001	00000001	00000010
dziesiętnie	10	1	1	2
maska C	11111111	11111111	11111111	11111100
dziesiętnie	255	255	255	252
Adres sieci	10	1	1	0

W tak skonstruowanej podsieci mamy do dyspozycji dwa adresy (10.1.1.1 i 10.1.1.2) dla interfejsów sieciowych oraz dwa adresy specjalne (sieć — 10.1.1.0, broadcast — 10.1.1.3). Zauważ, że przy tak ustawionej masce możemy ustalić od razu adresy w kolejnej podklasie : sieć — 10.1.1.4, interfejsy — 10.1.1.5 i 6 oraz broadcast — 10.1.1.7.

Z maskami sieci wiąże się jeszcze pojęcie bitowego wykładnika potęgi. Brzmi to nieco groźnie, ale jest wyjątkowo prostym pojęciem. Otóż bitowy wykładni potęgi dla maski sieci to liczba początkowych jedynek w masce sieci. Adres naszego interfejsu wewnętrznego możemy przedstawić jako 192.168.1.1/255.255.255.0, ale w wielu przypadkach akceptowany jest również zapis 192.168.1.1/24. Podobnie dla interfejsu zewnętrznego adres można przedstawić jako 10.1.1.1/255.255.255.252 lub 10.1.1.1/30.

A teraz zadanie. Administrator podał nam informację, że dla interfejsu sieciowego naszego komputera przydzielił adres 10.0.2.15/23. Podaj pozostałe parametry sieci, w której ma pracować nasz interfejs. Zacznijmy od maski sieci — 23 czyli ostatnie 9 bitów to bity adresu hosta więc w sieci może znajdować się 510 interfejsów, natomiast 23 początkowe bity to adres sieci. Wynika z tego, że czwarty oktet to w całości część hosta natomiast w drugim oktecie tylko ostatni bit może się zmieniać. Więc adresy sieci dla tak utworzonej maski będą następujące : 10.0.0.0, 10.0.2.0, 10.0.4.0 itd. I już mamy wszystkie informacje :

Adres sieci — 10.0.2.0

Broadcast — 10.0.3.255

Maska sieci — 255.255.254.0

Jeśli doszedłeś do takich samych wyników, to świetnie, jeśli jednak nie, to przeczytaj ten rozdział jeszcze raz.

6.2 Konfiguracja interfejsów

Z wielu względów nie będę tu opisywał konfiguracji sieci w środowisku graficznym. Trzeba by było uwzględnić wszystkie możliwe środowiska oraz wersje dystrybucji. Poza tym narzędzia dostępne w środowiskach graficznych są bardzo intuicyjne i z całą pewnością poradzisz sobie samodzielnie.

Zdarzają się jednak sytuacje, gdy z różnych względów nie możemy wykorzystać środowiska graficznego lub jego funkcjonalność nie wystarcza do rozwiązania problemu. W takim wypadku pozostaje nam konsola i ręczna edycja plików tekstowych. Tymi sytuacjami zajmiemy się dokładniej.

6.2.1 Karty sieciowe

Mamy dwa rodzaje kart sieciowych PCI i ISA. Z pierwszą grupą kart z reguły nie ma problemów, gdyż są automatycznie wykrywane przez instalator oraz aplikację `kudzu`. Kiedy karta taka zostanie znaleziona w procesie instalacji, instalator doda odpowiednie wpisy do pliku `/etc/modules.conf`, zapyta o parametry interfejsu sieciowego i dokona odpowiednich wpisów konfigurujących ten interfejs. Ogólne parametry konfiguracji sieci znajdują się w pliku `/etc/sysconfig/network` natomiast konfiguracja interfejsu w pliku `/etc/sysconfig/network-scripts/ifcfg-eth0` (to dla pierwszego interfejsu). Niestety często, szczególnie przy dwóch kartach, karty zostaną wykryte jednak instalator pozwoli na ustawienie parametrów tylko jednego interfejsu. Jeśli masz do czynienia z sytuacją, gdy nie możesz korzystać z drugiego interfejsu, to poszukiwanie powinniśmy zacząć od polecenia `ifconfig` :

```
[root@dreptak root]# ifconfig -a
eth0 Link encap:Ethernet  HWaddr 00:50:04:67:CC:93
     inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:33 errors:0 dropped:0 overruns:0 frame:0
     TX packets:85 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:3979 (3.8 Kb)  TX bytes:12821 (12.5 Kb)
     Interrupt:15 Base address:0xd000

eth1 Link encap:Ethernet  HWaddr 00:20:AF:BE:96:D7
     BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
     TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:0
     RX bytes:0 (0.0 Mb)  TX bytes:0 (0.0 Kb)
     Interrupt:10 Base address:0x300

lo   Link encap:Local Loopback
     inet addr:127.0.0.1  Mask:255.0.0.0
     UP LOOPBACK RUNNING  MTU:16436  Metric:1
     RX packets:2 errors:0 dropped:0 overruns:0 frame:0
     TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:0
     RX bytes:100 (100.0 b)  TX bytes:100 (100.0 b)

[root@dreptak root]#
```

Z powyższego wynika, że system rozpoznał dwie karty sieciowe, ale z jakiegoś względu nie "podniósł" drugiego interfejsu. Przechodzimy zatem do katalogu `/etc/sysconfig/network-scripts` i kopiujemy plik `ifcfg-eth0` na `ifcfg-eth1` i modyfikujemy jego zawartość do postaci odpowiadającej naszym wymaganiom :

```
DEVICE=eth1
BOOTPROTO=none
BROADCAST=10.1.1.3
IPADDR=10.1.1.2
NETMASK=255.255.255.252
NETWORK=10.1.1.0
GATEWAY=10.1.1.2
ONBOOT=yes
TYPE=Ethernet
```

Nazwy poszczególnych pozycji są wymowne, więc nie będę się nad nimi rozwodził. Jedynie `BOOTPROTO` wymaga wyjaśnienia. Pozycja ta opisuje sposób postępowania przy uruchamianiu interfejsu. W przypadku statycznego adresu IP wpisujemy `none`, gdyż nie potrzebujemy żadnego protokołu ustalającego adres. Mogą tu jednak pojawić się wpisy `dhcp` w przypadku ustalania konfiguracji sieci przy pomocy klienta DHCP lub `bootp` w przypadku startu całego systemu na podstawie danych pochodzących z sieci.
Teraz jeszcze restart sieci poleceniem "`service network restart`" i już mamy dwie karty sieciowe.

Niestety nie zawsze jest tak prosto. Karty ISA nie są przez system rozpoznawane i trzeba je dodać ręcznie. Instalację takiej karty rozpoczynamy od rozpoznania co to za karta : jej nazwa, producent oraz symbol chipa. Ja mam kartę EtherLink II firmy 3Com opartej na chipsecie 3c509B i na tej podstawie opiszę dalsze postępowanie. Pierwszą sprawą jest ustalenie jaki sterownik obsługuje naszą kartę. Ponieważ sterowniki to elementy jądra systemu, to informacji należy szukać w dokumentacji jądra, a dokładniej w katalogu `/usr/src/linux-2.4/Documentation/networking` (źródła jądra znajdują się w pakiecie kernel-sources). Dla mojej karty sprawa jest prosta, gdyż zaraz na początku katalogu znajduje się plik `3c509.txt`. Możliwe, że Ty będziesz musiał trochę poszukać, ale zapewniam Cię, że warto. W dokumentacji jest wiele informacji pomocnych przy konfigurowaniu karty. Z dokumentacji dla mojej karty wynika, że domyślnie uruchamiana jest na przerwaniu 10. To bardzo ważna informacja. Dzięki niej możemy w BIOSie maszyny zarezerwować to przerwanie dla urządzeń ISA tak, aby nie zajęły go urządzenia PCI. Możemy oczywiście próbować przydzielić naszej karcie ISA odpowiednie przerwanie na etapie uruchamiania kernela, ale doświadczenie wskazuje, że ustawienia w BIOSie są skuteczniejszą metodą. Kiedy już mamy skonfigurowany sprzęt możemy przystąpić do dodania karty do systemu. W tym celu otwieramy plik `/etc/modules.conf` i dodajemy dwie linie :

```
alias eth1 3c509
options 3c509 irq=10
```

Pierwsza linia "wiąże" sterownik `3c509` z interfejsem o nazwie `eth1`. Druga linia ustawia parametry urządzenia wykorzystywane w momencie startowania urządzenia. W tym wypadku jest to numer przerwania ustawiony na 10 (`irq=10`). Często należy podawać również adres oraz kilka innych parametrów. Szczegóły opisane są w dokumentacji sterownika. Teraz należy utworzyć odpowiedni plik konfiguracyjny tak, jak to opisałem wyżej oraz załadować sterownik urządzenia poleceniem `modprobe eth1`. I to właściwie wszystko. Czasami zdarza się, że trzeba jeszcze wykonać restart serwisu network, ale to już umiesz.

Opisując plik konfiguracyjny podałem sposób na konfigurację w przypadku posiadania statycznego adresu IP. Często się jednak zdarza, że nie mamy stałego adresu, lecz uzyskujemy go z serwisu DHCP. W takim przypadku plik `ifcfg-eth1` wyglądałby dużo prościej :

```
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=dhcp
```

Wszystkie informacje niezbędne do skonfigurowania sieci otrzymamy z serwera DHCP w czasie uruchamiania interfejsu eth1. Oczywiście w przypadku posiadania tylko jednego interfejsu należy stosować `eth0` jako nazwę interfejsu.

Uwaga. Do poprawnego funkcjonowania serwisu DHCP po stronie klienta niezbędna jest instalacja pakietu dhclient.

Skoro już mamy skonfigurowane i uruchomione interfejsy sieciowe należałoby je przetestować. Pierwsze testy dotyczyć będą funkcjonowania sieci jako takiej. Wykorzystamy do tego celu polecenie `ping`. Polecenie to wysyła datagramy ICMP o rozmiarze 56B pod wskazany adres. Zaczynamy od pingowania samego siebie :

```
[root@dreptak root]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.197 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.144 ms

--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.144/0.170/0.197/0.029 ms, pipe 2
[root@dreptak root]# ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=0 ttl=64 time=0.190 ms
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.139 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=0.149 ms

--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 0.139/0.159/0.190/0.024 ms, pipe 2
[root@dreptak root]#
```

Jak widać oba interfejsy odpowiadają na pingi, więc przechodzimy do następnych testów. Polecenie `route` pokaże nam konfigurację tablicy routingu naszej maszyny. O routingu powiemy dokładniej w dalszej części, tutaj tylko ogólnie :

```
[root@dreptak root]# route
Kernel IP routing table
```

```
Destination     Gateway   Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *         255.255.255.0   U     0      0        0 eth0
10.1.1.0        *         255.255.255.0   U     0      0        0 eth1
169.254.0.0     *         255.255.0.0     U     0      0        0 eth1
127.0.0.0       *         255.0.0.0       U     0      0        0 lo
default         10.1.1.2  0.0.0.0         UG    0      0        0 eth1
[root@dreptak root]#
```

Z powyższego wynika, że odwołania do naszej sieci LAN (192.168.1.0) mają być kierowane przez interfejs `eth0`, odwołania do sieci zewnętrznej (10.1.1.0) mają być kierowane przez interfejs `eth1`, natomiast pozostałe (default) mają być kierowane pod adres 10.1.1.2, czyli nasz gateway. Wszystko wygląda poprawnie, więc możemy przeprowadzić próby wysyłania pingów do interfejsów innych maszyn. Problemów raczej nie przewiduję na tym etapie, ale ...

Właściwie można na tym zakończyć omawianie konfiguracji interfejsów sieciowych, gdyby nie jeden drobiazg. A mianowicie szybkość sieci. Wprawdzie nie mówiliśmy jeszcze o konfiguracji serwera ftp, więc muszę Cię odesłać do odpowiedniego rozdziału opisującego ten temat. Kiedy już będziesz miał uruchomiony serwer ftp możesz przeprowadzić test szybkości działania sieci. W tym celu spróbuj przesłać duży plik (ok. 600MB) pomiędzy serwerem i komputerem położonym w tej samej sieci. Sprawdź z jaką szybkością zostały przesłane dane. Jeśli szybkość ta wynosi więcej niż 70% teoretycznej przepustowości, to wszystko jest w porządku. Jeśli jednak jest dużo mniejsza, to należy się zastanowić nad przyczynami.

Uwaga. Pamiętaj, że prędkość kart podawana jest w mega bitach, programy podają zaś prędkość transferu w kilo i mega bajtach. Jeśli dla kart 10Mb uzyskasz przesył na poziomie 1MB, to bardzo dobrze.

Jednym z częściej spotykanych powodów problemów z szybkością jest źle dobrana prędkość sąsiednich interfejsów. Ale wyjaśnijmy kolejno. Karty sieciowe mogą pracować z różnymi prędkościami. Starsze obsługiwały prędkość 10Mb/s, nowsze 100Mb/s. Czasami możesz się spotkać także z kartami 1000Mb/s oznaczanymi również jako Gigabit Ethernet. Ostatnio pojawiły się na rynku karty o prędkości 10Gb/s, ale to jeszcze rzadkość. Jakby tego było mało karty mogą pracować w dwu trybach : half-duplex i full-duplex. Aby ułatwić konfigurowanie kart w wielu wypadkach możliwe jest ustawienie auto-negocjacji prędkości i jest to tryb domyślny pracy karty. Przy wykorzystaniu tego trybu pracy karty sąsiadujące w sieci starają się wynegocjować szybkość i tryb pracy i ustawić się do uzyskania optymalnego przesyłu danych. Pomysł bardzo dobry, ale jak zawsze z bywa z automatami zdarzają się przypadki błędnego działania. Szczególnie uciążliwy jest błędnie wynegocjowany tryb pracy.

Do zarządzania kartami sieciowymi możemy wykorzystać dwa programy : `mii-tool` oraz `ethtool`. Ale tu lojalnie uprzedzam, że nie wszystkie sterowniki współpracują z tymi programami. Przykładem takiego sterownika jest dmfe.o obsługujący karty Davicom. Zawsze warto jednak sprawdzić :

```
[root@dreptak root]# mii-tool
eth0: no autonegotiation, 10baseT-HD, link ok
SIOCGMIIPHY on 'eth1' failed: Operation not supported
[root@dreptak root]# ethtool eth1
Settings for eth1:
        Supported ports: [ TP AUI ]
        Supported link modes:   10baseT/Half 10baseT/Full
        Supports auto-negotiation: No
        Advertised link modes:  Not reported
        Advertised auto-negotiation: No
        Speed: 10Mb/s
```

```
        Duplex: Half
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: off
        Current message level: 0x00000002 (2)
        Link detected: yes
[root@dreptak root]# ethtool eth0
Settings for eth0:
No data available
[root@dreptak root]#
```

Jak widzisz w moim przypadku karta eth0 współpracuje z `mii-tool`, ale nie z `ethtool`. W przypadku eth1 jest odwrotnie. Wykorzystując podane programy narzędziowe ustawiamy "na sztywno" prędkość karty i sprawdzamy, czy transfer się poprawił. Oczywiście ustawienie musi być identyczne po obu stronach łącza. Kiedy już wiesz jakie ustawienia potrzebujesz możesz wywołanie odpowiedniego programu umieścić w `/etc/rc.d/rc.local`. Dla mojej karty eth0 mogłoby to wyglądać następująco :

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
/sbin/mii-tool -F 10baseT-HD eth0
/etc/rc.d/rc.firewall
```

Więcej szczegółów o obu programach znajdziesz w manualu systemowym.

Jest jednak inny sposób na ustawienie szybkości karty i do tego skuteczny również w przypadku kart, które nie chcą współpracować z omawianymi programami. W dokumentacji sterownika możemy znaleźć informację o numerach trybów pracy sterownika, które możemy wykorzystać w opcjach ustawianych w pliku `/etc/modules.conf`. Dla karty eth1 (3c509) znajdujemy informację, że dla uzyskanie szybkości 10Mb half-duplex należy ustawić transiver na wartość 4. W związku z tym możemy zmodyfikować wpis w pliku `modules.conf` do postaci :

```
alias eth1 3c509
options 3c509 irq=10 xcvr=4
```

Teraz przeładowujemy sterownik poleceniami `rmmod eth1` i `modprobe eth1` i już mamy ustawienia jakie byśmy chcieli uzyskać. Pamiętaj jednak, że wykorzystanie takich możliwości wymaga zapoznania się z dokumentacją sterownika obsługującego Twoją kartę sieciową.

6.2.2 Modemy

6.2.3 Karty PCMCIA

6.3 Konfiguracja routingu

6.4 Resolver

Korzystanie z adresów IP jest dość uciążliwe. Ponadto jest wiele przypadków, gdy adres IP nie daje nam możliwości dostępu do wszystkich serwisów udostępnionych na danym serwerze (doskonałym przykładem są serwery wirtualne www). Dlatego potrzebny jest nam jakiś serwis

pozwalający mapować adresy IP na nazwy i odwrotnie. Zapewne w tym momencie pomyślałeś o DNSie. I słusznie, ale trzeba w jakiś sposób poinformować nasz system, gdzie tego serwisu szukać i jak z niego korzystać. Zaczniemy jednak od czegoś prostszego. Zanim jeszcze wymyślono DNS również starano się ułatwić sobie życie stosując nazwy symboliczne dla adresów IP. Nazwy te zapisywano w pliku `/etc/hosts`. W naszym systemie plik ten również istnieje i możemy z niego skorzystać, aby ułatwić sobie pracę. Dlaczego w przeglądarce wpisywać www.aurox.org skoro można wpisać aurox i wejść na witrynę? Jak to zrobić? Nic prostszego. Najpierw musimy ustalić adres IP serwera aurox, a następnie dokonać odpowiednich wpisów w pliku `hosts`. Adres tego serwera to 62.111.243.84.

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost
62.111.243.84   www.aurox.org   aurox
```

Zanim wykorzystamy ten wpis musimy jeszcze poinformować system, że ma z niego korzystać. Otwieramy zatem plik `/etc/host.conf` i sprawdzamy jego zawartość :

```
order hosts,bind
```

Ten wpis informuje resolver (program rozwiązujący nazwy) o wymaganej przez nas kolejności stosowania metod rozwiązywania nazw. Wpis `hosts` nakazuje korzystanie z pliku `hosts`, wpis `bind` nakazuje korzystanie z serwera DNS. Kolejność tych wpisów decyduje o kolejności stosowanych rozwiązań. Jeśli masz wpis analogiczny do powyższego, to możesz już korzystać z nazwy `aurox` jako adresu serwera www.aurox.org. W ten sposób można wpisać wiele adresów ułatwiając sobie pracę. Jednak przy większej ich liczbie zarządzanie takim plikiem byłoby uciążliwe. Ponadto wpisy obowiązują tylko w ramach danego systemu i inne komputery nie mogą z niego korzystać. I dlatego właśnie wymyślono DNS. Aby z niego korzystać należy nasz system nieco przygotować. W tym celu należy odpowiednio przygotować plik `/etc/resolv.conf`. W przypadku naszej sieci plik ten powinien wyglądać mniej więcej tak :

```
search domek.pl
nameserver 194.204.159.1
nameserver 194.204.152.34
```

Pierwsza linia informuje resolver, że w przypadku podania samej tylko nazwy hosta bez nazwy domeny ma go poszukiwać właśnie w domenie domek.pl. W tej chwili nie ma to większego znaczenia, gdyż nie mamy jeszcze własnego serwera DNS. Kolejne linie to adresy IP serwerów DNS, z których ma korzystać nasz resolver. W powyższym przypadku podałem serwery dns.tpsa.pl oraz dns2.tpsa.pl, Ty powinieneś podać serwery najbliższe. Prawdopodobnie Twój ISP ma własny serwer DNS i zapewne podał Ci jego adres, wykorzystaj go. W pliku `resolv.conf` możemy podać do trzech serwerów DNS. W przypadku korzystania z usług serwera DHCP plik ten jest automatycznie modyfikowany przez program `dhclient` w momencie uruchomienia tej usługi. I to już wszystko czego potrzebujemy, aby nasz system mógł korzystać z serwisów DNS udostępnianych przez innych użytkowników internetu. Budowę własnego serwera DNS omówimy w osobnym rozdziale.

Sprawdźmy zatem działanie naszego resolvera. Wykorzystamy do tego celu polecenie `host` :

```
[root@dreptak root]# host www.aurox.org
www.aurox.org has address 62.111.243.84
[root@dreptak root]# host 62.111.243.84
84.243.111.62.in-addr.arpa domain name pointer host-ip84-243.crowley.pl.
[root@dreptak root]#
```

I tu mamy niespodziankę. Okazuje się, że www.aurox.org to nazwa wirtualna dla adresu 62.111.243.84. Dokładniejsze informacje o korzystaniu z serwerów DNS znajdziesz w rozdzia-

le opisującym tę usługę.

6.5 Synchronizacja czasu

Problematyka synchronizacji czasu jest często ignorowana przez początkujących administratorów. Jest to jednak poważny błąd. Efektem braku synchronizacji czasu są, przykładowo, listy z przyszłości. W przypadku wysyłania takiego listu do systemu, w którym uruchomiono oprogramowanie antyspamowe maże nas spotkać niemiła niespodzianka. To tylko jeden przykład, ale problemów może być znacznie więcej. Wszystkie narzędzia, których będziemy potrzebowali znajdują się w pakiecie `ntp`, instalujemy go i przystępujemy do konfiguracji.

6.5.1 Synchronizacja okresowa

Ponieważ każdy komputer ma wbudowany układ zegarowy, to można przyjąć, że po jednorazowym ustawieniu czasu układ ten będzie podawał poprawny czas. Układ ten jednak działa z określoną dokładnością i po pewnym czasie zaczyna się spóźniać lub spieszyć i należy taki błąd skorygować. Do jednorazowego ustawienia czasu wykorzystujemy polecenie `ntpdate`. Polecenie to wymaga podania adresu serwera czasu, z którym ma dokonać synchronizacji. Ja do tego celu wykorzystuję publicznie dostępny serwer vega.cbk.poznan.pl (150.254.183.15), ale równie dobrze można korzystać z innych serwerów położonych bliżej w sieci.

```
[root@dreptak root]# ntpdate 150.254.183.15
13 Jun 10:30:07 ntpdate[2952]: adjust time server 150.254.183.15
offset 0.001001 sec
[root@dreptak root]#
```

Jeśli uzyskałeś analogiczny wynik to znaczy, że wszystko jest w porządku i możesz przystąpić do dalszych kroków. Jeśli jednak pojawiły się jakieś problemy, to prawdopodobnie komunikacja z serwerami czasu jest zablokowana na ogniomurku. Synchronizacja czasu odbywa się z wykorzystaniem protokołu UDP, przy czym wykorzystywany jest port 123 po obu stronach połączenia (na kliencie i serwerze). Jeśli jesteś administratorem ogniomurka, to przepuść opisany ruch, jeśli nie, to skontaktuj się z lokalnym guru i ustal sposób postępowania. Możliwe, że w Twojej sieci lokalnej funkcjonuje jakiś serwer czasu i nie ma potrzeby sięgania poza sieć lokalną.

Skoro mamy już dostęp do serwera czasu, to zajmijmy się zapewnieniem okresowej synchronizacji czasu na naszej stacji. Mamy do rozwiązania dwa zagadnienia :

1. synchronizacja czasu przy starcie systemu — w tym wypadku dopisujemy polecenie pokazane wyżej do pliku `/etc/rc.d/rc.local`, co zapewni nam wykonanie synchronizacji przy każdym starcie systemu, ale już po podniesieniu interfejsów sieciowych

2. okresowa synchronizacja — realizacja tego zadania wymagana jest tylko w przypadku systemów pracujących nieprzerwanie przez kilka dni lub w przypadkach stwierdzenia problemów ze stabilnością zegara systemowego; polecenie podane wyżej dodajemy do crona przy czym warto dodać parametr `-s` powodujący wygenerowanie odpowiednich komunikatów do sysloga; przykładowy wpis w cronie może wyglądać następująco

   ```
   00 */2 * * * /usr/sbin/ntpdate -s 150.254.183.15 >/dev/null 2>&1
   ```

6.5.2 Serwer czasu

Rozwiązanie opisane powyżej jest dobrym wyjściem dla stacji roboczej z bezpośrednim dostępem do internetu. W naszym jednak przypadku mamy do zbudowania sieć lokalną z serwerem

pełniącym rolę bramy internetowej. W takim wypadku lepszym rozwiązaniem jest odseparowanie stacji w LANie od sieci internet. W przypadku problematyki czasu polegać to będzie na synchronizacji czasu serwera ze źródłami czasu z sieci zewnętrznych i udostępnienie czasu hostom pracującym w sieci wewnętrznej. Do realizacji tego zadania wykorzystamy demona ntpd. Uruchomienie tego demona wymaga kilku zmian w plikach konfiguracyjnych. Pierwszym plikiem jest `/etc/ntp.conf`. Najbardziej podstawowa konfiguracja może wyglądać następująco :

```
server  127.127.1.0     # local clock
fudge   127.127.1.0 stratum 10
server  150.254.183.15
server 192.36.143.151
driftfile /var/lib/ntp/drift
broadcastdelay  0.008
keys            /etc/ntp/keys
```

Pierwszy serwer zgodnie z opisem to zegar w lokalnym systemie. Wykorzystywany będzie w sytuacjach awaryjnych, gdy z jakichś powodów nie będzie dostępu do zewnętrznych źródeł czasu. Wyjaśnienia wymaga fraza `stratum`. W dużym uproszczeniu możemy przyjąć, że jest to poziom zaufania do źródła czasu. Im mniejsza wartość `stratum`, tym większy poziom zaufania. Kolejne dwa serwery to zewnętrzne źródła czasu. Jak widzisz możemy podać kilka serwerów, a demon samodzielnie dojdzie do tego, który jest najpewniejszy i według niego będzie ustawiał zegar systemowy. Pliki podane w opcjach `driftfile` i `keys` już istnieją w systemie. Wykorzystywane są przez mechanizmy wewnętrzne demona i nie będziemy się nimi dokładniej zajmować. Pliki te muszą istnieć i demon musi mieć do nich swobodny dostęp. Bardziej przezornym adminom polecam zapoznanie się z opcją `restrict`. Komentarze zawarte w pliku `/etc.ntp.conf` wyjaśniają sposób wykorzystania tej opcji.

Demon ma jednak pewne ograniczenie. Otóż potrafi on dokonać synchronizacji tylko wówczas, gdy różnica czasu pomiędzy systemem lokalnym i źródłem zdalnym nie przekracza 1000 sekund. W przypadku większej różnicy demon kończy działanie. Aby zapewnić prawidłowy start demona trzeba wstępnie ustawić zegar systemowy. Funkcjonalność taką zapewnia polecenie `ntpdate` wywoływane ze skryptu `/etc/rc.d/init.d/ntpd`. Skrypt ten jednak wymaga, aby lista zewnętrznych serwerów zawarta była w pliku `/etc/ntp/step-tickers`. W najnowszych wersjach Auroksa plik ten już istnieje i wystarczy do niego wpisać listę adresów IP serwerów czasu. W starszych wersjach trzeba ten plik utworzyć ręcznie.

Kiedy już mamy przygotowane pliki konfiguracyjne możemy wystartować serwis poleceniem "`service ntpd start`". Po uruchomieniu serwisu powinniśmy odczekać kilka minut i sprawdzić jak się sprawuje nasz demon :

```
[root@dreptak root]# ntpq
ntpq> peer
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)        10 l    8   64  377    0.000    0.000   0.015
*vega.cbk.poznan .PPS.            1 u  827 1024  377   38.762   28.686  10.510
+Time2.Stupi.SE  .PPS.            1 u  883 1024  257   71.337    0.684   0.358
+info.cyf-kr.edu ntp1-rz.rrze.un  2 u  884 1024  377   42.997   17.456   2.779
ntpq> quit
[root@dreptak root]#
```

Polecenie `ntpq` uruchamia środowisko pozwalające na zarządzanie serwerem czasu. Listę poleceń można poznać wydając komendę `help`. Komenda `peer` wyświetla informację o źródłach czasu. Z powyższego przykładu widać, że za główne źródło został przyjęty serwer vega (gwiazdka przed nazwą). Serwery vega i Time2 bazują na generatorach czasu (zapis `.PPS.`), z reguły są to generatory cezowe. Dzięki temu `stratum` dla tych źródeł jest równe 1. Nieco innym przypadkiem jest

źródło info. Tutaj `stratum` wynosi 2, gdyż serwer ten korzysta z innego serwera (tutaj ntp1-rz) nie zaś z generatora czasu.

Teraz należy jeszcze zadbać o to, by demon startował automatycznie przy starcie systemu. Wykonamy to zadanie poleceniem "`chkconfig --level 35 ntpd on`", które zapewnia start serwisu w runlevelu 3 (konsola tekstowa) i 5 (start w środowisku graficznym).

Rozdział 7

Instalacja aplikacji

Jednym z najczęściej wykonywanych zadań administracyjnych jest zarządzanie aplikacjami. Dystrybucja Aurox Linux udostępnia nam zaawansowane narzędzia pozwalające na wykonanie tych zadań. Aplikacje dostarczane w ramach dystrybucji zostały zgromadzone w postaci pakietów rpm. Takie rozwiązanie pozwala na utrzymanie porządku w systemie i dostarcza wielu cennych informacji o tym, co mamy na dyskach. Podstawowym narzędziem administratora systemu jest polecenie `rpm` z pakietu o tej samej nazwie. W przypadku konieczności prac naprawczych lub przebudowy pakietów wykorzystujemy polecenie `rpmbuild` z pakietu `rpm-build`.

Zarządzanie dużą ilością pakietów przy pomocy wymienionych poleceń jest nieco uciążliwe, szczególnie dla początkujących adeptów sztuki magicznej. Dlatego twórcy dystrybucji dołączyli do niej kilka narzędzi ułatwiających zarządzanie pakietami. Najczęściej wykorzystywanym narzędziem jest `redhat-config-packages` działające w środowisku graficznym. Dostępne jest ono również z poziomu menu jako *"Dodaj aplikacje"*. To samo narzędzie znamy z instalatora systemu. Narzędzie jest wyjątkowo proste w obsłudze, ale ma też duże ograniczenia, szczególnie od czasu, gdy zabrakło w nim możliwości instalacji pojedynczych pakietów. Z tych względów w niniejszym podręczniku nie będziemy się nim zajmować.

Innym rozwiązaniem jest menadżer `apt`. Jest to aplikacja przejęta z dystrybucji Debian. Aurox jest chyba jedyną dystrybucją opartą na pakietach rpm, która włączyła tą aplikację do standardowego zestawu pakietów. Stosowanie `apta` zapewnia zachowanie spójności bazy pakietów, umożliwia korzystanie z wielu różnych repozytoriów również zdalnych. Wykorzystując `apta` możemy również wykonać aktualizację całego systemu. Ponieważ `apt` jest aplikacją konsolową, to dla ułatwienia korzystania z niego w dystrybucji znajdziemy aplikację o nazwie `synaptic`, która jest nakładką na `apta` działającą w środowisku graficznym. Ze względu na zalety tego rozwiązania zajmiemy się nim dokładniej.

Ostatnim elementem wymagającym omówienia jest instalacja aplikacji "ze źródeł". Ze względu na różnorodność stosowanych tutaj rozwiązań temat zostanie omówiony dość ogólnikowo wskazując bardziej metody postępowania, niż konkretne rozwiązania.

7.1 Menadżer pakietów RPM

Zacznijmy od wyjaśnienia czym są pakiety rpm. Nazwa rpm jest skrótem od *"Red Hat Package Manager"*. Tak, rpm został stworzony przez firmę Red Hat w celu ułatwienia zarządzania aplikacjami i bibliotekami instalowanymi w systemie. System pakietowania rpm ma ogromne możliwości, jednak tylko część z nich wykorzystywana jest w praktyce. Przykładem niewykorzystanej funkcjonalności jest możliwość konfiguracji aplikacji już na etapie jej instalacji znana z

dystrybucji Debian. Otóż pakiety rpm zawierają sekcję `POSTINSTALL`, w której umieszczane są skrypty wykonywane na zakończenie procesu instalacji. Wystarczy umieścić w tej sekcji odpowiedni skrypt, który zapytałby operatora o wartości parametrów niezbędnych do konfiguracji aplikacji, a następnie zapisał je w plikach konfiguracyjnych. Niestety nie spotkałem do tej pory pakietu, który zapewniałby taką funkcjonalność, co nie znaczy, że skrypty `POSTINSTALL` nie są wykorzystywane. Wręcz przeciwnie. Wiele pakietów zawiera takie skrypty wykorzystywane np. do ustawiania uprawnień w systemie plików wymaganych do funkcjonowania aplikacji.

Kolejnym problemem w systemie pakietów rpm są zależności. Ponieważ aplikacje bardzo często korzystają z funkcji znajdujących się w bibliotekach tworzonych przez innych developerów, to trzeba było w jakiś sposób powiązać instalację danego pakietu ze sprawdzeniem, czy w systemie dostępne są wymagane biblioteki. W tym celu w sekcji `REQUIRED` umieszczane są nazwy bibliotek wymaganych do poprawnego działania aplikacji z danego pakietu. W momencie instalacji pakietu `rpm` sprawdza te zależności i jeśli wymagania nie są spełnione odmawia zainstalowania pakietu podając przyczynę. Samo zastosowanie sekcji `REQUIRED` jest wspaniałym pomysłem, ale brak narzędzia umożliwiającego automatyczne zainstalowanie pakietów zawierających te brakujące biblioteki. Częściowo problem ten został rozwiązany przez aplikacje `redhat-config-packages` oraz `redhat-install-packages`. Ze względu na ograniczenia tych aplikacji, o których mówiłem wcześniej, nie będziemy się bliżej nimi zajmować. W dystrybucji Aurox jest o tyle dobrze, że `rpm` wymieniając brakujące biblioteki podaje również listę sugerowanych pakietów do instalacji. To jest pewne udogodnienie, jednak te sugerowane pakiety również mają wymagania i dostajemy kolejną listę niespełnionych wymagań i tak w kółko. Tak więc należy stwierdzić, że pomysł stosowania zależności jest dobry, ale praktyczne wykorzystanie nastręcza problemów.

Ze wszystkim oczywiście możemy sobie poradzić. Przejdźmy zatem do praktyki i zobaczmy jak to działa. Zaczniemy od zadań, które nie będą w żaden sposób ingerować w nasz działający system. Pierwszym zadaniem, któremu się bliżej przyglądniemy jest zadawanie pytań do menadżera `rpm`.

```
[tuptus@dreptak tuptus] rpm -qa | less
```

To proste polecenie wyświetli nam listę wszystkich pakietów aktualnie zainstalowanych w systemie. Istotnym jest tu parametr `-q` informujący `rpm` , że zadajemy pytanie. Parametr `-a` oznacza `all` czyli wszystko. Rozwiązanie to możemy wykorzystać do wyszukania zainstalowanego pakietu. Jeśli znamy fragment nazwy, to możemy to zadanie wykonać jeszcze prościej :

```
[tuptus@dreptak tuptus] rpm -qa | grep kernel
```

i otrzymujemy listę pakietów zawierających w nazwie *kernel*. Skoro już znamy pełne nazwy pakietów, to możemy dowiedzieć się więcej o danym pakiecie pytając o jego opis, listę plików czy pliki z dokumentacją.

```
[tuptus@dreptak tuptus] rpm -qi kernel
(...)
[tuptus@dreptak tuptus] rpm -ql kernel | less
(...)
[tuptus@dreptak tuptus] rpm -qd httpd | less
```

Zauważ, że do odpytywania `rpm-a` nie trzeba być rootem, można to zadanie wykonać z konta zwykłego użytkownika. Po zainstalowaniu nowego pakietu powinniśmy się zapoznać z jego dokumentacją i ostatnie z przedstawionych poleceń podaje nam informację, gdzie ta dokumentacja wylądowała. Powyższe pytania dotyczyły zainstalowanych pakietów. Jednak bardzo często mamy na dysku jakiś pakiet jeszcze nie zainstalowany i chcemy się czegoś o nim dowiedzieć. Nic prostszego, wystarczy do parametrów zapytania dodać parametr `-p` i zamiast nazwy pakietu podać nazwę pliku :

```
[tuptus@dreptak tuptus] cd /mnt/cdrom/Aurox/RPMS
[tuptus@dreptak RPMS] rpm -qip php-pgsql-4.3.6-9.4.aur.1.i386.rpm
(...)
[tuptus@dreptak RPMS] rpm -qdp php-pgsql-4.3.6-9.4.aur.1.i386.rpm | less
```

Skoro umiemy już odpytywać menadżera o pakiety zajmijmy się nieco poważniejszym zadaniem. Spróbujemy zainstalować jakiś pakiet. Zanim przystąpimy do faktycznej instalacji warto sprawdzić, czy mamy wszystko, co nam jest potrzebne do przeprowadzenia tej czynności. Musimy przelogować się na użytkownika root i przejść do katalogu z naszymi pakietami, a następnie wykonać test instalacji :

```
[tuptus@dreptak tuptus] su -
Password:
[root@dreptak root]# cd /usr/src/redhat/RPMS/i386
[root@dreptak i386]# rpm -ivh --test kmyfirewall-0.9.6.1-0.i386.rpm
Przygotowywanie... ########################################### [100%]
[root@dreptak i386]#
```

Wygląda na to, że pakiet kmyfirewall możemy zainstalować. Ale nie zawsze jest tak różowo :

```
[root@dreptak i386]# rpm -ivh --test enigma-0.81-1.i386.rpm
błąd: Niespełnione zależności:
        libSDL_image-1.2.so.0 jest wymagany przez enigma-0.81-1
        libSDL_mixer-1.2.so.0 jest wymagany przez enigma-0.81-1
    Sugerowane sposoby spełnienia:
        SDL_image-1.2.3-3.i386.rpm
        SDL_mixer-1.2.5-9.3.aur.3.i386.rpm
[root@dreptak i386]#
```

Wprawdzie pakietu enigma nie możemy zainstalować wprost, ale `rpm` podał nam rozwiązanie problemu. Aby pakiet enigma mógł być zainstalowany należy zainstalować pakiety SDL_image oraz SDL_mixer. Kopiujemy zatem wymagane pakiety do katalogu z pakietem enigma i wykonujemy ponowny test :

```
[root@dreptak i386]# rpm -ivh --test enigma-0.81-1.i386.rpm\
SDL_image-1.2.3-3.i386.rpm SDL_mixer-1.2.5-9.3.aur.3.i386.rpm

ostrzeżenie: SDL_image-1.2.3-3.i386.rpm: Sygnatura V3 DSA: NOKEY,
 key ID 4f2a6fd2
błąd: Niespełnione zależności:
        libsmpeg-0.4.so.0 jest wymagany przez SDL_mixer-1.2.5-9.3.aur.3
    Sugerowane sposoby spełnienia:
        smpeg-0.4.4-9.3.aur.4.i386.rpm
[root@dreptak i386]#
```

Tym razem okazuje się, że jeszcze brakuje pakietu smpeg. Należy go skopiować itd. itd. To jest właśnie ta niedogodność `rpm-a`, o której pisałem wcześniej. Kiedy już `rpm` przestanie narzekać na niespełnione zależności możemy wykonać właściwą instalację :

```
[root@dreptak i386]# rpm -ivh enigma-0.81-1.i386.rpm\
SDL_image-1.2.3-3.i386.rpm SDL_mixer-1.2.5-9.3.aur.3.i386.rpm\
smpeg-0.4.4-9.3.aur.4.i386.rpm

Przygotowywanie... ########################################### [100%]
(...)
[root@dreptak i386]#
```

I już możemy się cieszyć nową aplikacją w naszym systemie.

Ale w przypadku próby instalacji pakietów spoza dystrybucji możemy się natknąć na znacznie poważniejszy problem :

```
[root@dreptak i386]# rpm -ivh --test khttrack-0.9-1.i586.rpm
błąd: Niespełnione zależności:
        libcrypto.so.0 jest wymagany przez khttrack-0.9-1
        libGLcore.so.1 jest wymagany przez khttrack-0.9-1
        libhttrack.so jest wymagany przez khttrack-0.9-1
        liblcms.so.1 jest wymagany przez khttrack-0.9-1
        libssl.so.0 jest wymagany przez khttrack-0.9-1
[root@dreptak i386]#
```

To jeszcze nie jest tragedia, ale mamy poważny problem. Pakiet khttrack wymaga pewnych bibliotek, a nasz menadżer nie potrafi powiedzieć w jakich pakietach biblioteki te się znajdują. W takiej sytuacji doświadczeni czarodzieje wybierają się na http://www.google.pl lub http://rpmfind.net i sprawdzają w jakich pakietach znajdą potrzebne biblioteki. Mając już listę potrzebnych pakietów wyszukują te pakiety na krążkach z dystrybucją lub w internecie (dokładnie w takiej kolejności) i wykonują instalację wszystkich niezbędnych elementów.

Skoro już umiemy instalować pakiety, to przejdźmy do kolejnego zadania stanowiącego jedno z podstawowych obowiązków administratora systemu. Zadaniem tym jest aktualizacja pakietów. Zadanie to jest stosunkowo proste, jeśli posiadamy odpowiedni zestaw pakietów niezbędnych do tego zadania. Gromadzimy w jednym katalogu wszystkie pakiety, które chcemy zaktualizować i wykonujemy polecenie :

```
[root@dreptak root]# rpm -Uvh *
```

Teraz pozostaje nam już tylko czekać, aż zakończy się proces aktualizacji i po jego zakończeniu wykonanie restartu zaktualizowanych serwisów. Oczywiście mogą się tutaj pojawić komunikaty o problemach z zależnościami, gdyż nowe wersje aplikacji mogą zawierać nową funkcjonalność, a więc również nowe wymagania. Jednak z wcześniejszych informacji już wiemy jak sobie z nimi radzić.

Jak zapewne zauważyłeś nazwy pakietów kończą się łańcuchem znaków `i386.rpm`. Oznacza to, że pakiet taki został przygotowany dla procesorów zgodnych z procesorami Intel i386. Na krążkach instalacyjnych można jednak znaleźć również pakiety z innym zakończeniem nazwy np. `i686.rpm`. To również są pakiety przeznaczone dla procesorów Intel IA32 jednak zostały skompilowane w taki sposób, aby kod wynikowy był zoptymalizowany do pracy na procesorach Intel Pentium II i lepszych. W sieci można również spotkać pakiety dla innych procesorów : `ppc.rpm` — PowerPC i Risk, `alpha.rpm` — procesory DEC Alpha. Tych nie ruszaj, bo na komputerze PC z cała pewnością nie będą działać.

7.2 Źródłowe pakiety src.rpm

Przeglądając krążki i sieć zapewne natknąłeś się na pakiety zakończone łańcuchem `src.rpm`. Jak nazwa wskazuje są to pakiety zawierające źródła aplikacji. Bezpośrednia instalacja tych pakietów nie daje nam możliwości uruchomienia aplikacji (fakt zainstalowania pakietu nie zostanie nawet odnotowany w bazie zainstalowanych pakietów). Pakiet taki należy najpierw "przebudować". Zadanie takie możemy wykonać na kilka sposobów. Podam tu najprostszy z nich prowadzący do wygenerowania pakietu binarnego zoptymalizowanego dla naszego procesora. W tym celu, jako root, wydajemy polecenie `rpmbuild` (wymaga wcześniejszego zainstalowania pakietu rpm-build) z opcją informującą o typie procesora, dla którego ma zostać zoptymalizowany pakiet:

```
[root@dreptak SRPMS]# rpmbuild --rebuild --target=i686 kadu-0.3.7-1.src.rpm
(...)
Zapisano: /usr/src/redhat/RPMS/i686/kadu-0.3.7-1.i686.rpm
```

```
Zapisano: /usr/src/redhat/RPMS/i686/kadu-debuginfo-0.3.7-1.i686.rpm
(...)
[root@dreptak SRPMS]#
```

Oczywiście budowa pakietów ma inne wymagania niż instalacja, więc nie zdziw się, że zostaniesz poinformowany o braku jakichś pakietów. Prawdopodobnie będą to pakiety ze słówkiem `devel` w nazwie. Pakiety te zawierają informacje niezbędne do wykonania kompilacji źródeł do postaci binarnej. Zatem zainstaluj wymagane pakiety i wykonaj ponowne budowanie pakietu. Pod koniec potoku informacji jakie zobaczysz na ekranie pojawią się linie podobne do tych, które przedstawiłem powyżej. Informują one o miejscu na dysku, w którym możemy znaleźć wynik kompilacji. Jeśli informacji takiej nie znajdziemy, a będą tam jakieś komunikaty błędów, to mamy grubszy problem, którego rozwiązaniem zajmiemy się dokładniej przy omawianiu instalacji ze źródeł.

Teraz możemy przejść do podanego katalogu i wykonać instalację pakietu lub aktualizację, jeśli już jest zainstalowany taki pakiet w wersji "podstawowej". W przypadku upgrade'u może pojawić się problem. Otóż `rpm` nie rozróżnia pakietów różniących się jedynie napisem pomiędzy kropką i kończącym `rpm`. Dla menadżera pakietów pakiet już zainstalowany jest identyczny z tym, który chcemy zainstalować. Aby wymusić jednak dokonanie aktualizacji możemy wykorzystać opcję `--force` :

```
[root@dreptak root]# cd /usr/src/redhat/RPMS
[root@dreptak RPMS]# rpm -Uvh --force i686/kadu-0.3.7-1.i686.rpm
```

Uwaga. Opcje `--force` i `--nodeps` należy używać ze szczególną ostrożnością, gdyż może to prowadzić do zerwania zależności między pakietami i doprowadzić do zdestabilizowania całego systemu. Wykorzystuj te opcje tylko wtedy, gdy ponad wszelką wątpliwość wiesz, co robisz.

Skoro już wiemy jak pakiety instalować i aktualizować pozostaje jeszcze jedna sprawa, a mianowicie kasowanie pakietów. Zadanie to wykonujemy poleceniem `rpm` z parametrem `-e` :

```
[root@dreptak root]# rpm -e kadu
```

Zauważ, że podajemy tu nazwę pakietu, a nie nazwę pliku. W pewnych sytuacjach awaryjnych wymagane jest również podanie wersji pakietu, ale to rzadki przypadek. Oczywiście, tak jak przy instalacji, tak i tutaj zależności mogą dać znać o sobie uniemożliwiając wykonanie deinstalacji pakietu. Rozwiązaniem jest podanie do deinstalacji wszystkich wymienionych pakietów lub pozostawienie pakietu w systemie.

Oczywiście powyższy kurs to jedynie wierzchołek góry lodowej. Więcej szczegółów znajdziesz w `man rpm` oraz dokumentacji "Maximum RPM" zamieszczonej na krążkach dystrybucyjnych.

7.3 Zarządzanie pakietami rpm - apt

Brak odpowiedniego oprogramowania umożliwiającego skuteczne wykorzystanie mechanizmu zależności był od zawsze źródłem krytyki. W dystrybucjach takich jak Debian czy PLD stworzono odpowiednie mechanizmy zapewniające spójność systemu, natomiast w Red Hacie i dystrybucjach pochodnych ciągle brakowało odpowiedniego narzędzia. Wprawdzie prowadzone były próby stworzenia takich aplikacji (np. purp - aplikacja konsolowa o interfejsie zbliżonym do Midnight Commandera), ale nie zyskały większego uznania. Dopiero adaptacja aplikacji `apt` z Debiana rozwiązała większość problemów. Jako że rozwiązanie to cieszy się dużą popularnością i w Auroksie wychodzi na prowadzenie w dziedzinie zarządzania pakietami zajmiemy się nim dokładniej.

7.3.1 Apt z wiersza poleceń

Wszystko co potrzebujemy do sprawnego wykorzystania `apt-a` znajduje się w jednym pakiecie o nazwie apt-0.5.15.5-9.3.aur.1.i386.rpm (dystrybucja Aurox 9.3, u Ciebie może być nowsza wersja). Po zainstalowaniu tego pakietu mamy do dyspozycji kilka nowych zaklęć :

apt-get podstawowe zaklęcie w pakiecie, pozwala na instalowanie i usuwanie pakietów, naprawianie problemów ze spójnością systemu oraz zarządzanie buforem pakietów

apt-cache zaklęcie pozwalające na przeszukiwanie repozytorium pakietów, uzyskiwanie szczegółowych informacji o pakietach i ich zawartości

apt-cdrom zaklęcie wykorzystywane sporadycznie umożliwiające dodawanie krążków CD-ROM do repozytorium pakietów (krążki z pakietami rpm muszą być w odpowiedni sposób przygotowane na potrzeby apt-a)

Zanim jednak zaczniemy korzystać z tego dobrodziejstwa musimy zadbać o odpowiednią konfigurację. Pliki konfiguracyjne znajdują się w katalogu `/etc/apt`. Nas w tej chwili najbardziej interesuje plik `sources.list` zawierający informacje o typie repozytoriów, sposobie dostępu oraz adresach repozytoriów. Konfigurację zaczniemy od zapewnienia dostępu do repozytorium pakietów przeznaczonych dla Auroksa 9.3 znajdującego się na serwerze `tuwien`. Plik `sources.list` powinien wyglądać następująco :

```
# Aurox 9.3
rpm ftp://ftp.tuwien.ac.at/linux aurox/9.3/PL/apt base updates
```

Jak widzisz jest to bardzo prosty plik. Pierwsza linia jest komentarzem pomagającym zorientować się czego dotyczy dany fragment pliku. Linia druga (może ich być więcej) opisuje konkretne repozytorium i składa się z czterech pól :

1. rodzaj pakietów znajdujących się w repozytorium, w naszym przypadku będą to pakiety binarne `rpm`
2. protokół i adres serwera zawierającego pakiety, w naszym przypadku serwerem jest `ftp.tuwien.ac.at`, a dostęp do repozytorium uzyskamy wykorzystując protokół ftp; zamiast ftp może być również http oraz file (repozytorium na dysku lokalnym), a także cdrom
3. dystrybucja, a właściwie ścieżka do repozytorium znajdującym się pod podanym adresem
4. składniki repozytorium, z których będziemy korzystać; my korzystamy z pakietów podstawowych dystrybucji oraz ich aktualizacji, nie korzystamy z części `testing`

Właściwie możemy już zacząć korzystać z `apt-a`, ale zanim do tego przejdziemy warto jeszcze zerknąć na zawartość pliku `apt.conf` Plik ten zawiera informacje konfigurujące samą aplikację. W większości przypadków zawartość tego pliku możemy pozostawić bez zmian, ale są sytuacje, gdy należy je nieco zmodyfikować. Pierwsza sprawa to problemy z dostępem do serwera z repozytorium. Jeśli mamy problemy ze ściągnięciem pakietów, to warto się zastanowić nad umożliwieniem aplikacji wykonanie kilku prób pobrania pakietu. Wykonujemy to modyfikując pozycję `Retries` w sekcji `Acquire`. Domyślnie ustawiona jest na wartość 0, więc możemy spróbować nadać jej wyższą wartość. Kolejnym elementem, którym warto się zająć jest cache, czyli miejsce, w którym `apt` składuje ściągnięte pakiety. Tutaj mamy kilka możliwości ingerencji :

- `Clean-Install` w sekcji `APT` warto ustawić na “true”; takie ustawienie zapewni automatyczne czyszczenie bufora z zainstalowanych pakietów

- `Cache-Limit` w tej samej sekcji pozwala na ustawienie maksymalnego rozmiaru bufora przez podanie wartości w bajtach; ustawienie istotne w systemach o ograniczonym miejscu na dyskach
- podsekcja `Cache` w sekcji `Dir`; sekcji tej nie ma w pliku `apt.conf` znajdującym się w pakiecie apt, należy ją dodać jeśli chcemy zmienić położenie plików.

Aby dowiedzieć się jak ustawione są aktualne parametry pracy `apt-a` wykorzystujemy polecenie `apt-config`. Ponieważ w tej chwili interesują nas ustawienia dotyczące położenia plików, to informację nieco odfiltrujemy :

```
[tuptus@dreptak tuptus]$ apt-config dump | grep Dir
Dir "/";
Dir::State "var/state/apt/";
Dir::State::lists "lists/";
Dir::State::cdroms "cdroms.list";
Dir::State::prefetch "prefetch";
Dir::Cache "var/cache/apt/";
Dir::Cache::archives "archives/";
Dir::Cache::srcpkgcache "srcpkgcache.bin";
Dir::Cache::pkgcache "pkgcache.bin";
Dir::Etc "etc/apt/";
Dir::Etc::sourcelist "sources.list";
Dir::Etc::sourceparts "sources.list.d";
Dir::Etc::vendorlist "vendors.list";
Dir::Etc::vendorparts "vendors.list.d";
Dir::Etc::main "apt.conf";
Dir::Etc::parts "apt.conf.d";
Dir::Etc::preferences "preferences";
Dir::Etc::rpmpriorities "rpmpriorities";
Dir::Etc::translatelist "translate.list";
Dir::Etc::translateparts "translate.list.d";
Dir::Bin "";
Dir::Bin::methods "/usr/lib/apt/methods";
Dir::Bin::rpm "/bin/rpm";
Dir::Bin::scripts "/usr/lib/apt/scripts";
Dir::Locale "/usr/share/locale";
[tuptus@dreptak tuptus]$
```

Powiedzmy, że chcemy zmienić położenie bufora przesuwając go do katalogu `/tmp/apt`. W tym celu do pliku `apt.conf` należy dodać sekcję `Dir` w następującej postaci :

```
Dir "/"
{
  Cache "tmp/apt/";
};
```

Oczywiście, przed użyciem `apt-a`, należy jeszcze przenieść katalog `apt` z katalogu `/var/cache` do `/tmp`.

Podane powyżej informacje to jedynie wskazanie pewnych możliwości i nie musisz ich wykorzystywać w Twoim systemie. Jednak w przypadku problemów warto sięgnąć po tego typu środki zaradcze.

Począwszy od Auroksa w wersji 9.4 do naszej listy repozytoriów możemy dodawać krążki CD z dystrybucją. Ponieważ jest to raczej standardowa operacja, zadbano o to, żeby ułatwić administratorowi pracę. Nie musimy ręcznie dokonywać żadnych zmian w pliku `sources.list` Wszystkie potrzebne wpisy wykona za nas zaklęcie `apt-cdrom add`. Wystarczy postępować zgodnie z poleceniami pojawiającymi się na ekranie.

Skoro mamy już skonfigurowane środowisko możemy przystąpić do właściwej pracy. Pierwszym zadaniem, które musimy wykonać jest założenie bazy danych dostępnych pakietów.

Wykorzystamy do tego celu polecenie `apt-get update`. Opcja `update` powoduje połączenie się naszego systemu ze wszystkimi repozytoriami pakietów wymienionymi w pliku `sources.list` i pobranie listy dostępnych pakietów oraz informacji o nich. Dopiero teraz możemy wykorzystywać polecenia `apt-get` z opcjami `install` i `upgrade` oraz `apt-cache`.

Uwaga. Ponieważ zawartość repozytorium może ulegać zmianom (część updates zmienia się z cała pewnością) polecenie `apt-get update` należy wykonywać przed każdą instalacją czy aktualizacją pakietów. W przypadku serwera dobrym pomysłem jest dodanie tego zaklęcia do listy zadań wykonywanych w ramach zadania `cron.daily`

Skoro mamy już wszystkie elementy układanki przejdźmy do praktycznego przykładu wykorzystania tego narzędzia. Załóżmy, że chcemy skompilować pakiet `src.rpm` pochodzący spoza dystrybucji. W trakcie kompilacji pojawił się błąd "/usr/bin/ld ... brak -lcom_err". Komunikat tego typu informuje nas, że linker nie mógł znaleźć biblioteki, w tym wypadku chodzi o plik `libcom_err.so`. Zapytajmy zatem `apt-a`, co wie na ten temat :

```
[tuptus@dreptak Aurox-PT]$ apt-cache search libcom_err
e2fsprogs - Utilities for managing the second extended (ext2) filesystem.
[tuptus@dreptak Aurox-PT]$
```

A więc coś wie na ten temat. My natomiast wiemy, że do przeprowadzenia kompilacji potrzebne są nie tylko same biblioteki, ale również pliki nagłówkowe itp. znajdujące się z reguły w pakietach `*-devel`. Możesz zresztą sprawdzić, że powyższy pakiet jest juz zainstalowany, ale nie zawiera poszukiwanego pliku. Pytamy zatem `apt-a` dalej :

```
[tuptus@dreptak Aurox-PT]$ apt-cache search e2fsprogs
e2fsprogs - Utilities for managing the second extended (ext2) filesystem.
e2fsprogs-devel - Ext2 filesystem-specific static libraries and headers.
[tuptus@dreptak Aurox-PT]$
```

Jak widać faktycznie odpowiedni pakiet jest w repozytorium. Należy zatem zainstalować ten pakiet :

```
[root@dreptak root]# apt-get install e2fsprogs-devel
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  e2fsprogs-devel
0 upgraded, 1 newly installed, 0 removed and 1 not upgraded.
Need to get 0B/142kB of archives.
After unpacking 383kB of additional disk space will be used.
Committing changes...
Preparing...                ########################################### [100%]
   1:e2fsprogs-devel        ########################################### [100%]
Done.
[root@dreptak root]#
```

W moim systemie już są zainstalowane pakiety wymagane przez `e2fsprogs-devel`. Jeśli brakowałoby jakiegoś pakietu, to `apt` poinformowałby nas o tym i zaproponował jego doinstalowanie. Jeśli korzystasz z repozytorium `cdrom`, to na ekranie zobaczysz jeszcze kilka dodatkowych linii m.in. prośbę o umieszczenie odpowiedniego krążka w napędzie. Teraz możemy przystąpić do kompilacji naszego źródła i po zakończeniu będziemy mogli go zainstalować poleceniem `rpm -i`.

Więcej szczegółowych informacji znajdziesz w manualu systemowym. Dostępne są podręczniki do poleceń `apt-get` i `apt-cache`, a także manuale opisujące pliki konfiguracyjne `sources.list` i `apt.conf`. W katalogu `/usr/share/doc/apt` znajdują się przykładowe pliki

konfiguracyjne z komentarzami umożliwiającymi zrozumienie poszczególnych opcji. Polecam zapoznanie się z powyższą dokumentacją.

7.3.2 Aktualizacja systemu

Do tej pory wykorzystaliśmy jedynie część możliwości `apt-a`, a mianowicie instalację pakietu wraz z pakietami wymaganymi przez ten pakiet. Ale `apt` ma znacznie większe możliwości. Wykorzystując tego menadżera możemy w prosty sposób zadbać o to, aby w naszym systemie zainstalowane były zawsze najświeższe pakiety dostępne dla dystrybucji. Tak jak do instalacji pakietów wykorzystaliśmy polecenie `apt-get install`, to do aktualizacji pojedynczego pakietu możemy wykorzystać polecenie `apt-get upgrade` lub polecenie `apt-get dist-upgrade`, jeśli chcemy aktualizować cały system. Zaczniemy od sprawdzenia co mamy do zaktualizowania :

```
[root@dreptak root]# apt-get --simulate dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
0 upgraded, 0 newly installed, 0 removed and 0 not upgraded.
[root@dreptak root]#
```

Akurat mój system jest zaktualizowany, ale u Ciebie może się pojawić lista pakietów do aktualizacji. To była tylko symulacja dla stwierdzenia, czy jest coś do roboty. Jeśli pojawiła się lista pakietów, to trzeba wykonać faktyczną aktualizację. Ponieważ z pobieraniem pakietów z odległego serwera może być różnie proponuję podzielić zadanie na dwa etapy. Najpierw ściągniemy pakiety do bufora, a po zakończeniu sukcesem tego kroku wykonamy faktyczną aktualizację. Zatem do dzieła :

```
[root@dreptak root]# apt-get --download-only dist-upgrade
Reading Package Lists... Done
(...)
[root@dreptak root]# apt-get dist-upgrade
(...)
[root@dreptak root]
```

W ten sposób zaktualizowałeś system.

Z wykorzystaniem polecania `dist-upgrade` wiąże się jednak pewne niebezpieczeństwo. Otóż w przypadku tego polecenia `apt` uruchamia system rozwiązywania konfliktów. System ten analizuje zależności pomiędzy pakietami istniejącymi w systemie i tymi, które mają być zainstalowane. Na podstawie tej analizy usuwa z systemu pakiety, które uważa za mało istotne, a stwarzają problemy z zależnościami. W związku z tym, po wykonaniu aktualizacji dystrybucji z systemu mogą zniknąć pewne aplikacje. Oczywiście `apt` informuje o tym, które pakiety chce usunąć. Warto zatem wykonać symulację upgrade'u i wyniki zapisać do pliku, aby po dokonaniu aktualizacji można było przywrócić dawną funkcjonalność przez doinstalowanie pojedynczych pakietów.

7.3.3 Synaptic — apt w środowisku graficznym

7.3.4 Prywatne repozytorium

Korzystanie z repozytorium na zdalnym serwerze oraz pakietów na krążkach jest bardzo dobrym rozwiązaniem dla stacji roboczej. Jednak w przypadku zarządzania większą liczbą komputerów rozwiązanie takie jest mało efektywne. Wyobraź sobie, że jesteś administratorem pracowni komputerowej i musisz zainstalować jakiś pakiet na wszystkich komputerach. Musisz podejść do każdego komputera z krążkami i wykonać instalację. Również w przypadku upgrade'u warto by było mieć repozytorium pakietów już we własnej sieci, a nie ciągnąć pakiety z internetu na każdą

stację osobno, to niepotrzebne obciążanie łącza i tak z reguły zapchanego. Kolejnym problemem są pakiety, które nie weszły do dystrybucji, a mogą być wykorzystane w naszym systemie. Może masz pakiety, które sam zbudowałeś i przetestowałeś. Nie masz możliwości dodania tych pakietów do publicznych repozytoriów, a chciałbyś mieć do nich dostęp tak samo, jak do pakietów dystrybucyjnych.

Rozwiązaniem tych problemów może być stworzenie lokalnego repozytorium. Jako przykład przygotujemy na serwerze repozytorium pakietów na potrzeby pracowni komputerowej w oparciu o serwer ftp. W tym momencie przyjmuję, że mamy już działający serwer ftp z dostępem anonimowym. Temat ten będzie dokładniej omówiony w części opisującej usługi serwerowe. W przypadku standardowej konfiguracji pliki udostępniane przez serwer ftp znajdują się w katalogu `/var/ftp`. W katalogu tym zakładamy przedstawioną poniżej strukturę katalogów :

```
[tuptus@dreptak ftp]$ tree
.
‘-- apt
    ‘-- 9.3
        |-- RPMS.contrib
        |-- RPMS.os
        ‘-- RPMS.updates

5 directories, 0 files
[tuptus@dreptak ftp]$
```

Do katalogu `RPMS.os` kopiujemy wszystkie pakiety znajdujące się na krążkach dystrybucyjnych, natomiast do `RPMS.contrib` pakiety, które pochodzą spoza dystrybucji. Kiedy już skompletujesz wszystkie pakiety musisz wygenerować odpowiednie pliki opisujące repozytorium. Pliki te będą pobierane przez polecenie `apt-get update`. Zadanie to wykona za nas zaklęcie `genbasedir` znajdujące się w pakiecie apt.

```
[root@dreptak ftp]# genbasedir --flat --bloat /var/ftp/apt/9.3
Creating base directory... done
Components: contrib os updates
Processing pkglists... contrib os updates [done]
Processing srclists... [done]
Creating component releases... contrib os updates [done]
Creating global release file... [done]
Appending MD5Sum... contrib os updates [done]
All your base are belong to us!!!
[root@dreptak ftp]#
```

W zasadzie to wszystko, co musimy zrobić. Warto jednak zaglądnąć do powstałego katalogu `base` i zmodyfikować wpisy w plikach `release*`.

Aby korzystać z tak utworzonego repozytorium trzeba jeszcze dokonać modyfikacji pliku `/etc/apt/sources.list` na wszystkich stacjach w naszej sieci tak, aby pobierały dane z naszego serwera :

```
[root@dreptak root]# cat /etc/apt/sources.list
# Aurox Linux 9.3 - lokalnie
rpm ftp://dreptak.domek.pl apt/9.3 os updates contrib
```

i oczywiście wykonać `apt-get update`.

Komponent `updates` to trochę większy problem. Pakiety do tego repozytorium pozyskać możemy jedynie z internetu i ich lista może się dość często zmieniać. Aby utrzymać w aktualności nasz komponent trzeba stworzyć mirror serwera w internecie udostępniającego pakiety aktualizacyjne. Do tego celu możemy wykorzystać aplikacje znajdujące się w dystrybucji takie jak `ftpcopy` lub `wget`. Można również wykorzystać aplikacje spoza dystrybucji specjalizowane do tego typu zadań. W tym przypadku mogę polecić aplikację mirror, której sam używam z

powodzeniem od dłuższego czasu. Aby jednak nie komplikować rozwiązania wykorzystamy narzędzia dostępne w dystrybucji. Stworzymy więc odpowiedni skrypt i będziemy go wywoływać przez crona. Nasz skrypt powinien pobierać nowe pakiety ze zdalnego serwera oraz regenerować informacje o repozytorium. Zatem jako root tworzymy w katalogu `/usr/local/bin` skrypt o nazwie `mirror.sh`.

```
#!/bin/sh

/usr/bin/wget -a /var/log/mirror.log -m -b -nH -nd\
-P /var/ftp/apt/9.3/RPMS.updates\
ftp://ftp.tuwien.ac.at/linux/aurox/9.3/PL/apt/RPMS.updates

/usr/bin/genbasedir --flat -bloat /var/ftp/apt/9.3 contrib
```

Skrypt należy przetestować uruchamiając go ręcznie. Jeśli działa poprawnie możemy jego wywołanie wpisać do crona :

```
30 0 * * * /usr/local/bin/mirror.sh >/dev/null 2>&1
```

Jak zapewne zauważyłeś wynikiem działania skryptu, oprócz ściągniętych pakietów, jest plik logu `/var/log/mirror.log`. Zastosowanie opcji `-a` zapewnia nam ciągłość informacji o działaniu naszego mirrora. Plik ten jednak będzie cały czas przyrastał, co stwarza zagrożenie dla stabilności systemu i utrudnia jego przeglądanie. Zatem wskazane jest zadbanie o rotację tego pliku tak, jak pozostałych logów w systemie. To zadanie jednak pozostawiam Tobie do samodzielnego wykonania.

7.4 Instalacja “ze źródeł”

Czytając różne materiały dotyczące linuksa zapewne spotkałeś się z określeniem *Open Source*. Dosłowne tłumaczenie oznacza otwarte źródła a bardziej zrozumiale chodzi o programy udostępniane w postaci kodów źródłowych do samodzielnej modyfikacji i kompilacji. I to jest właśnie tajemnica linuksa. Większość aplikacji przeznaczonych dla tego systemu operacyjnego dostępna jest za darmo w formie kodów źródłowych. Pakiety rpm to już skompilowane i odpowiednio przygotowane składniki dystrybucji. Ale oczywiście nie wszystkie programy zostają włączone do dystrybucji a możliwe, że jakieś aplikacje będą Ci potrzebne. Wspominałem wcześniej o pakietach src.rpm i przygotowaniu ich do instalacji. Jest to pewna forma instalacji “ze źródeł” ale w przypadku src.rpm wszystko dzieje się automatycznie i ktoś musiał o tą automatykę zadbać. Tutaj zajmiemy się czystym kodem.

Zacznijmy od tego co nam będzie potrzebne. Skoro mówimy o kompilacji to z całą pewnością będzie to kompilator. W dystrybucji mamy do dyspozycji kilka kompilatorów jednak najczęściej potrzebne nam będą dwa z nich:

- gcc - kompilator języka C
- g++ - kompilator języka C++ (znajduje się w pakiecie gcc-c++)

Oczywiście same kompilatory to za mało, potrzebne będą jeszcze odpowiednie biblioteki. Jeśli do instalacji kompilatorów wykorzystamy apta to te najistotniejsze biblioteki zostaną zainstalowane razem z kompilatorami. Ale są to tylko podstawowe biblioteki. Do poprawnej kompilacji programów potrzebne nam będą również inne biblioteki, w zależności od wymagań danej aplikacji, oraz pliki nagłówkowe tych bibliotek. Trudno w tej chwili powiedzieć, które biblioteki będą potrzebne więc w tym miejscu tylko ogólna porada: jeśli w trakcie budowania aplikacji pojawią

się komunikaty o braku jakiejś biblioteki to powinieneś zainstalować pakiet o nazwie odpowiedniej do nazwy brakującego pakietu z przyrostkiem *devel* np. pakiet *qt-devel* jest potrzebny do budowania aplikacji działających w środowisku KDE.

Przejdźmy zatem do samej procedury instalacji. Zaczynamy od ściągnięcia pakietu. Na ogół jest to plik o nazwie *nazwa-wersja.tar.gz* lub *nazwa-wersja.tag.bz2*. Plik taki to spakowane archiwum zawierające pliki z kodami źródłowymi aplikacji. Archiwum takie należy rozpakować:

```
[tuptus@dreptak tuptus]$ gunzip nazwa-wersja.tar.gz
[tuptus@dreptak tuptus]$ tar -xf nazwa-wersja.tar
```

W przypadku archiwum bz2 polecenia wyglądają odrobinę inaczej:

```
[tuptus@dreptak tuptus]$ bunzip2 nazwa-wersja.tar.bz2
[tuptus@dreptak tuptus]$ tar -xf nazwa-wersja.tar
```

W wyniku tych operacji otrzymamy w katalogu bieżącym podkatalog o nazwie *nazwa-wesja*. I to jest ta prostsza część procesu instalacji. Mamy już rozpakowane archiwum więc wchodzimy teraz do powstałego katalogu i szukamy dwóch plików: `README` oraz `INSTALL`. Pliki te (lub o podobnej nazwie) zawierają opis samej aplikacji oraz procesu instalacji. Zapoznanie się z opisem znajdującym się w tych plikach to podstawa powodzenia. Bardzo często znajdziemy tam opis wymagań co do wykorzystywanych bibliotek i ich wersji, możliwe, że będziemy musieli coś doinstalować. Wprawdzie opis procesu instalacji znajduje się w wymienionych plikach ale tutaj podam podstawowe zasady.

Pierwszym poleceniem, które powinniśmy wykonać jest wywołanie skryptu `configure` znajdującego się katalogu aplikacji. Skrypt ten sprawdza nasz system pod kątem architektury sprzętowej oraz zainstalowanych bibliotek i narzędzi programistycznych. Najbardziej podstawowe wywołanie tego skryptu wykonujemy przez wpisanie:

```
[tuptus@dreptak tuptus]$ cd nazwa-wersja
[tuptus@dreptak nazwa-wersja]$ ./configure
```

Przy tak wywołanym poleceniu instalacja aplikacji zostanie skierowana do katalogu `/usr/local` i jego podkatalogów. Jest to całkiem rozsądne rozwiązanie gdyż rozdzielamy aplikacje instalowane z pakietów od tych, które instalujemy ze źródeł samodzielnie kompilując. Jeśli jednak chcielibyśmy wykonać instalację tak jak robią to pakiety to powinniśmy o tym poinformować `configure` podając parametr `--prefix=/usr`. Należy jednak pamiętać, że w takim przypadku cała instalacja zostaje przeniesiona do wskazanego katalogu. Problemem może tu być katalog z plikami konfiguracyjnymi gdyż chcielibyśmy aby był to katalog `/etc` a w tej chwili będzie to `/usr/etc`. Oczywiście możemy to zmienić dodając kolejny parametr wywołania skryptu `configure`: `--sysconfdir=/etc`. Dokładnie taki sam problem mamy z katalogiem `/var` i rozwiązujemy go dodając parametr `--localstatedir=/var`. Zatem wywołanie konfiguracji ostatecznie wyglądać będzie tak:

```
[tuptus@dreptak nazwa-wersja]$ ./configure --prefix=/usr --sysconfig=/etc --local\
statedir=/var
(...)
[tuptus@dreptak nazwa-wersja]$
```

Podane parametry nie wyczerpują możliwości konfiguracyjnych. Oprócz tych podstawowych parametrów każda aplikacja może mieć własne parametry związane z wymaganiami odnośnie bibliotek. Parametry te mogą być odpowiedzialne za sposób kompilacji lub wskazywać katalogi, w których powinny być poszukiwane składniki. Więcej informacji o dostępnych parametrach konfiguracyjnych uzyskamy wywołując skrypt `configure` z parametrem `--help`:

```
[tuptus@dreptak nazwa-wersja]$ ./configure --help | less
(...)
```

```
[tuptus@dreptak nazwa-wersja]$
```

Ładnie to wygląda ale to właśnie na tym etapie pojawia się najwięcej problemów. Ponieważ nie mamy tutaj żadnego mechanizmu doinstalowującego brakujące składniki to problemy takie mogą wystąpić i musimy nauczyć się jak sobie z nimi poradzić. Zaczniemy od najbardziej podstawowego przypadku (mam nadzieję, że Ty nigdy nie zobaczysz takiego komunikatu):

```
[tuptus@dreptak nazwa-wersja]$ ./configure --prefix=/usr --sysconfig=/etc --local\
statedir=/var
(...)
checking for gcc... no
checking for cc... no
checking for cc... no
checking for cl... no
configure: error: no acceptable C compiler found in $PATH
```

Bardzo częsty błąd początkujących — konfigurator nie mógł znaleźć kompilatora, prawdopodobnie nie został zainstalowany pakiet *gcc*.

Nieco trudniejszy do rozszyfrowania jest następny przykład:

```
[tuptus@dreptak nazwa-wersja]$ ./configure --prefix=/usr --sysconfig=/etc --local\
statedir=/var
(...)
creating Makefile
(...)
/usr/bin/ld: cannot find -lX11
collect2: ld returned 1 exit status
[tuptus@dreptak nazwa-wersja]$
```

Jak widzisz konfigurator znalazł wszystkie narzędzia potrzebne do kompilacji i przystąpił do budowania pliku *Makefile* i tutaj pojawił się problem. Program `ld` nie znalazł biblioteki *X11*. Takiego pliku w systemie próżno szukać, ale `libX11.so` ... to możesz znaleźć. Skoro linker nie znalazł to pewnie też nie znajdziesz dlatego, że nie został zainstalowany pakiet *XFree86-devel*. Może się jednak zdarzyć, że pakiet devel jest zainstalowany a mimo to linker nie może znaleźć biblioteki. Przyczyną jest brak pliku o odpowiedniej nazwie. W naszym przykładzie pokazałem przykład pliku, który jest poszukiwany. W rzeczywistości plik taki nie istnieje ale istnieją inne pliki np. `libX11.so.6.2` a na potrzeby kompilacji tworzy się link symboliczny. Akurat w przypadku *libX11* link taki już istnieje ale pokażę jak to się robi. Zaczynamy od wyszukania w systemie potrzebnej biblioteki. Kiedy już znajdziemy taką bibliotekę to przechodzimy do katalogu, w którym się ona znajduje i robimy link

```
[tuptus@dreptak tuptus]$ find / -name "libX11*" 2>/dev/null
/usr/X11R6/lib/libX11.so.6.2
/usr/X11R6/lib/libX11.so.6
/usr/X11R6/lib/libX11.a
/usr/X11R6/lib/libX11.so
[tuptus@dreptak tuptus]$ su -
Password:
[root@dreptak root]# cd /usr/X11R6/lib
[root@dreptak lib]#  ln -s libX11.so.6.2 libX11.so
ln: ‘libX11.so’: Plik istnieje
[root@dreptak lib]# ldconfig
[root@dreptak lib]#
```

W tym przypadku pojawił się komunikat błędu bo link już istnieje ale skoro jeszcze czytasz ten rozdział to prawdopodobnie brakowało Ci takiego linku i komunikat podobny do powyższego nie pojawi się. Ostatnie polecenie służy do odświeżenia informacji o dostępnych bibliotekach i należy je wykonywać zawsze po wprowadzeniu zmian w tym zakresie.

Jeżeli dojdziesz już do ładu z konfiguratorem to przychodzi kolej na właściwą kompilację.

```
[tuptus@dreptak nazwa-wersja]$ make
(...)
[tuptus@dreptak nazwa-wersja]$
```

Wygląda to bardzo niegroźnie ale tutaj również mogą się pojawić błędy. Będą to komunikaty podobne do tych jakie przed chwilą omawialiśmy. Problemy wynikają z tego, że konfigurator nie sprawdził wszystkich zależności i problemy pojawią się dopiero w trakcie kompilacji. Postępowanie jest analogiczne do opisanego wcześniej więc nie będę się tutaj rozpisywał.

Ostatnim krokiem jest właściwa instalacja. Jak zapewne zauważyłeś, do tej pory pracowałem na koncie zwykłego użytkownika. Do instalacji musimy jednak przelogować się na konto roota gdyż tylko root może zapisywać pliki w katalogach z binariami. Wykonujemy zatem polecenie

```
[tuptus@dreptak nazwa-wersja]$ su -
Password:
[root@dreptak root]# cd ~tuptus/nazwa-wersja
[root@dreptak nazwa-wersja]# make install
(...)
[root@dreptak nazwa-wersja]# exit
[tuptus@dreptak nazwa-wersja]$
```

W tym kroku raczej nie zdarzają się problemy chyba, że np. zabraknie miejsca na dysku ale to już inna sprawa.

Pozostaje jeszcze jedna sprawa do wyjaśnienia. Odinstalowanie pakietu nie stanowi większego problemu ale jak odinstalować program instalowany ze źródeł? Większość autorów zadbała jednak o rozwiązanie tego problemu.

```
[tuptus@dreptak tuptus]$ su -
Password:
[root@dreptak root]# cd ~tuptus/nazwa-wersja
[root@dreptak nazwa-wersja]# make uninstall
(...)
[root@dreptak nazwa-wersja]# exit
[tuptus@dreptak tuptus]
```

I to by było na tyle. Jeśli ta metoda nie zadziała to musisz ręcznie kasować pliki zainstalowane wcześniej ale to dość mozolna praca i wymaga zapoznania się z zawartością pliku `Makefile`. Plik ten to opis działań, które ma wykonać program `make`. Analiza zapisów sekcji `install:` może nam pomóc w ustaleniu listy plików, które zostały zainstalowane.

7.5 Budujemy pakiet rpm

Znamy już metodę instalacji programów ze źródeł, znamy również wady tej metody. Wady te można wyeliminować budując własne pakiety rpm. Nie jest to taka trudna sprawa choć wymaga nieco pracy. Oczywiście potrzebne nam będą narzędzia developerskie (tak jak przy kompilacji źródeł) oraz pakiet *rpm-build*, o którym wspominałem przy okazji omawiania pakietów `src.rpm`. Jeżeli pakiety te już mamy zainstalowane możemy przystąpić do stworzenie środowiska potrzebnego do budowy pakietów. W katalogu `/usr/src/redhat` mamy juz potrzebną strukturę katalogów ale to miejsce zarezerwowane dla roota. W związku z tym tworzymy w swoim katalogu domowym odpowiedni podkatalog i tworzymy analogiczną strukturę:

```
[tuptus@dreptak tuptus]$ mkdir rpmbuid
[tuptus@dreptak tuptus]$ cd rpmbuid
[tuptus@dreptak rpmbuid]$ mkdir BUILD
[tuptus@dreptak rpmbuid]$ mkdir RPMS
```

```
[tuptus@dreptak rpmbuid]$ mkdir SOURCES
[tuptus@dreptak rpmbuid]$ mkdir SPECS
[tuptus@dreptak rpmbuid]$ mkdir SRPMS
[tuptus@dreptak rpmbuid]$
```

Kolejnym elementem środowiska jest plik zawierający makrodefinicje wykorzystywane przez system rpm. Tworzymy zatem ten pliki w katalogu domowym i wpisujemy odpowiednie definicje:

```
[tuptus@dreptak tuptus]$ touch .rpmmackros
[tuptus@dreptak tuptus]$ mcedit .rpmmacros
%_topdir /home/tuptus/rpmbuild
%_tmppath /tmp
%packager Wojciech Gabor <tuptus@motocyckle.org>
%vendor Tuptus

[tuptus@dreptak tuptus]$
```

Jak widzisz, nazwa makra zaczyna się od znaku `%`. Pierwsze dwa makra maja charakter porządkowy i nie wpływają na budowę samego pakietu a jedynie wskazują katalogi dla systemu rpm. Szczególnie istotna jest definicja `%_topdir` bo to ona wskazuje drzewo katalogów, z którego ma korzystać `rpmbuild`. Kolejne dwa to już elementy dodawane do tworzonych przez nas pakietów. Więcej makrodefinicje można znaleźć w plikach macros w katalogu `/usr/lib/rpm` oraz podkatalogach. Warto się zapoznać z tymi plikami, gdyż przy budowaniu naszych pakietów będziemy często korzystać z tych definicji.

Mamy już przygotowane środowisko i możemy przystąpić do budowania pakietu. Jako przykład wykorzystam odkrytą ostatnio aplikację mysqlcc (MySql Control Center). Zaczynamy od pobrania z sieci archiwum ze źródłami, w tym wypadku będzie to pakiet *mysqlcc-0.9.4-src.tar.gz*. Pakiet ten umieszczamy w katalogu `%_topdir/SOURCES`. Teraz przechodzimy do katalogu `%_topdir/SPECS` i tworzymy nasz plik spec np. poleceniem `touch`.

Uwaga. W omawianym pakiecie znajduje się plik mysqlcc.spec przygotowany dla dystrybucji Mandrake Linux. Możemy oczywiście wykorzystać ten plik i przystosować go do naszej dystrybucji.

Zawartość pliku spec odpowiedzialna jest za zawartość finalnego pakietu ale również za przebieg budowy pakietu. Tworzenie tego pliku zaczniemy od zdefiniowania danych początkowych.

```
Name:           mysqlcc
Version:        0.9.4
Release:        1.aur.T
License:        GPL
Group:          Databases
Summary:        MySQL Control Center
URL:            http://www.mysql.com/products/mysqlcc/
BuildRoot:      %{_tmppath}/%{name}-%{version}-buildroot
Source:         %{name}-%{version}-src.tar.gz

%description
mysqlcc is a platform independent graphical MySQL administration client.
It is based on Trolltech's Qt toolkit.

%description -l pl
mysqlcc jest niezależnym od platformy graficznym środowiskiem administratora
bazy MySQL.
Aplikacja bazuje na zestawie narzędzi Qt firmy Trolltech.
```

Pierwsze dwie pozycje są oczywiste i wpisujemy tutaj nazwę i wersję aplikacji. Pozycja `Release` to już wg. naszego uznania. Ja podałem `1.aur.T` bo jest to pierwsza wersja tego pakietu wykona-

na dla Auroksa przez Tuptusia. Kolejna pozycja to licencja. Zanim coś tutaj wpiszesz, zapoznaj się z licencją samej aplikacji. Z całą pewnością będzie w pakiecie. Pozycja `Group` jest w miarę dowolna. Zapewne zauważyłeś podczas wybierania pakietów w programie instalacyjnym czy też później doinstalowując jakiś pakiet, że pakiety są uporządkowane w grupy. Nie musisz jednak trzymać się tych grup, możesz stworzyć nowa i też nic się nie stanie. Pole `Summary` to skrócony opis zawartości pakietu — pełny opis umieszczamy w sekcji `%description`. Kolejne dwa pola to już poważniejsza sprawa. Pole `BuildRoot` definiuje katalog, do którego będzie wykonywana "instalacja". Oczywiście nie jest to instalacja rzeczywista a jedynie symulacja niezbędna do zbudowania właściwego pakietu. W dalszej części pliku spec będziemy się odwoływać do tego katalogu przez zmienną `RPM_BUILD_ROOT` Ostatnie pole zawiera nazwę pliku ze źródłami. Plik ten będzie poszukiwany w katalogu `%_topdir/SOURCES`. Dalej następuje sekcja opisu, o której już wspominałem. Zauważ, że ja stworzyłem dwie takie sekcje. Pierwsza jest wersją domyślną, druga będzie wykorzystana jeśli system jest skonfigurowany do obsługi języka polskiego.

Zbudowaliśmy podwaliny naszego pakietu a teraz zajmiemy się właściwą budową.

```
%prep
%setup -n %{name}-%{version}-src -q

%build

./configure
make

%install
rm -rf $RPM_BUILD_ROOT
%__mkdir_p $RPM_BUILD_ROOT%{_bindir}
%__mkdir_p $RPM_BUILD_ROOT%{_datadir}/%{name}/translations
%__install -m 755 mysqlcc $RPM_BUILD_ROOT%{_bindir}
%__install -m 644 {*.wav,syntax.txt} $RPM_BUILD_ROOT%{_datadir}/%{name}
%__install -m 644 translations/*.{qm,ts} $RPM_BUILD_ROOT%{_datadir}/%{name}/translations

%clean
rm -rf $RPM_BUILD_ROOT

%files
%defattr(-,root,root)
%doc Changelog.txt INSTALL.txt LICENSE.txt README.txt TODO.txt
%{_bindir}/mysqlcc
%{_datadir}/%{name}

%changelog
* Sun Dec 18 2004 Wojciech Gabor <tuptus@motocykle.org>
- initial version
```

Budowę rozpoczynamy od przygotowania źródeł do kompilacji. Akcję tą wykonujemy w sekcji `%prep`. Najprostszą metodą jest wywołanie makrodefinicji `%setup -q`. W naszym jednak przypadku metoda ta nie sprawdzi się gdyż `%setup` domyślnie ustawia zmienną `RPM_BUILD_DIR` na wartość `%_topdir/BUILD/%name-%version` tymczasem po rozpakowaniu naszego pakietu w katalogu `BUILD` powstaje katalog mysqlcc-0.9.4-src. Dlatego musimy w sposób jawny podać katalog z rozpakowanymi źródłami i dzięki temu zmienna `RPM_BUILD_DIR` otrzyma właściwą wartość.

Kolejnym krokiem jest budowa aplikacji, którą definiujemy w sekcji `%build`. Jak widzisz, w naszym przypadku, nie ma żadnej filozofii, polecenia dokładnie takie same jakie znasz z kompilacji źródeł.

Skoro już mamy zbudowaną aplikację to przyszła kolej na instalację, którą definiujemy w sekcji `%install`. W przypadku instalacji ze źródeł wykonywałeś polecenie *make install* i nie zastanawiałeś się zapewne jak to działa. Tutaj musimy ręcznie wykonać to co wykonuje ta

komenda. Zaczynamy od skasowania katalogu `$RPM_BUILD_ROOT`, który mógł pozostać po wcześniejszych próbach budowy pakietu. Następnie tworzymy potrzebne nam katalogi. Zastosowane tutaj makro `%__mkdir_p` to nic innego jak polecenie `mkdir -p`. Polecenie `mkdir` oczywiście znasz zauważ jednak w jaki sposób podajemy nazwy katalogów do założenia. O ile to możliwe, zawsze wykorzystujemy makra zdefiniowane w systemie rpm. Tutaj mamy trzy istotne wartości:

- $RPM_BUILD_ROOT — zmienna powstająca w trakcie budowania pakietu, zawiera nazwę katalogu stanowiącego początek drzewa katalogów instalacji
- %_bindir — makro zawierające ścieżkę do katalogu zawierającego pliki binarne, domyślnie jest to katalog `/usr/bin`
- %_datadir — makro zawierające ścieżkę do katalogu zawierającego katalogi z zasobami poszczególnych aplikacji, domyślnie jest to katalog `/usr/share`.

Mając już przygotowane odpowiednie katalogi możemy przystąpić do instalowania w nich odpowiednich plików. Wykonujemy to zadanie poleceniem *install*. Oczywiście staramy się wykorzystać makrodefinicje i zamiast polecenia wywołujemy makro `%__install`. Polecenie *install* tak na prawdę jest rozszerzoną wersją polecenia *cp*. Ogólna forma tego polecenia wygląda następująco:

```
install -m <uprawnienia> <co kopiować> <dokąd kopiować>
```

W wyniku działania tego polecenia zostaną skopiowane odpowiednie pliki do podanej lokalizacji i zostaną im nadane uprawnienia podane po parametrze `-m`. Dokładnie takie same działania wykonuje wywołanie *make install*. Aby dokładnie zrozumieć konstrukcję tego polecenia przyjmij zasadę, że znajdujesz się w katalogu z rozpakowanymi źródłami (czyli w `$RPM_BUILD_DIR`) i wykonujesz instalacje do katalogów poniżej `RPM_BUILD_ROOT` traktując ten katalog jako początek drzewa katalogów. Zatem, jeśli nasz plik wykonywalny *mysqlcc* ma docelowo wylądować w katalogu `/usr/bin` to, w tym momencie, musimy go instalować do `$RPM_BUILD_ROOT/usr/bin` lub stosując notację z makrami — `$RPM_BUILD_ROOT%{_bindir}`.

Po dokonaniu instalacji wypadałoby posprzątać o sobie. Do tego służy sekcja `%clean`. Właściwie nie ma tu czego tłumaczyć, w sekcji tej kasujemy niepotrzebne już katalogi.

Natomiast następna sekcja jest bardzo istotna. Sekcja `%files` opisuje sposób umieszczenia plików i katalogów w pakiecie i tym samym w docelowym systemie plików. Pierwsza definicja występująca w tej sekcji jest szalenie istotna więc zatrzymamy się przy niej na chwilkę. Zauważ, że do tej pory pracowałeś jako zwykły użytkownik. Skutek tego jest taki, że "zainstalowane" pliki wprawdzie mają ustawione uprawnienia ale jako zwykły użytkownik nie możesz zmieniać ani właściciela ani grupy dla plików i katalogów. Ten etap budowy pakietu możemy zrealizować dopiero w momencie pakowania plików do pakietu. Makro `%defattr` przyjmuje cztery parametry: uprawnienia dla plików, właściciel, grupa i uprawnienia dla katalogów. Ponieważ uprawnienia dla plików i katalogów już mamy ustawione to na pierwszym miejscu podajemy `-` a ostatni parametr pomijamy — dla systemu rpm jest to informacja, że nie ma zmieniać uprawnień. Ponieważ binarki i ich zasoby powinny być w systemie własnością roota to musimy taką informacje podać w tej definicji. Istotne jest to, że ustawienia w `%defattr` są ustawieniami domyślnymi dla wszystkich plików i katalogów w pakiecie. Jeśli dla jakiegoś pliku lub katalogu musimy mieć inne parametry niż te domyślne to na początku linii opisującej taki plik podajemy makro `%attr(<uprawnienia>,<user>,<grupa>)` i dla tego konkretnego pliku/katalogu system rpm ustawi uprawnienia inne niż te podane w `%defattr`.

A co się stanie jeśli nie podamy tej definicji? Efekty mogą być bardzo różne. Jeśli w systemie istnieje konto identyczne z nazwą konta "budowniczego" pakietu to nastąpi próba instalacji pakietu z uprawnieniami tego właśnie użytkownika. Jeśli tego konta nie ma to pojawi się komunikat, że konta nie znaleziono i system rpm przyjął konto root.

Dalej następuje lista plików i katalogów, które mają się znaleźć w pakiecie. Wyjątek stanowią pliki i katalogi podane jako parametry makra `%doc`. Niezależnie od tego gdzie zostały one "zainstalowane" w pakiecie trafią one do katalogu zdefiniowanego jako `%_defaultdocdir/%name-%version`. Dodatkowo pliki te będą wykazane jako dokumentacja pakietu, którą można zobaczyć wykonując polecenie `rpm -qd <nazwa pakietu>`

Ostatnia sekcja nie jest obowiązkowa ale często stosowana przez developerów. Sekcja `%changlog` zawiera inforamcje o zmianach wprowadzonych do kolejnych wersji pakietu.

Oczywiście powyższy opis nie wyczerpuje wszystkich możliwości systemu rpm. Wyobraź sobie sytuację, że aplikacja znajdująca się w budowanym pakiecie ma działać z uprawnieniami jakiegoś specjalnego użytkownika. Nie jest to wcale rzadkie zjawisko. Przecież aplikacje takie jak apache, mysql czy ntpd mają własne konta. Jak zatem zrealizować taką funkcjonalność przy pomocy pliku spec? Okazuje się, że jest to możliwe. W pliku spec możemy umieścić cztery sekcje, o których do tej pory nie mówiliśmy: `%pre, %post, %preun i %postun`. W sekcjach tych umieszczamy skrypty powłoki wykonywane przez system rpm odpowiednio: przed instalacją, po instalacji, przed odinstlowaniem pakietu i po odinstalowaniu pakietu. Tworząc jednak takie skrypty pamiętaj, że muszą to być skrypty samodzielne, nie mogą one oczekiwać na interakcję użytkownika gdyż system rpm nie dopuszcza do takiej interakcji.

Skoro mamy już przygotowany nasz plik spec możemy przystąpić do budowy pakietu. Podajemy zatem polecenie `rpmbuild -ba mysqlcc.spec` i czekamy z nadzieją, że nie pojawią się komunikaty błędów. Jeśli wszystko zrobiliśmy poprawnie to końcówka komunikatów wyglądać będzie podobnie do poniższych:

```
Szukanie niespakietowanych plików: /usr/lib/rpm/check-files /tmp/mysqlcc-0.9.4-buildroot
Zapisano: /home/tuptus/rpmbuild/RPMS/i386/mysqlcc-0.9.4-1.aur.i386.rpm
Wykonywanie(%clean): /bin/sh -e /tmp/rpm-tmp.99580
+ umask 022
+ cd /home/tuptus/rpmbuild/BUILD
+ cd mysqlcc-0.9.4-src
+ rm -rf /tmp/mysqlcc-0.9.4-buildroot
+ exit 0
```

Otrzymaliśmy informację gdzie znajdują się nasze pakiety ("Zapisano:...") i co najważniejsze: w ostatniej linii "exit 0" — operacja zakończyła się powodzeniem.

W zasadzie możemy już wykonać instalację nowego pakietu ale ...przezorny zawsze ubezpieczony. Radziłbym jednak przyglądnąć się pakietowi od środka i zobaczyć czy zawiera to co chcieliśmy aby zawierał. Proponuję wykorzystać do tego celu Midnight Commandera (polecenie `mc`). Przechodzimy zatem do katalogu z naszym pakietem, najeżdżamy kursorem na plik pakietu i `ENTER` . Zobaczymy listę następujących "plików": `/INFO, CONTENTS.cpio, HEADER, INSTALL, UPGRADE`. Tak naprawdę to nie są to żadne pliki tylko wyobrażenie tego co się znajduje w pakiecie. Nas w tej chwili interesują dwie pierwsze pozycje. Ustawiamy kursor na "katalog" `INFO` i wchodzimy do środka. Jeśli definiowaliśmy sekcje `%pre` lub `%post` to zobaczymy tutaj koleny katalog o nazwie `SCRIPTS` a w nim bedą nasze skrypty — warto się im jeszcze raz przygladnąć. Teraz wróćmy do `CONTENTS.cpio`. Wygląda jak zwykły plik o rozmiarze zero bajtów. Nic bardziej mylnego, ustawiamy kursor na tym pliku i `ENTER` . I co? Ciekawy widok? Co ważne, możemy się tutaj poruszać jak w normalnym drzewie katalogów, podglądać uprawnienia i co czasami się przydaje — możemy z tego zasobu kopiować pliki a nawet całe katalogi do naszego filesystemu. Ileż to razy się zdarza, że przez nieuwagę skasujemy jakiś istotny plik konfiguracyjny albo wprowadzimy tak wiele zmian, że już nie pamiętamy co zmienialiśmy a aplikacja nie chce ruszyć. Nie musimy przeinstalowywać całego pakietu, wystarczy, że wykorzystamy `mc` i wyciągniemy pojedynczy plik.

Jeśli już przeglądnąłeś pakiet i uważasz, że spełnia wymogi to możesz przystąpić do instalacji jak każdego innego pakietu. Oczywiście pakiet, który przed chwilą instalowaliśmy nie jest

pakietem skomplikowanym. Jeśli jednak będziesz budować bardziej wyrafinowane pakiety lub chciałbyś tak zbudowany pakiet wykorzystać na kilku innych systemach to warto się zastanowić nad budową lokalnego repozytorium apta i umieszczaniu w nim takich pakietów. Takie postępowanie z cała pewnością ułatwi instalację i utrzymanie zależności między pakietami.

Rozdział 8

Archiwizacja

Rozdział 9

Kopia bezpieczeństwa

Rozdział 10

Jądro systemu

10.1 Kernel z dystrybucji

10.2 Jądro "waniliowe"

10.3 Kompilacja jądra

10.4 Kernel 2.6.x w Auroksie 9.x

Rozdział 11

Zapora ogniowa

Jednym z elementów podwyższających poziom bezpieczeństwa naszego systemu jest dobrze skonfigurowany firewall. Zagadnienie to wymaga znajomości wielu informacji nie omawianych w tym podręczniku. W rozdziale tym postaram się omówić te elementy, które będą istotne w naszym przypadku, ale zachęcam do dokładniejszego zapoznania się z tematyką zapór ogniowych.

11.1 Zasada działania

Omawianie zasady działania zapory ogniowej zaczniemy od kilku informacji o podstawach działania sieci. Zapewne zdajesz sobie sprawę, że dane przesyłane przez sieć nie stanowią jednej całości, tylko przesyłane są mniejszymi porcjami. Porcje te nazywane są pakietami. Żeby system odbiorcy umiał odtworzyć przesyłaną informację, pakiety zawierają nie tylko dane, ale również pewne informacje kontrolne zawarte w nagłówku pakietu. W zależności od tego jaki to typ pakietu, nagłówki mogą zawierać nieco inne dane. Istotny jest tutaj stosowany protokół. W naszym przypadku zajmiemy się tylko trzema protokołami :

1. ICMP - jest to protokół pozwalający na przesyłanie tzw. datagramów kontrolnych; przykładem wykorzystania może być polecenie `ping` stosowane do sprawdzenia działania systemu zdalnego komputera,
2. UDP - protokół ten pozwala na przesyłanie danych pomiędzy hostami w sieci, nie daje jednak gwarancji, że wysłane dane dotrą do nadawcy, gdyż nie zawiera mechanizmów kontroli stanu
3. TCP - protokół najbardziej rozbudowany, ale jednocześnie najmniej wydajny, zawiera kontrolę kolejności pakietów, umożliwia fragmentację pakietów i co najważniejsze umożliwia kontrolę stanu połączenia.

Protokół ICMP nie posiada informacji o portach nadawcy i odbiorcy, jak to jest w przypadku dwóch pozostałych protokołów, ma natomiast zestaw zdefiniowanych typów komunikatów. Podstawowe typy komunikatów ICMP zostały przedstawione w tabeli 11.1 Tworząc reguły filtrowania tego protokołu musisz podawać typ komunikatu, jaki chcesz poddać filtrowaniu.

W przypadku protokołu UDP sytuacja przedstawia się nieco inaczej. Pakiety takie wysyłane są z jakiegoś portu i dostarczane do celu również na jakiś port. Filtrując pakiety tego protokołu podajesz te numery określając szczegółowo, jakich pakietów dotyczy filtrowanie. Należy jednak pamiętać, że pakiety UDP są całkowicie niezależne od siebie. Wygląda to tak, jakbyś wrzucał garść pocztówek do skrzynki pocztowej i nie wiesz ile z nich zostanie dostarczona do adresata. Nie

Typ	Opis
0	Odpowiedź echa (odpowiedź na `ping`)
3	Nieosiągalne miejsce przeznaczenia
5	Przekierowanie
8	Żądanie echa (generowany przez `ping`)
11	Upłynął czas życia pakietu (nie dotarł do celu)

Tabela 11.1: Komunikaty ICMP

wiesz również w jakiej kolejności. Dodatkowo, pakiety UDP mogą zostać zdublowane w procesie przesyłania. Z tego wynika problem filtrowania. Mianowicie nie można w prosty sposób określić, czy przychodzący pakiet jest nowym połączeniem, czy odpowiedzią na nasze zapytanie. Problem ten można obejść, ale o tym za moment.

Protokół TCP daje największe możliwości filtrowania. W nagłówku pakietu TCP zawarte są między innymi flagi wykorzystywane do określenia, czego dany pakiet dotyczy. Nas w tej chwili interesują tylko dwie flagi `ACT` i `SYN`. Kiedy komputer kliencki chce nawiązać połączenie z naszym serwerem wykonuje "powitanie" znane jako "triple greeting". Nazwa ta wywodzi się z faktu, że połączenie klienta z serwerem opiera się na trzech rodzajach pakietów, przy czym pierwsze dwa służą do zestawienia połączenia. Prześledźmy zatem te kroki :

1. klient wysyła do serwera pakiet z ustawioną flagą `SYN` (`ACT=0,SYN=1`)
2. serwer odpowiada wysyłając pakiet z ustawionymi oboma flagami (`ACT=1,SYN=1`)
3. dalsza komunikacja odbywa się z ustawioną jedynie flagą `ACT` (`ACT=1,SYN=0`)

Jak widzisz, śledząc ustawienie odpowiednich flag z łatwością możemy określić czy dany pakiet jest próbą otworzenia nowej sesji, czy też należy do już istniejącego połączenia. Jeśli dodamy do tego jeszcze wykorzystywanie określonych portów, to mamy całkiem pokaźne możliwości filtrowania.

Niestety te mechanizmy nie wystarczą do poprawnego filtrowania pakietów. Jako przykład omówimy protokół FTP bazujący na TCP. Dopóki połączenie między klientem i serwerem odbywa się w trybie "normalnym", to sprawa jest w miarę prosta. Klient nadaje z portów powyżej 1024, a serwer nasłuchuje na porcie 21 - polecenia, 20 - przesyłanie danych. Sytuacja jednak się komplikuje, gdy zastosowany zostanie tryb "passive". Komunikacja na porcie 21 pozostaje bez zmian, ale dane przesyłane są z portu powyżej 1024 na serwerze na port powyżej 1024 na kliencie. Nie wiemy na jakim porcie, więc jak to filtrować? I tu uwidacznia się potęga `iptables`. Otóż w kernelu zawarte są moduły śledzenia połączeń (*conntrack*), które umieją sobie radzić w takich sytuacjach oraz w wielu podobnych.

Skoro doszliśmy już do kernela, to powiedzmy sobie nieco o tym, jak kernel sobie radzi z pakietami (jeśli obsługuje filtrowanie pakietów). Zaczniemy od tego, że reguły filtrowania łączone są w łańcuchy. W systemie mamy trzy podstawowe łańcuchy :

- INPUT — łańcuch obsługujący pakiety przychodzące do naszego interfejsu
- OUTPUT — łańcuch obsługujący pakiety wychodzące z interfejsu sieciowego
- FORWARD — łańcuch obsługujący pakiety przechodzące przez nasz host (pomijane są wówczas łańcuchy INPUT i OUTPUT)

Łańcuchem FORWARD zajmiemy się przy okazji omawiania udostępniania dostępu do internetu komputerom z naszej sieci LAN, tutaj zajmiemy się natomiast tylko pierwszymi dwoma łańcuchami.

Kiedy do kernela dociera jakiś pakiet w pierwszej kolejności określane jest, czy jest to pakiet przychodzący, czy też wychodzący. W zależności od wyniku tego testu pakiet poddawany jest sprawdzeniu w ramach reguł należących do danego łańcucha. I teraz najważniejsze. Reguły przetwarzane są sekwencyjnie. Oznacza to, że w momencie, gdy nastąpi dopasowanie pakietu do danej reguły zostanie on przekazany do "celu" i kończy się przetwarzanie w ramach danego łańcucha. Dalsze reguły są pomijane. Wyjątkiem od tej zasady jest "cel" LOG powodujący przekazanie odpowiedniej informacji do systemu logowania, ale pakiet zostaje przekazany do kolejnej reguły filtrującej. Jeżeli pakiet nie zostanie dopasowany do żadnej reguły, to zastosowana zostanie polityka domyślna. Jako "cel" można podać trzy podstawowe cele lub cele zdefiniowane w modułach dodatkowych. Podstawowe "cele" to wspomniany już LOG, ACCEPT - przepuszczenie pakietu i DROP - odrzucenie pakietu. Z celów dodatkowych często korzysta się z celu REJECT umożliwiającego wygenerowanie datagramu ICMP z informacją o odrzuceniu pakietu (DROP nie daje takiej informacji).

I to by było na tyle informacji podstawowych. Przejdziemy teraz do bardziej praktycznych zagadnień.

11.2 Prace przygotowawcze

W ramach prac przygotowawczych należy :

- utworzyć plik/skrypt, do którego będziemy zapisywać polecenia konfigurujące nasza zaporę ogniową
- sporządzić wykaz adresów IP hostów o specjalnym znaczeniu np. hosty, z których będziemy łączyć się przez ssh
- sporządzić wykaz usług, które chcemy przepuszczać przez nasz ogniomurek

Pierwszy punkt nie stanowi większego problemu, ale podam propozycję tak dla kompletu informacji. Wzorując się na systemach BSD proponuję utworzyć plik `rc.firewall` w katalogu `/etc/rc.d`. Ponieważ ma to być skrypt, to należy mu nadać uprawnienia do wykonania, więc :

```
[root@dreptak root]# cd /etc/rc.d
[root@dreptak rc.d]# touch rc.firewall
[root@dreptak rc.d]# chmod 700 rc.firewall
```

Teraz otwieramy nasz plik do edycji i wpisujemy dane rozpoczynające skrypt :

```
#!/bin/bash

IPTABLE=/sbin/iptables

# interfejsy sieciowe
INET_IFACE=eth0
LAN_IFACE=eth1

# adresy na serwerze
INET_ADR=10.1.1.2/255.255.255.255
LAN_ADR=192.168.1.1/255.255.255.255
LAN=192.168.1.0/255.255.255.0

# Wyczysc wszystkie reguly
$IPTABLE -F
$IPTABLE -X
```

W tej chwili nasz skrypt nie robi jeszcze nic pożytecznego, jedynie czyści wszystkie łańcuchy z reguł, które mogłyby zakłócić działanie naszego skryptu.

Dwa pozostałe punkty wykonujemy na kartce i będziemy dopisywać stopniowo do skryptu. Numery portów potrzebne do konfigurowania usług działających na protokołach TCP i UDP znajdziesz w pliku `/etc/services`. W pierwszej kolumnie tego pliku znajduje się symboliczna nazwa usługi — można ją wykorzystywać zamiennie z numerem portu (`iptables` umie korzystać z tego pliku). W kolumnie drugiej znajduje się numer portu i nazwa protokołu dla danej usługi. Trzecia kolumna, jeśli jest wypełniona, zawiera inną nazwę usługi tzw. alias. W kolumnie czwartej może znajdować się opis usługi.

Pamiętaj, aby zapewnić sobie dostęp do konsoli serwera. W czasie konfigurowania zapory ogniowej może się zdarzyć, że popełnisz jakiś błąd i odetniesz się od serwera. W takiej sytuacji jedynym wyjściem jest dostęp lokalny. Można również zastosować inne rozwiązanie. Mianowicie, tworzy się skrypt wywołujący skrypt `rc.firewall` na określony okres czasu, po którym następuje wyłączenie firewalla. Rozwiązanie to jednak pozostawiam Tobie do samodzielnej realizacji.

11.3 Polityka bezpieczeństwa

Tworzenie zapory ogniowej należy zacząć od przyjęcia odpowiedniej polityki. Najczęściej przyjmuje się jedną z dwóch polityk :

1. przepuszczamy wszystko z wyjątkiem tego, co uważamy za zbędne/niebezpieczne
2. blokujemy wszystko z wyjątkiem tego, co jest nam potrzebne

Wielu użytkowników Auroksa korzysta z aplikacji *lokkit* znanej jako *"Poziom bezpieczeństwa"*. Aplikacja ta wykorzystuje pierwszą politykę. Jednak większość podręczników zaleca stosowanie drugiej polityki i tak czyni większość doświadczonych administratorów. Uzasadnienie jest proste. W pierwszym przypadku musimy pamiętać o wszystkich przypadkach, które chcemy zablokować. Jeśli o jakimś zapomnimy, to może to doprowadzić do powstania luki w systemie bezpieczeństwa. Jeśli zapomnimy o czymś w drugim przypadku, to konsekwencją będą problemy z działaniem jakiejś aplikacji, ale to nie stanowi problemu z bezpieczeństwem. Na podstawie logów możemy łatwo dojść do tego, czego nam w konfiguracji ogniomurka brakuje i poprawić jego konfigurację.

Dlatego też dla naszego serwera przyjmuję drugą politykę, czyli zamykamy całość połączeń i stopniowo będziemy otwierać te połączenia, które będą nam potrzebne. Zatem do dzieła. Otwieramy `rc.firewall` do edycji i dopisujemy polecenia ustawiające domyślne polityki.

```
# Ustaw policy
$IPTABLE -P INPUT DROP
$IPTABLE -P FORWARD DROP
$IPTABLE -P OUTPUT DROP
# $IPTABLE -P OUTPUT ACCEPT
```

Ostatnia linia jest zakomentowana, więc nie ma wpływu na konfigurację. Dodałem ją jednak z rozmysłem. O ile dla łańcuchów INPUT i FORWARD zawsze stosujemy politykę DROP, o tyle dla OUTPUT można nieco rozluźnić politykę dopuszczając wysyłanie przez serwer wszystkich pakietów. Wystarczy wstawić `#` przed pierwszą linią OUTPUT, a skasować przed drugą. Oczywiście należy wcześniej sprawdzić co nasz serwer wysyła.

Przy takich ustawieniach nasz serwer staje się głuchy jak pień, a przecież nie o to nam chodziło. Jeśli dla OUTPUT przyjąłeś drugie rozwiązanie, to serwer będzie wysyłał pakiety, ale nie przyjmie żadnego pakietu, gdyż ustawienia dla INPUT nie pozwolą na to. Poradzimy sobie z tym problemem dodając regułę do łańcucha INPUT zezwalającą na przyjęcie pakietów stanowiących odpowiedź na nasze zapytania. Dodamy również regułę pozwalającą na ruch na interfejsie loopback.

```
# Przepusc wszystko na loopbacku
$IPTABLE -A INPUT -i lo -j ACCEPT

/sbin/modprobe ip_conntrack

# Przepusc pakiety wracajace
$IPTABLE -A INPUT -i $INET_IFACE -j ACCEPT -m state --state ESTABLISHED,RELATED
$IPTABLE -A INPUT -i $LAN_IFACE -j ACCEPT -m state --state ESTABLISHED,RELATED
```

Dzięki załadowaniu modułu ip_conntrack uruchamiamy śledzenie sesji połączenia. W tym momencie nasz ogniomurek staje się filtrem pakietów znanym jako *"statefull inspection"*. W regułach zastosowany został moduł `state` odpowiedzialny za badanie stanu pakietów. Reguły te należy rozumieć jako *"przepuść pakiety należące do zestawionej sesji lub z taką sesją związane"*.

Teraz należy wykonać skrypt `rc.firewall` i nasz serwer zaczyna działać całkiem poprawnie. Wprawdzie nie można się do niego dostać zdalnie (do żadnych usług), ale z serwera możemy sięgać wszędzie, gdzie nam się podoba. Jeśli jednak przyjąłeś dla OUTPUT to bardziej restrykcyjne rozwiązanie, to nadal nie można z serwera wysyłać żadnych pakietów. Aby pakiety mogły z serwera wychodzić trzeba dodać odpowiednie reguły do łańcucha OUTPUT. Podam przykład umożliwiający dostęp do stron www, jednak w dalszej części przyjmuję, że zastosowano mniej restrykcyjną politykę :

```
$IPTABLE -A OUTPUT -p tcp --dport 80 -j ACCEPT
```

Zatem pracując na serwerze możemy sięgać do zdalnych stron www przykładowo korzystając z `lynx'a`.

Mówiąc o polityce bezpieczeństwa trzeba jeszcze zwrócić uwagę na kilka drobnych wpisów, choć bardzo istotnych. Zaczniemy od ustawienia parametrów pracy jądra.

```
# konfiguracja jadra
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
```

Polecenia powyższe wstawiamy do pliku `rc.firewall` jeszcze przed poleceniami ładującymi moduły. Pierwsza linia włącza weryfikację adresu źródłowego zapobiegając atakowi IP Spoofing. Druga linia zapobiega atakowi polegającemu na fragmentacji pakietów. Trzecia linia powoduje ignorowanie datagramów ICMP typ 5 czyli zapobiega przekierowywaniu pakietów, które może być wykorzystane przez napastnika do skanowania naszej sieci.

Kolejnym elementem, którym powinniśmy się zająć, to odpowiadanie na pingi. Wiele osób uważa, że należy tak konfigurować zapory ogniowe, aby nasz host nie odpowiadał na pingi uzasadniając to obroną przed skanowaniem sieci. Faktycznie, odpowiedź na ping z danego adresu może napastnikowi dostarczyć pewnych informacji, ale nader skromnych. Te same informacje uzyska innymi metodami. Brak odpowiedzi na ping stanowi jednak utrudnienie dla administratora, próbującego dokonać diagnostyki sieci. Dlatego też uważam, że blokowanie pingów nie podnosi bezpieczeństwa, a jedynie utrudnia administracje siecią i dlatego zalecam przepuszczanie pingów. Regułę odpowiadającą za pingi umieszczamy możliwie blisko początku listy reguł najlepiej zaraz za regułą dla loopbacku.

```
# Odpowiada/przyjmuje ping
$IPTABLE -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Kolejna reguła, którą zalecałbym dodać, wymaga pewnego wyjaśnienia. Chodzi mianowicie o usługę `ident`. Usługa ta odpowiada na pytania o nazwę użytkownika. Z oczywistych względów ujawnianie listy użytkowników jest niewskazane i w związku z tym usługi takiej nie będziemy

na naszym serwerze uruchamiać (czasami jest to konieczne np. przy korzystaniu z serwisu IRC). Ponieważ przyjęliśmy politykę DROP dla łańcucha INPUT, to już mamy pewność, że zapytania kierowane do identa będą odrzucane. Ale jest pewien problem. Otóż DROP nie informuje nadawcy o odrzuceniu pakietu i ten czeka na odpowiedź, aż upłynie limit czasu (nazywany timeout). Tymczasem z identa korzystają również całkiem "legalne" serwisy jak choćby serwer pocztowy, a nasza konfiguracja powoduje niepotrzebną zwłokę w jego działaniu. W związku z tym proponuję dopisanie reguły, która odrzuci pakiety kierowane do identa wysyłając jednocześnie datagram ICMP informujący klienta, że taka usługa nie istnieje na naszym serwerze :

```
# Pytanie o ident
$IPTABLE -A INPUT -p tcp --dport 113 -j REJECT --reject-with icmp-port-unreachable
```

Po tych zmianach mamy już całkiem elegancko skonfigurowany ogniomurek. Należy tylko ponownie uruchomić nasz skrypt, aby wprowadzone zmiany odniosły skutek.

11.4 Dostęp z sieci wewnętrznej

Zajmiemy się teraz konfigurowaniem dostępu do naszego serwera z hostów pracujących w naszej sieci. Nasi klienci muszą mieć dostęp do swoich skrzynek pocztowych znajdujących się na serwerze (usługa POP3), muszą mieć możliwość wysyłania poczty (usługa SMTP) oraz potrzebują dostępu do serwisu www i ftp. Ponieważ nie chcemy generować niepotrzebnego ruchu w sieci, to przyjmujemy, że nasz serwer będzie pełnił rolę serwera DNS i NTP (synchronizacja czasu) dla komputerów z sieci lokalnej. Dodatkowo administrator musi mieć dostęp do shella na serwerze, więc potrzebny będzie dostęp do serwisu SSH, ale tylko z tego jednego adresu. Tak wyglądają założenia dla naszej sieci LAN, ale możliwe, że u Ciebie będzie to wyglądało nieco inaczej. Jeśli odpowiednio przyłożyłeś się do prac przygotowawczych, to już widzisz różnice.

Przystępujemy zatem do modyfikowania pliku `rc.firewall`. Zaczniemy od dodania hosta administratora :

```
(...)
LAN=192.168.1.0/255.255.255.0

ADMIN_IP=192.168.1.2

# konfiguracja jadra
(...)
```

Ponieważ większość usług ma być traktowana analogicznie dla wszystkich hostów w naszej sieci, to warto stworzyć listę portów odpowiednich dla tych usług, a następnie wykorzystać ją w pojedyńczej regule. Ze względu na serwis FTP należy również zadbać o załadowanie modułu ip_conntrack_ftp odpowiedzialnego za śledzenie sesji FTP — bez tego modułu nie będzie działać tryb *passive*.

```
(...)
ADMIN_IP=192.168.1.2

TCP_LAN=20,21,80,443,25,110
UDP_LAN=53,123

# konfiguracja jadra
(...)
/sbin/modprobe ip_conntrack
/sbin/modprobe ip_conntrack_ftp

# Przepusc pakiety wracajace
(...)
```

Mamy już przygotowane wszystkie elementy niezbędne do opracowania odpowiednich reguł, więc przystępujemy do ich utworzenia :

```
# Pytanie o ident
(...)
$IPTABLE -A INPUT -p tcp --dport 113 -j REJECT --reject-with icmp-port-unreachable

# Wpuszczam ssh od admina
$IPTABLE -A INPUT -i $LAN_IFACE -p tcp -s $ADMIN_IP --dport ssh -j ACCEPT

# Dopuszczone z LAN'u
$IPTABLE -A INPUT -i $LAN_IFACE -p tcp -s $LAN -j ACCEPT -m multiport \
--destination-port $TCP_LAN
$IPTABLE -A INPUT -i $LAN_IFACE -p udp -s $LAN -j ACCEPT -m multiport \
--destination-port $UDP_LAN

#$IPTABLE -A INPUT -s $LAN -j LOG --log-prefix '[z lanu]'
```

Ostatnia linia jest zakomentowana, aby nasz log nie zapełniał się informacjami, które nas w tej chwili nie interesują. Ponieważ jest to ostatnia linia, to po odkomentowaniu trafią do niej te wszystkie pakiety, które nie pasowały do wcześniejszych reguł, a więc zostaną odrzucone. Inaczej mówiąc w logach znajdą się informacje o działaniach naszych klientów, które uważamy za niepożądane.

Teraz wykonujemy nasz skrypt i możemy zacząć testowanie działania naszego ogniomurka. Nie przewiduję tu większych problemów z jedną wszakże uwagą. Czy pamiętałeś o zapewnieniu sobie dostępu do systemu? Jeśli popełniłeś błąd przy wpisywaniu reguł i nie masz dostępu do konsoli, to masz poważny problem. Mam jednak nadzieję, że nie doszło do tak tragicznej sytuacji i możesz przejść do kolejnych kroków.

11.5 Dostęp z internetu

Analogicznie jak dla sieci lokalnej konfigurujemy dostęp z internetu. Tu jednak przepuszczamy znacznie mniejszą ilość pakietów. Nasz serwer będzie głównie klientem, czyli będzie wysyłał zapytania i odbierał odpowiedzi. Otwarcie sesji przez nasz system wiąże się z przejściem pakietu przez łańcuch OUTPUT, a ten przepuszcza wszystko. Pakiet powracający trafia do łańcucha INPUT, a ten stwierdzi, że jest to pakiet należący do sesji przez nas otwartej i również go przepuści. Inaczej sytuacja wyglądać będzie tylko w przypadku serwisów udostępnianych dla internetu. Takimi serwisami będą strony www, serwis FTP oraz nadchodząca poczta (protokół SMTP). Przystępujemy zatem do modyfikowania naszego pliku `rc.firewall` :

```
(...)
UDP_LAN=53,123
TCP_INET=20,21,25,80,443

# konfiguracja jadra
(...)
# Dopuszczone z Inetu
$IPTABLE -A INPUT -i $INET_IFACE -p tcp -d $INET_ADR -j ACCEPT -m multiport \
--destination-port $TCP_INET

#$IPTABLE -A INPUT -s !$LAN -j LOG --log-prefix '[z inetu]'
```

Podobnie jak dla sieci LAN ostatnia linia, odpowiedzialna za logowanie odrzucanych pakietów jest zakomentowana. Jak zwykle musimy jeszcze wykonać nasz skrypt i przetestować działanie naszego ogniomurka. Tym razem jednak trzeba to zrobić z hosta znajdującego się poza naszą siecią, więc konieczna może się okazać pomoc kogoś znajomego.

W efekcie naszych prac powinieneś otrzymać skrypt wyglądający mniej więcej tak :

```
1  #!/bin/bash
2
3  IPTABLE=/sbin/iptables
4
5  # interfejsy sieciowe
6  INET_IFACE=eth0
7  LAN_IFACE=eth1
8
9  # adresy na serwerze
10 INET_ADR=10.1.1.2/255.255.255.255
11 LAN_ADR=192.168.1.1/255.255.255.255
12 LAN=192.168.1.0/255.255.255.0
13 ADMIN_IP=192.168.1.2
14
15 TCP_LAN=20,21,80,443,25,110
16 UDP_LAN=53,123
17 TCP_INET=20,21,25,80,443
18
19 # konfiguracja jadra
20 echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
21 echo 1 > /proc/sys/net/ipv4/tcp_syncookies
22 echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
23
24 /sbin/modprobe ip_conntrack
25 /sbin/modprobe ip_conntrack_ftp
26
27 # Wyczysc wszystkie reguly
28 $IPTABLE -F
29 $IPTABLE -X
30
31 # Ustaw policy
32 $IPTABLE -P INPUT DROP
33 $IPTABLE -P FORWARD DROP
34 # $IPTABLE -P OUTPUT DROP
35 $IPTABLE -P OUTPUT ACCEPT
36
37 # Przepusc wszystko na loopbacku
38 $IPTABLE -A INPUT -i lo -j ACCEPT
39
40 # Przepusc pakiety wracajace
41 $IPTABLE -A INPUT -i $INET_IFACE -j ACCEPT -m state --state ESTABLISHED,RELATED
42 $IPTABLE -A INPUT -i $LAN_IFACE -j ACCEPT -m state --state ESTABLISHED,RELATED
43
44 # Odpowiada/przyjmuje ping
45 $IPTABLE -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
46
47 # Pytanie o ident
48 $IPTABLE -A INPUT -p tcp --dport 113 -j REJECT --reject-with icmp-port-unreachable
49
50 # Wpuszczam ssh od admina
51 $IPTABLE -A INPUT -i $LAN_IFACE -p tcp -s $ADMIN_IP --dport ssh -j ACCEPT
52
53 # Dopuszczone z LAN'u
54 $IPTABLE -A INPUT -i $LAN_IFACE -p tcp -s $LAN -j ACCEPT -m multiport \
55 --destination-port $TCP_LAN
56 $IPTABLE -A INPUT -i $LAN_IFACE -p udp -s $LAN -j ACCEPT -m multiport \
57 --destination-port $UDP_LAN
58
59 #$IPTABLE -A INPUT -s $LAN -j LOG --log-prefix '[z lanu]'
60
61 # Dopuszczone z Inetu
```

```
62 $IPTABLE -A INPUT -i $INET_IFACE -p tcp -d $INET_ADR -j ACCEPT -m multiport \
63 --destination-port $TCP_INET
64
65 #$IPTABLE -A INPUT -s !$LAN -j LOG --log-prefix '[z inetu]'
```

11.6 Prace końcowe

Kiedy już mamy dopracowany skrypt `rc.firewall`, to należy zadbać o to, aby startował automatycznie wraz ze startem systemu. Naturalnym wydaje się dopisanie wywołania naszego skryptu do pliku `rc.local`. I faktycznie nasz ogniomurek będzie się uruchamiał wraz ze startem systemu, a ponieważ `rc.local` wywoływany jest jako ostatni element uruchamiania systemu mamy pewność, że interfejsy sieciowe są już uruchomione zanim spróbujemy uruchomić firewall. Rozwiązanie to ma jednak jedną poważną wadę. W przypadku jakichś modyfikacji interfejsów sieciowych i wykonania restartu sieci (poleceniem `service network restart`) nasz ogniomurek przestanie działać i trzeba ręcznie uruchomić skrypt. Niestety w pośpiechu bardzo łatwo zapomnieć o takim drobiazgu, a to może oznaczać problemy. Dlatego proponuję nieco inne rozwiązanie. Mianowicie proponuję wprowadzić modyfikacje do skryptu `/etc/rc.d/init.d/network`. W tym celu należy odszukać w tym pliku sekcję `start` i na końcu tej sekcji dopisać wywołanie naszego skryptu :

```
(...)
  sysctl -e -p /etc/sysctl.conf >/dev/null 2>&1

        touch /var/lock/subsys/network

# Wywolanie naszego ogniomurka
  /etc/rc.d/rc.firewall

  ;;
  stop)
(...)
```

Rozwiązanie to również nie jest idealne. Często wykorzystywane są urządzenia sieciowe, do których nie ma sterowników w jądrze systemu. W takich sytuacjach pobierane są drivery ze strony producenta sprzętu i uruchamiane "ręcznie" na ogół właśnie w pliku `rc.local`. Niestety plik `rc.local` wykonywany jest po skrypcie `network` i nasz firewall nie zadziała. Znając te zależności musisz samodzielnie wybrać sposób uruchomienia ogniomurka.

Kolejnym elementem, którym powinniśmy się zająć są logi systemowe. Komunikaty, które przekazuje "cel" LOG rozpoznawane są przez demona syslog jako komunikaty kernela. Standardowo trafiają one do pliku `/var/log/messages`, jednak możemy to zmienić, aby je wyodrębnić z natłoku innych informacji. W tym celu musimy zmodyfikować plik `/etc/syslog.conf` i wykonać restart demona.

```
kern.*                                                  /var/log/fw.log
(...)
*.info;mail.none;authpriv.none;cron.none;kern.none      /var/log/messages
```

W oryginalnym pliku logi z kernela trafiają na konsolę (urządzenie /dev/console). My tę konfigurację zmieniamy i kierujemy je do pliku `fw.log`. Aby komunikaty nie dublowały się wyłączamy przekazywanie ich do pliku `messages` (opcja kern.none). Teraz należy jeszcze utworzyć plik `/var/log/fw.log` i wykonać restart demona syslogd poleceniem `service syslog restart`. Warto jeszcze zadbać o rotację pliku `fw.log` oraz zmodyfikować skrypty logwatcha (jeśli go używasz).

Uwaga. Pamiętaj, że firewall podnosi poziom bezpieczeństwa Twojego systemu, jednak nie jest jego gwarantem. Uruchomienie go nie zwalnia Cię z obowiązku aktualizacji pakietów ani analizy logów systemowych.

Część VI

Usługi serwerowe

Rozdział 12

DNS - serwer nazw domenowych

Jednym z elementów konfiguracji usług sieciowych jest system rozwiązywania nazw. Ale czy rzeczywiście potrzebujesz serwera DNS? W małej sieci (dwa, trzy hosty) nie musisz mieć serwera DNS, ale i tak będziesz korzystał z usług tego serwisu tyle, że sięgać będziesz do serwera znajdującego się gdzieś w sieci internet. W naszej przykładowej instalacji mamy nieco więcej komputerów, więc uruchomienie serwera DNS jest jak najbardziej wskazane. Nie będziemy jednak budować serwera specjalnie rozbudowanego, gdyż będzie to serwer wyłącznie na potrzeby naszej sieci lokalnej i nie będzie dostępny z internetu. Serwerem z prawdziwego zdarzenia jest serwer prowadzony przez naszego dostawcę internetu i to on zadba o to, aby nasz serwer był odpowiednio widziany w sieci. Nawet jeśli nie masz dużej sieci lokalnej, to warto przeczytać ten rozdział abyś wiedział o czym rozmawiać z dostawcą internetu.

Zacznijmy zatem poznawanie mechanizmu rozwiązywania nazw od podstawowych elementów. Komputery w sieci porozumiewają się korzystając z adresów IP. Jednak dla człowieka zapamiętanie adresów IP jest raczej kłopotliwe więc od dawna próbowano jakoś zastąpić adresy IP nazwami zrozumiałymi i łatwymi do zapamiętania przez użytkowników. Pierwszym takim rozwiązaniem, stosowanym do dzisiaj jest plik `/etc/hosts`. Plik ten zawiera informacje o adresach IP, pełnych nazwach hostów oraz aliasach tych nazw. Na naszym serwerze plik ten będzie wyglądał następująco :

```
127.0.0.1       localhost.localdomain   localhost
192.168.1.1     dreptak.tupward.pl      dreptak server
```

Możemy do tego pliku dodać jeszcze dwie linie opisujące znane hosty :

```
192.168.1.2     admin.tupward.pl        admin
192.168.1.3     vip.tupward.pl          vip
```

Wprowadzanie większej ilości hostów do tego pliku mija się z celem, gdyż hosty w pracowni będą miały adresy IP przyznawane dynamicznie. Kolejnym problemem jest to, że wpisy znajdujące się w tym pliku wykorzystywane są tylko i wyłącznie przez system, w którym ten plik się znajduje. Z tego wynika wniosek, że na wszystkich komputerach w sieci trzeba by było utworzyć takie pliki. I wyobraź sobie teraz, że z jakiegoś powodu musisz dokonać jakichś zmian w konfiguracji sieci. W efekcie, powstałe zmiany musisz odwzorować w plikach `hosts` na wszystkich komputerach. A jeśli w sieci masz 100 komputerów?

Teraz jeszcze należy zadbać o to, aby system wiedział czego szukać do celu rozwiązywania nazw podanych jako adresy wywołań. Do tego celu wykorzystuje się plik `/etc/host.conf`. Plik ten zawiera tylko jedną linię :

```
order hosts,bind
```

Tak to wygląda po zainstalowaniu systemu i nie będziemy tego zmieniać. Zapis powyższy oznacza, że system ma rozwiązywać nazwy na adresy IP korzystając z pliku `hosts`, a w przypadku niepowodzenia szukać rozwiązania w serwisie DNS.

Aby sprawdzić działanie naszego serwisu rozwiązywania nazw możemy wykonać prosty test. Sprawdzimy czy możemy pingować hosty podając jako adres nazwę zapisaną w pliku `hosts`.

```
[tuptus@dreptak tuptus]$ ping -c4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.063 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.037 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 0.035/0.043/0.063/0.011 ms, pipe 2
[tuptus@dreptak tuptus]$
[tuptus@dreptak tuptus]$ ping -c4 localhost
PING localhost.localdomain (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=0 ttl=64 time=0.056 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=3 ttl=64 time=0.046 ms

--- localhost.localdomain ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3016ms
rtt min/avg/max/mdev = 0.036/0.046/0.056/0.007 ms, pipe 2
[tuptus@dreptak tuptus]$
[tuptus@dreptak tuptus]$ ping -c4 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.061 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.037 ms

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3017ms
rtt min/avg/max/mdev = 0.037/0.044/0.061/0.010 ms, pipe 2
[tuptus@dreptak tuptus]$
[tuptus@dreptak tuptus]$ ping -c4 server
PING dreptak.tupward.pl (192.168.1.1) 56(84) bytes of data.
64 bytes from dreptak.tupward.pl (192.168.1.1): icmp_seq=0 ttl=64 time=0.058 ms
64 bytes from dreptak.tupward.pl (192.168.1.1): icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from dreptak.tupward.pl (192.168.1.1): icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from dreptak.tupward.pl (192.168.1.1): icmp_seq=3 ttl=64 time=0.032 ms

--- dreptak.tupward.pl ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3017ms
rtt min/avg/max/mdev = 0.032/0.041/0.058/0.010 ms, pipe 2
[tuptus@dreptak tuptus]$
```

Wygląda na to, że wszystko działa poprawnie. Możemy zatem przejść do bardziej zaawansowanej technologii.

Opisana powyżej metoda jest dobra w małej sieci, może również uprościć dostęp do często wykorzystywanych adresów, ale nie jest rozwiązaniem zadowalającym jeśli chcemy korzystać z internetu. Nie będziemy przecież wpisywać tysięcy adresów do pliku `hosts`. Wspominałem już o pliku `host.conf` i kolejności poszukiwania rozwiązania nazwy na adres IP. Pisałem wówczas, że w drugiej kolejności nasz system ma korzystać z serwisu DNS. Ale my nie mamy takiego serwisu, więc gdzie go szukać? Na to pytanie odpowiada zawartość pliku `/etc/resolv.conf`. W najprostszej postaci plik ten zawiera listę serwerów DNS, z których ma korzystać nasz system.

```
nameserver 80.50.50.100
nameserver 80.50.50.150
```

W podanym wyżej przykładzie wykorzystałem adresy serwerów sieci *tpnet.pl*, ale Ty powinieneś wpisać serwery, które poda Ci dostawca internetu. Jeśli korzystasz z serwisu DHCP przy dostępie do internetu, to aplikacja *dhclient* samodzielnie dokona odpowiednich wpisów w tym pliku.

Uwaga. *dhclient* modyfikuje plik `resolv.conf` próbując ułatwić nam pracę, ale czasami jest to działanie szkodliwe, gdyż uniemożliwia nam skonstruowanie własnego pliku. Aby zapobiec modyfikowaniu pliku `resolv.conf` powinniśmy do pliku `/etc/sysconfig/network-scripts/ifcfg-eth0` wpisać linię `PEERDNS=no`.

Na liście możesz mieć wiele serwerów, ale w obecnej wersji systemu wykorzystywane są tylko pierwsze trzy. Nie oznacza to, że wszystkie są odpytywane. System wysyła zapytanie do pierwszego serwera i jeśli otrzyma odpowiedź, to drugiego już nie będzie odpytywać. Zwróć uwagę na to, że informacja o braku poszukiwanej nazwy w systemie DNS również jest odpowiedzią. Za brak odpowiedzi uważany jest jedynie przypadek, gdy w określonym czasie serwer nic nie odpowie. W takiej sytuacji zapytanie kierowane jest do serwera, którego adres znajduje się na kolejnej pozycji.

Jeśli już wprowadziłeś odpowiednie adresy, to czas na przetestowanie działania systemu.

```
[tuptus@dreptak Aurox-PT]$ host www.aurox.org
www.aurox.org has address 62.111.243.84
[tuptus@dreptak Aurox-PT]$
```

Wygląda na to, że system działa poprawnie. Możemy teraz dokonać analogicznych wpisów na pozostałych komputerach w sieci. Piszę "możemy", gdyż istnieją również inne metody ustawienia serwisu rozwiązywania nazw na komputerach w naszej sieci LAN. Pisałem już o korzystaniu z DHCP, w naszej sieci również będziemy uruchamiać ten serwis, więc komputery klienckie będą otrzymywać odpowiednie informacje z tego serwisu. Jest tylko pewien problem, a właściwie dwa problemy. Po pierwsze nigdzie w internecie nie znajdziemy wpisów dotyczących hostów z naszej sieci LAN. Po drugie wszystkie hosty z naszej sieci odwoływałyby się do serwerów DNS znajdujących się poza naszą siecią LAN generując ruch na bramce. Możemy ten ruch znacznie ograniczyć jeśli maszyny lokalne będą odwoływać się do lokalnego serwera DNS, a ten będzie do internetu sięgał tylko wtedy, gdy zajdzie taka potrzeba (część informacji będzie przechowywana w pamięci podręcznej tego serwera). Tak więc dochodzimy do istoty tego rozdziału — potrzebujemy własnego serwera DNS.

12.1 Konfiguracja serwera

Konfigurację naszego serwera DNS zaczynamy od zainstalowania odpowiednich programów. Potrzebne nam będą trzy pakiety: `bind`, `bind-utils` oraz `caching-nameserver`. Należy je zainstalować wraz z pakietami powiązanymi zależnościami z wymienionymi pakietami. Teraz możemy przystąpić do budowania własnego serwisu. W tym momencie musimy podjąć ważną decyzję: korzystamy ze środowiska *root jail*, czy nie? Decyzja ta jest o tyle istotna, że rozmieszczenie plików konfiguracyjnych będzie się nieco różnić w zależności od wybranego rozwiązania. Środowisko *root jail* jest rozwiązaniem bezpieczniejszym, ale powoduje nieco problemów przy konfiguracji. Podstawą jest odpowiedni wpis w pliku `/etc/sysconfig/named` :

```
#ROOTDIR=/var/named/chroot
```

```
ROOTDIR=/
```

Jak widzisz pierwszą linię zakomentowałem, gdyż nie zamierzam korzystać z *root jail*. Jeśli jednak zdecydujesz się na stosowanie tego środowiska, to wszystkie opisywane pliki musisz odnosić do katalogu podanego w zmiennej `ROOTDIR`, gdyż dla demona ten katalog będzie początkiem drzewa katalogów. Zatem jeśli ja będę pisał o pliku `/etc/named.conf`, to w środowisku *chroot* będzie to `/var/named/chroot/etc/named.conf`.

Kiedy już podjęcie pierwszej decyzji mamy za sobą możemy przystąpić do konfiguracji naszego serwisu DNS. Zaczynamy od pliku `/etc/named.conf`. Plik ten zawiera ogólną konfigurację serwera, prawa dostępu oraz informacje o obsługiwanych domenach. Ponieważ serwer ma działać tylko na rzecz naszej sieci lokalnej wprowadzimy drobne zmiany w jego konfiguracji.

```
options {
    directory "/var/named";
    listen-on { 192.168.1.1; 127.0.0.1; };
    };
```

Opcja `listen-on` ustawia interfejsy, na których nasz serwer będzie nasłuchiwał pytań. Dzięki temu wpisowi będzie odporny na pytania pochodzące z inetu.

Teraz możemy wykonać restart demona `named`, a na stacjach klienckich podać serwer 192.168.1.1 jako serwer DNS zamiast serwerów spoza naszej sieci. Na naszym serwerze warto też zmodyfikować plik `/etc/resolve.conf` wpisując na pierwszym miejscu `nameserver 127.0.0.1`. Konfiguracja taka znana jest jako serwer cache. Działanie tego serwera polega na gromadzeniu w pamięci danych zebranych z sieci. Kiedy stacja z naszej sieci zgłosi zapotrzebowanie na rozwiązanie nazwy na adres IP lub odwrotnie serwer najpierw sprawdzi informacje w cache'u i jeśli je znajdzie, to przekaże klientowi. Jeśli informacji takich nie posiada, to wyszuka je w sieci, zapisze w swoim buforze i udostępni klientowi.

To mamy zrobiony pierwszy krok — zmniejszyliśmy ruch w sieci. Teraz kolej na stworzenie odpowiednich wpisów dla naszej sieci lokalnej. Zaczynamy od dopisania do pliku `/etc/named.conf` informacji o plikach stref, czyli plikach zawierających informację o naszej domenie.

```
zone  "tupward.pl" {
        type master;
        file  "tupward.pl.zone";
};

zone  "1.168.192.in-addr.arpa" {
        type master;
        file  "1.168.192.in-addr.arpa.zone";
};
```

Pierwsza sekcja opisuje informacje o naszej domenie : w pliku `/var/named/tupward.pl.zone` (patrz opcja `directory`) znajduje się główna konfiguracja dla domeny *tupward.pl*. Oprócz typu `master` może być jeszcze kilka innych typów. Tutaj wspomnę jedynie typ `slave`. Wyobraź sobie sytuację, gdy jakiś serwer DNS w internecie z jakiegoś powodu przestaje odpowiadać. W tym momencie część sieci staje się niedostępna. Oczywiście niewybaczalne jest dopuszczenie do takiej sytuacji. Dlatego wymyślono serwery *primary* (to te z ustawionym typem `master`) oraz *secondary* (to te z ustawionym typem `slave`). W określonych odstępach czasu serwery *secondary* sprawdzają, czy nastąpiły jakieś zmiany na serwerze *primary* i jeśli takie zmiany zaszły to "ściągają" te dane do siebie.

Proces ten nazywany jest transferem strefy. I tutaj bardzo ważna informacja. Zapytania DNS przesyłane są na port 53/UDP i ogniomurek musi przepuszczać pakiety wysyłane na ten port i powracające z tego potu. Natomiast transfer strefy dokonywany jest na porcie 53/TCP. Ponieważ wiele metod włamań wykorzystuje tę właściwość, to na ogniomurku należy bardzo dokładnie

określić z jakimi adresami taką komunikację dopuszczamy. Oczywiście w konfiguracji samego demona również określamy adresy, którym zezwalamy na dokonywanie transferu. Ponieważ naszej strefy nie będziemy nikomu udostępniać, to w konfiguracji blokujemy ruch na porcie 53/TCP.

Druga sekcja to opis tzw. strefy odwrotnej. Strefa ta pozwala na odwzorowanie adresu IP na nazwę, czyli odwrotnie do tego, co dotychczas staraliśmy się osiągnąć. Z informacji tej korzysta wiele serwisów sprawdzając czy klient nie próbuje podszywać się pod inny adres. W wielu wypadkach połączenie jest odrzucane gdy wynik zwracany przez revDNS różni się od nazwy jaką przedstawił się klient lub gdy brak jest wpisu w revDNS.

12.2 Pliki stref

Przejdziemy teraz do przygotowania plików stref. Opcja `directory` określa katalog, w którym należy utworzyć odpowiednie pliki. W naszym przypadku jest to katalog `/var/named`. Przechodzimy zatem do tego katalogu i tworzymy dwa pliki : `tupward.pl.zone` i `1.168.192.in-addr.arpa.zone`. Konfigurację zaczniemy od pliku strefy prostej :

```
$ORIGIN .
tupward.pl              IN SOA  dreptak.tupward.pl. root.dreptak.tupward.pl. (
                                280820041  ; serial
                                28800      ; refresh (8 hours)
                                7200       ; retry (2 hours)
                                604800     ; expire (1 week)
                                86400      ; minimum (1 day)
                                )
                        NS      dreptak.tupward.pl.
                        MX      100 dreptak.tupward.pl.
$ORIGIN tupward.pl.
dreptak                 A       192.168.1.1
admin                   A       192.168.1.2
vip                     A       192.168.1.3
www                     CNAME   dreptak
```

Omówimy teraz poszczególne elementy tego pliku. Definicję strefy zaczynamy od zdefiniowania rekordu SOA. Rekord ten zawiera informacje o hoście zawierającym definicję strefy, adres administratora (zwróć uwagę na brak znaku @ w adresie) oraz informacje o czasie ważności danych zawartych w pliku strefy. Bardzo istotna jest tu pierwsza wartość określająca numer seryjny pliku. Na podstawie czterech kolejnych wartości inne serwery DNS wiedzą jak często powinny sprawdzać aktualność danych. Jeśli stwierdzają, że nadszedł czas na aktualizację, to sprawdzany jest właśnie numer seryjny i na tej podstawie wiedzą, czy nastąpiła zmiana, czy nie. W naszym wypadku nie ma to większego znaczenia, ale "czym skorupka za młodu ...". Bardzo często administratorzy jako numer seryjny wykorzystują bieżącą datę i numer kolejnej zmiany w danym dniu.

Kolejny rekord opisuje serwer DNS dla domeny, którą właśnie definiujemy. Właściwie linia ta powinna wyglądać następująco :

```
tupward.pl.             NS      dreptak.tupward.pl.
```

Początkowy ciąg możemy jednak pominąć, gdyż jest on przyjmowany domyślnie. Istotny by był, gdybyśmy chcieli dokonać delegacji poddomeny np. :

```
klasa.tupward.pl.       NS      ns.klasa.tupward.pl.
$ORIGIN tupward.pl.
```

Pamiętaj jednak, że po wpisaniu takiej poddomeny jej nazwa staję się ciągiem, który będzie dodawany w miejscu, gdzie powinna być wpisana domena. Właśnie dlatego przed listą hostów

należy przywrócić domyślną wartość nazwy domeny. Zwróć też uwagę na kropki kończące nazwy poszczególnych hostów. Kropka ta zapobiega dodawaniu nazwy domeny do wcześniejszego ciągu. Żeby to dokładniej zrozumieć :

- `dreptak.tupward.pl.` rozumiane jest dokładnie tak jak wpisano, czyli `dreptak.tupward.pl`
- `dreptak.tupward.pl` zostanie rozwinięte do `dreptak.tupward.pl.tupward.pl`, co byłoby oczywistym błędem

Zapewne już pomyślałeś "to dlaczego nie wpisać samego dreptak?". Tak, masz rację. Można tak postąpić, ale w przypadku definiowania serwerów nazw i serwerów pocztowych (następny rekord) radziłbym zachować pełną formę zapisu.

Skoro już o tym wspomniałem, to przejdźmy do kolejnego rekordu. Nazwa `MX` oznacza *mail exchanger*, czyli serwer pocztowy. Rekordów takich może być kilka, a kolejność wybierania serwera, do którego przesłana zostanie poczta ustalana jest na podstawie wartości parametru znajdującego się za MX. Im parametr ten jest większy, tym mniejszy jest priorytet serwera. W naszym przypadku rekord ten nie ma żadnego znaczenia, ale dodałem go dla porządku.

Teraz przechodzimy do opisu poszczególnych hostów. W pierwszej kolumnie wpisujemy nazwę hosta (możemy stosować zapis skrócony), dalej znacznik rekordu `A` oraz adres IP danego hosta. Myślę, że ten rekord nie wymaga specjalnego omawiania.

Ciekawym rekordem jest ostatni w naszym pliku rekord `CNAME`. Rekord ten wykorzystujemy do budowania serwerów wirtualnych i nazywany jest często aliasem. Zapis taki jak w naszym pliku strefy oznacza, że host o nazwie *www.tupward.pl* jest tym samym hostem, co *dreptak.tupward.pl*, tylko znanym pod inną nazwą. Taki prosty wpis, a jakie możliwości. Dzięki temu mechanizmowi możemy tworzyć wiele nazw dla jednego hosta i wykorzystywać je do tworzenia serwerów wirtualnych. Mechanizm ten jest szczególnie przydatny przy tworzeniu wirtualnych serwerów www.

12.3 Pliki stref odwrotnych

Podobnie jak w poprzednim przypadku otwieramy odpowiedni plik i wprowadzamy kolejne rekordy

```
$ORIGIN .
1.168.192.in-addr.arpa  IN SOA  dreptak.tupward.pl. root.dreptak.tupward.pl. (
                                280820041  ; serial
                                28800      ; refresh (8 hours)
                                7200       ; retry (2 hours)
                                604800     ; expire (1 week)
                                86400      ; minimum (1 day)
                                )
                        NS      dreptak.tupward.pl.
$ORIGIN 1.168.192.in-addr.arpa.
1                       PTR     dreptak.tupward.pl.
2                       PTR     admin.tupward.pl.
3                       PTR     vip.tupward.pl.
```

Jak widzisz struktura tego pliku jest bardzo zbliżona do pliku strefy prostej. Podstawową różnicą jest występowanie rekordu `PTR` zamiast rekordu `A`. W pierwszej kolumnie podajemy adres IP (w tym wypadku tylko ostatni oktet), dalej typ rekordu (czyli `PTR`) i na koniec nazwa hosta.

12.4 Testowanie DNS

Po wprowadzeniu wszystkich zmian wykonujemy restart demona named. Oczywiście należy sprawdzić w pliku `/var/log/messages` czy start serwera zakończył się powodzeniem. Jeśli nie popełniłeś jakiejś literówki, to najprawdopodobniej start serwera powiódł się i możemy przystąpić do testowania naszego serwisu. Do tego celu wykorzystamy aplikację *nslookup*. Zaczniemy od sprawdzenia co aplikacja powie nam na temat serwera naszej domeny, czyli pytać będziemy o rekordy NS :

```
[root@dreptak named]# nslookup
Note:  nslookup is deprecated and may be removed from future releases.
Consider using the 'dig' or 'host' programs instead.  Run nslookup with
the '-sil[ent]' option to prevent this message from appearing.
> set q=NS
> tupward.pl.
Server:         192.168.1.1
Address:        192.168.1.1#53

tupward.pl      nameserver = dreptak.tupward.pl.
>
```

Jak widać odpowiedź poprawna. Pytajmy zatem dalej. Dowiedzmy się co wiadomo o hoście *dreptak* :

```
> set q=A
> dreptak.tupward.pl.
Server:         192.168.1.1
Address:        192.168.1.1#53

Name:   dreptak.tupward.pl
Address: 192.168.1.1
>
```

W takim razie sprawdźmy czy równie poprawną odpowiedź otrzymamy pytając o adres IP :

```
> set q=PTR
> 192.168.1.1
Server:         192.168.1.1
Address:        192.168.1.1#53

1.1.168.192.in-addr.arpa        name = dreptak.tupward.pl.
>
```

Wykonajmy jeszcze kilka testów :

```
> set q=ANY
> www.tupward.pl.
Server:         192.168.1.1
Address:        192.168.1.1#53

www.tupward.pl  canonical name = dreptak.tupward.pl.
> tupward.pl.
Server:         192.168.1.1
Address:        192.168.1.1#53

tupward.pl
        origin = dreptak.tupward.pl
        mail addr = root.localhost.domek.pl
        serial = 56
        refresh = 28800
        retry = 7200
        expire = 604800
```

```
        minimum = 86400
tupward.pl      nameserver = dreptak.tupward.pl.
tupward.pl      mail exchanger = 100 dreptak.tupward.pl.
>^D
```

Nasze zabawy z DNS'em kończymy wciskając Ctrl + d . Wygląda na to, że nasz serwer działa całkiem poprawnie. Jeśli doszedłeś do tego momentu możesz być z siebie dumny — zrobiłeś duży krok naprzód w poznawaniu konfiguracji Auroksa.

Oczywiście powyższy opis to dopiero podstawy tworzenia serwisu DNS i nie możesz jeszcze mówić, że jesteś specem w tej dziedzinie. Niemniej tak skonstruowany serwer w zupełności wystarczy do naszych potrzeb.

Rozdział 13

Serwer SSH

Jedną z zalet systemów uniksowych jest możliwość zdalnego dostępu do powłoki systemu. Od dawna funkcjonalność taka jest znana i wykorzystywana głównie dzięki demonowi telnetd. Niestety demon ten ma jedną zasadniczą wadę — cała komunikacja (również autoryzacja) odbywa się otwartym tekstem co umożliwia łatwe podsłuchanie takiej sesji i przejęcie hasła dostępu. Aby temu zapobiec stworzono serwis ssh (Secure Shell). Serwis ten to znacznie więcej niż tylko otwieranie sesji shella na zdalnej maszynie. Umożliwia również wykonywanie zdalne pojedynczych poleceń (odpowiednik polecenia rsh) a także tworzenie szyfrowanego tunelu wewnątrz którego możemy uruchamiać usługi nie posiadające możliwości szyfrowania przesyłanych danych.

13.1 Podstawowa konfiguracja

Zaczniemy od skonfigurowania serwisu umożliwiającego zdalne logowanie. W zasadzie po zainstalowaniu pakietu openssh i uruchomieniu serwisu sshd możemy już korzystać z usługi zalecałbym jednak wprowadzenie pewnych modyfikacji do pliku konfiguracyjnego. Plik, który nas interesuje to `/etc/ssh/sshd_config`. Nazwy poszczególnych parametrów są dość wymowne a dzięki dodanym komentarzom plik ten jest łatwy do zrozumienia.

Pierwszym parametrem, który może nas zainteresować jest *ListenAddress*. Parametr ten określa, na jakim adresie nasz demon będzie nasłuchiwał. Domyślnie demon nasłuchuje na wszystkich dostępnych adresach ale my mamy dwa interfejsy - jeden dostępny z internetu i drugi dostępny z sieci LAN. Możemy zatem ograniczyć dostęp tylko dla użytkowników łączących się z komputerów w sieci lokalnej ustawiając ten parametr na

```
ListenAddress 192.168.1.1
```

Teraz nasz demon nie będzie odpowiadał na zapytania trafiające na interfejs “internetowy”.

Kolejny parametr, na który powinniśmy zwrócić uwagę to *PermitRootLogin*. Domyślnie parametr ten ustawiony jest na “yes” co oznacza, że można zdalnie logować się na konto root. Niektórzy administratorzy pozostawiają takie ustawienie bez zmian uznając, że szyfrowanie kanału komunikacyjnego jest wystarczającym zabezpieczeniem hasła przekazywanego w czasie logowania. Niestety nie do końca jest to prawda i dlatego zalecam zmianę tego parametru. Za zmianą przemawia jeszcze jeden argument. Jeśli zezwolimy na logowanie na konto root to w logach będziemy mieli jedynie informacje, że z komputera X dokonano logowania na to konto ale nie wiemy kto tego dokonał. Jeżeli jednak zablokujemy taki dostęp to użytkownik będzie musiał zalogować się najpierw na swoje konto a dopiero później może przejść na konto root wykorzystując polecenie `su`. Teraz w logach mamy informację o logowaniu użytkownika oraz jego przejściu na

konto root więc mamy pełną informację o autoryzacjach. Biorąc pod uwagę powyższe zalecam ustawienie

```
PermitRootLogin no
```

Następnym parametrem, którym należy się zainteresować jest parametr *KeyRegenerationInterval* odpowiedzialny za częstotliwość zmiany klucza szyfrującego. Domyślnie parametr ten ustawiony jest na 3600 sekund czyli jedną godzinę. Oznacza to, że co godzinę będzie następowała zmiana klucza szyfrującego transmisję. Wartości tej na ogół nie trzeba zmieniać ale niestety niektóre aplikacje klienckie mają problemy z tą funkcjonalnością. Jeśli użytkownik otworzy sesję ssh i będzie pozostawał nie aktywny przez podany okres czasu lub uruchomi jakiś długotrwały proces, w trakcie którego nie będzie żadnej interakcji użytkownika z powłoką to może nastąpić zerwanie sesji i tym samym konieczność ponownego logowania. W takim przypadku można wydłużyć czas pomiędzy kolejnymi zmianami kluczy lub wręcz wyłączyć tę funkcjonalność podając jako czas wartość 0. Należy jednak pamiętać, że w takim przypadku znacznie obniżamy bezpieczeństwo połączenia umożliwiając (dając czas) złamanie klucza szyfrującego i tym samym podsłuchanie transmisji.

I to właściwie wszystko co potrzebujemy aby uruchomić serwis. Warto jednak zainteresować się jeszcze innymi parametrami, których nie ma aktualnie w pliku konfiguracyjnym. To co może nas szczególnie zainteresować to parametry *AllowGroups,AllowUsers,DenyGroups,DenyUsers*. Parametry te pozwalają nam bardzo precyzyjnie określić kto może korzystać z naszego serwisu. W przypadku parametrów *Allow** logować się mogą tylko użytkownicy/grupy użytkowników znajdujący się na liście. W przypadku parametrów *Deny** sytuacja jest odwrotna, logować się mogą wszyscy z wyjątkiem wymienionych na liście. Zatem, jeśli chcemy aby prawo korzystania z serwisu mieli tylko użytkownicy *tuptus* i *test* to musimy do pliku konfiguracyjnego dodać linię

```
AllowUsers tuptus test
```

Po tych wszystkich zabiegach należy zadbać o to aby demon wprowadzone zmiany przyjął do wiadomości. Możemy oczywiście wykonać restart demona ale to dość brutalne posunięcie. W przypadku demona sshd można wykorzystać rozwiązanie łagodniejsze

```
[root@dreptak root]# service sshd reload
Ponowne ładowanie sshd:                                    [   OK   ]
[root@dreptak root]#
```

Skoro mamy już przygotowany serwer to możemy zająć się użytkowaniem serwisu. Zaczniemy od przetestowania serwera:

```
[tuptus@dreptak tuptus]$ ssh tuptus@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 36:c3:23:49:84:0f:66:3d:32:40:cc:41:f9:ae:5c:30.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
tuptus@localhost's password:
[tuptus@dreptak tuptus]$
```

Jak widzisz wykorzystałem tu skrótową formę polecenia `ssh`. Pełna wersja wyglądałaby następująco:

```
[tuptus@dreptak tuptus]$ ssh localhost -l tuptus
tuptus@localhost's password:
[tuptus@dreptak tuptus]$
```

Zapewne zauważyłeś, że tym razem nie pojawiła się informacja o problemach z autentykacją serwera. O co chodzi z tą autentykacją? Otóż na samym początku nawiązywania połączenia

serwer przedstawia się klientowi przesyłając swój klucz a właściwie jego "odcisk" nazywany `fingerprint` (dosłownie odcisk palca). Ponieważ było to pierwsze połączenie to klient zakomunikował nam, że takiego serwera nie zna i zapytał czy rzeczywiście chcemy się z nim połączyć. Jeśli odpowiesz "yes" to nazwa serwera oraz jego klucz publiczny zostaną zapisane do pliku `~/.ssh/known_hosts`. Przy kolejnej próbie połączenia następuje porównanie odcisku dostarczonego przez serwer z odciskiem klucza publicznego, który wcześniej zapamiętano i jeśli te informacje się zgadzają to następuje połączenie. W przeciwnym przypadku pojawia się dość groźnie wyglądający komunikat:

```
[tuptus@dreptak tuptus]$ ssh localhost
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
c7:3d:f4:33:dc:ca:eb:73:2c:5a:f9:54:1b:a9:91:5a.
Please contact your system administrator.
Add correct host key in /home/tuptus/.ssh/known_hosts to get rid of this message.
Offending key in /home/tuptus/.ssh/known_hosts:2
RSA host key for localhost has changed and you have requested strict checking.
Host key verification failed.
[tuptus@dreptak tuptus]$
```

Oczywiście w tym przypadku sprawa jest prosta. Na potrzeby kursu zmieniłem klucze serwera i stąd taki komunikat. Wystarczy otworzyć plik `~/.ssh/known_hosts`, skasować linię z kluczem serwera sprawiającego problemy i połączyć się ponownie - nowy klucz zostanie dopisany do pliku. W rzeczywistym świecie może to jednak być znacznie poważniejsza sprawa gdyż oznaczać może, że ktoś próbuje się podszywać pod serwer stosując atak typu "man in the middle".

Uwaga. Jeśli nie masz pewności co do zmian klucza na serwerze to koniecznie skontaktuj się z administratorem tego serwera i wyjaśnij sprawę.

Może zauważyłeś, że w tym ostatnim przypadku w ogóle nie podawałem nazwy użytkownika. Okazuje się, że nie ma takiej potrzeby gdyż klient ssh w naszym systemie umie sobie sprawdzić kto z niego korzysta i samodzielnie prześle nazwę użytkownika do serwera. Wprawdzie nie wszystkie aplikacje klienckie umieją zachować się tak przyjaźnie ale klient zawarty w pakiecie openssh jest naprawdę dobry. Jeżeli na serwerze wszystko działa to proponuję przejść teraz do innego komputera i sprawdzić czy działa połączenie. Właściwie nie powinno być problemów ale ... np. błędna konfiguracja ogniomurka na serwerze może nam przysporzyć wiele godzin poszukiwania błędów (sshd domyślnie nasłuchuje na porcie 22/tcp).

Pokazane do tej pory metody połączenie dają nam funkcjonalność analogiczną do usługi telnet tyle, że połączenie jest szyfrowane. Zajmiemy się teraz innymi możliwościami wykorzystania serwisu. Często zdarza się, że potrzebujemy wykonać jakieś jedno polecenie na zdalnym hoście. W systemach uniksowych istnieją dwa polecenia umożliwiające wykonanie tego zadania: `rsh` i `rexec`. Ze względu jednak na problemy z bezpieczeństwem ich wykorzystanie ogranicza się obecnie do bardzo specyficznych przypadków, na naszym serwerze nawet nie będziemy instalować demonów obsługujących te polecenia. Mamy jednak polecenie `ssh`. Jeśli na do polecenia wykorzystywanego do tej pory dodamy na końcu jakieś polecenie powłoki to polecenie to zostanie wykonane i nastąpi powrót to "macierzystego" systemu.

```
[tuptus@dreptak tuptus]$ ssh dreptak.tupward.pl "ls -l /var/www"
tuptus@localhost's password:
total 24
drwxr-xr-x   2 root   root   4096 Apr 23 14:04 cgi-bin
drwxr-xr-x   3 root   root   4096 Jul 29 18:57 error
drwxr-xr-x   7 root   root   4096 Oct  2 13:39 html
drwxr-xr-x   3 root   root   4096 Jul 29 19:08 icons
drwxr-xr-x  13 root   root   4096 Aug  6 20:29 manual
[tuptus@dreptak tuptus]$
```

Co ciekawe, standardowe wyjście takiego polecenia znajduje się na lokalnej maszynie:

```
[tuptus@dreptak tuptus]$ ssh dreptak.tupward.pl "ls -l /var/www" | grep html
tuptus@localhost's password:
drwxr-xr-x   7 root   root   4096 Oct  2 13:39 html
[tuptus@dreptak tuptus]$
```

a to oznacza, ze możemy tego typu wywołanie wykorzystać w skryptach działających na lokalnej maszynie. Tak, wykorzystać można ale co zrobić z pytaniem o hasło? Z tym problemem też sobie poradzimy tyle, że w następnej części.

Kolejną funkcjonalnością jaką się zajmiemy jest możliwość kopiowania plików między hostami. W lokalnym systemie do tego celu wykorzystujemy polecenie `cp`. Jeśli jednak trzeba skopiować plik na zdalną maszynę to polecenie to nie zadziała. W systemach uniksowych można znaleźć polecenie `rcp` ale tak jak wcześniej opisywane polecenia `rsh` i `rexec` jego wykorzystywanie stwarza pewne zagrożenia systemu bezpieczeństwa. Dlatego w pakiecie openssh znajduje się polecenie `scp`. Nie będę omawiał tutaj szczegółów tego polecenia gdyż na ogół wykorzystuje się jedynie jego podstawową formę. Jeśli chcemy z lokalnej maszyny skopiować jakiś plik na maszynę zdalną to polecenie przyjmuje następującą formę

```
[tuptus@test tuptus]$ scp rob/hcl.sql tuptus@dreptak.tupward.pl:/home/tuptus/
tuptus@dreptak.tupward.pl's password:
hcl.sql                                   100%   14KB 683.2KB/s   00:00
[tuptus@test tuptus]$ ssh dreptak.tupward.pl "ls -l hcl*"
-rw-------  1 tuptus tuptus 14453 paź 16 16:35 hcl.sql
```

Oczywiście nazwy katalogów i plików z tego przykładu są właściwe dla mojego środowiska i u Ciebie prawdopodobnie nie istnieją.

Z ciekawszych opcji tego polecenia należy wymienić opcję `-r`. Dzięki zastosowaniu tej opcji możemy przy pomocy polecenia `scp` kopiować rekurencyjnie całe katalogi wraz z podkatalogami.

W przykładzie powyżej kopiowaliśmy plik lokalny do katalogu na zdalnym hoście ale oczywiście nic nie sto na przeszkodzie aby sytuacje odwrócić i kopiować pliki ze zdalnego hosta na lokalną maszynę. Ten temat zostawiam Tobie do samodzielnego opracowania.

13.2 Autoryzacja na bazie kluczy

Jak już wcześniej wspomniałem często zachodzi potrzeba połączenia ze zdalnym hostem bez podawania hasła. Tworzenie konta z pustym hasłem jest z oczywistych względów wykluczone więc trzeba sobie jakoś inaczej poradzić. Rozwiązaniem tego problemu może być zastosowanie kluczy RSA lub DSA. Zamiast odpowiadać na pytanie o hasło możemy przekazywać do serwera fingerprint generowany z naszego klucza prywatnego a serwer dokonuje porównanie tego odcisku z kluczem publicznym znajdującym się na serwerze. Jednak nie chodzi tu o klucze hosta, o których była wcześniej mowa. Musimy wygenerować sobie swoje osobiste klucze. Do tego celu wykorzystamy polecenie `ssh-keygen`

```
[tuptus@dreptak tuptus]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tuptus/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tuptus/.ssh/id_rsa.
Your public key has been saved in /home/tuptus/.ssh/id_rsa.pub.
The key fingerprint is:
5f:86:b8:8d:03:09:e6:64:a5:7b:7e:5f:01:48:0e:c5 tuptus@dreptak.tupward.pl
[tuptus@dreptak tuptus]$
```

Dość istotne jest tutaj pytanie *Enter passphrase*. Chodzi tu o hasło do klucza a nie o Twoje hasło w systemie. Ponieważ nam zależy na tym aby korzystać z możliwości logowania bez konieczności podawania hasła to tutaj wciskamy ENTER . Gdybyśmy wprowadzili hasło to przy każdej próbie wykorzystania klucza pojawiłoby się pytanie o to hasło.

Po stronie klienta mamy sprawę załatwioną i teraz musimy przygotować konto na serwerze. W tym celu musimy na serwer "przetransportować" plik z kluczem publicznym czyli w tym wypadku plik `~/.ssh/id_rsa.pub`. Musimy przy tym zadbać o to żeby nasz klucz publiczny nie wpadł w niepowołane ręce. Kiedy ten plik już znajdzie się na serwerze musimy jego zawartość przenieść do pliku `~/.ssh/authorized_keys`. Jeśli jest to pierwszy klucz to możemy wykonać zwykłe kopiowanie pliku `id_rsa.pub` na `authorized_keys`. Jeśli jednak jest to już kolejny klucz to sprawa jest odrobinę trudniejsza, musimy operować na zawartości plików.

```
[tuptus@dreptak tuptus]$ cat id_rsa.pub >> .ssh/authorized_keys
[tuptus@dreptak tuptus]$
```

Teraz wypadałoby jeszcze przetestować połączenia ale zapewniam Cię, że takie rozwiązanie działa.

13.3 Tunelowanie usług

Zajmiemy się teraz tunelowaniem połączeń. Najprostszym do wytłumaczenia jest tunel wykorzystywany dla połączeń XWindow. Wystarczy, że w pliku konfiguracyjnym serwera ustawimy opcję *X11Forwarding* na "yes". Teraz restart serwera i już mamy możliwość uruchamiania aplikacji graficznych na zdalnym hoście ale oglądać je będziemy na maszynie lokalnej. Ponieważ wykorzystanie tej techniki w sieci lokalnej jest raczej mało przydatne a w internecie niepraktyczne (ze względu na szybkość transmisji) to nie będę się tym tematem zajmował dokładniej.

Znacznie ciekawsze jest tunelowanie "dowolnych" usług. Temat omówimy na dwóch przykładach: odbieranie poczty ze zdalnego serwera POP3 oraz korzystanie z repozytorium Subversion przez ogniomurek. Dlaczego tunelować połączenia POP3? To dobre pytanie. Przecież jest możliwość uruchomienia usługi SPOP3. Tak, istnieje taka możliwość ale ...

- nie wszystkie aplikacje klienckie obsługują ten protokół,
- niewielu administratorów uruchamia SPOP3 wychodząc z założenia, że nie ma sensu szyfrować lokalnie tego co przez internet wędruje otwartym tekstem
- znacznie częściej mamy możliwość przejścia przez ogniomurek z protokołem SSH niż POP3

Ostatni punkt wymaga pewnego komentarza. W wielu wypadkach, zwłaszcza w sieciach korporacyjnych, użytkownicy otrzymują skrzynki pocztowe i równocześnie blokowany jest dostęp do zewnętrznych serwerów POP3 (port 110). Jednym z argumentów za takim postępowaniem jest ochrona sieci korporacyjnej przed wirusami. Administrator takiej sieci uruchamia odpowiednie

zabezpieczenia, które mają zapobiec przedostawaniu się do LAN'u różnego "robactwa" i przyjmuje się, że poczta, która przeszła przez takie filtry jest bezpieczna. Takiego założenia nie można przyjąć w przypadku serwerów pocztowych poza siecią lokalną.

Uwaga. Pamiętaj, że stosowanie tunelowania w celu ominięcia zabezpieczeń sieci poczynionych przez administratora może być przyczyną poważnych problemów. Każdy przypadek stosowania tunelowania należy uzgadniać z administratorem i uzyskać jego zgodę na takie działania.

Wiemy już co chcemy osiągnąć więc możemy przystąpić do pracy. Mam nadzieję, że przeczytałeś wcześniejszy rozdział dotyczący kluczy autoryzacyjnych. W poniższym opisie zakładam, że klucze takie już masz przygotowane i nie będzie potrzeby podawania hasła przy nawiązywaniu połączenia. Na potrzeby poniższego opisu przyjmuję ponadto, że posiadasz "pełne" konto na hoście *host.domena.pl*, który udostępnia serwis POP3. Ponieważ w sieci lokalnej jak również na lokalnym hoście może być również uruchomiona usługa POP3 a także ze względu na uprawnienia w systemie (tylko root może zarządzać portami poniżej 1024) nasz tunel będzie udostępniał POP3 na porcie 1110. Uwzględniając powyższe założenia możemy zestawić tunel:

```
[tuptus@dreptak mnt]$ ssh -f -N -L 1110:localhost:110 host.domena.pl
[tuptus@dreptak mnt]$ netstat -an | grep 1110
tcp        0      0 127.0.0.1:1110          0.0.0.0:*               LISTEN
```

Jak widać połączenie zostało zestawione. Możemy teraz uruchomić klienta pocztowego i skonfigurować go tak aby odbierał pocztę z hosta *localhost* ale na porcie 1110 i sprawdzić czy działa odbieranie poczty.

Teraz kilka słów wyjaśnienia jak to połączenia działa. Zacznijmy od dwóch pierwszych opcji. Opcja `-f` powoduje automatyczne przeniesienie procesu `ssh` "do podziemia" inaczej mówiąc proces dalej działa w tle a my odzyskujemy dostęp do shella. Opcja `-N` zabrania wykonywania poleceń na zdalnym hoście. Opcja jest o tyle istotna, że domyślnie ssh udostępnia shell na zdalnym hoście a w tym przypadku tak funkcjonalność jest zbędna i wręcz szkodliwa gdyż musieli byśmy ręcznie podawać polecenia protokółu POP3.

Te dwie opcje stanowiły "przygotowanie" do tego co teraz ma nastąpić. Pozostałą część polecenia musimy rozpatrywać w całości. Zaczniemy od końca. Jak widzisz jest to nazwa hosta na którym masz konto i działa tam serwer ssh. Natomiast zapis `-L 1110:localhost:110` jest właściwym zestawieniem tunelu. Opcja `-L` informuje o tym, że tunel ma być otwierany w kierunku lokalnego hosta. To dość istotna informacja bo za moment zajmiemy się otwieraniem tunelu w kierunku zdalnego hosta. I teraz to co jest najważniejsze czyli określenie parametrów tunelu. W naszym przypadku podajemy kolejno: numer portu na lokalnej maszynie, nazwę/IP hosta z którego będziemy "przenosić" port i na końcu numer "przenoszonego" portu. Nazwa *localhost* może tu być myląca. Sztuczka polega na tym, że podajemy nazwę hosta względem hosta, na którym działa serwer ssh. W naszym przypadku serwer ssh i POP3 działają na jednej maszynie więc możemy stosować taki zapis. Jeśli na hoście *host.domena.pl* nie ma serwera POP3 lub chcielibyśmy wydostać się tunelem z sieci lokalnej w celu skorzystania z innego serwera POP3 to polecenie tworzące tunel. mógłby mieć np. taką postać:

```
[tuptus@dreptak mnt]$ ssh -f -N -L 1110:poczta.onet.pl:110 host.domena.pl
[tuptus@dreptak mnt]$
```

Pamiętaj jednak, że szyfrowane będzie tylko połączenie pomiędzy lokalną maszyną i serwerem ssh natomiast dalej, czyli od *host.domena.pl* do *poczta.onet.pl* połączenie już nie będzie szyfrowane.

Niestety nie ma możliwości eleganckiego zamknięcia takiego tunelu. Musimy odszukać proces ssh na liście procesów i wysłać mu odpowiedni sygnał powodujący zakończenie działania procesu.

```
[tuptus@dreptak tuptus]$ ps -uax | grep ssh
root       3233  0.0  0.1  5112  740 ?        S    Oct21   0:00 /usr/sbin/sshd
tuptus    19400  0.0  0.1  3956  540 ?        S    12:48
0:00 /usr/bin/ssh-agent /usr/share/apps/switchdesk/Xclients.icewm
tuptus    21764  0.0  0.3  4916 1688 ?        S    14:01
0:00 ssh -f -N -L 1110:localhost:110 host.domena.pl
tuptus    21768  0.0  0.1  4636  540 pts/1    S    14:01   0:00 grep ssh
[tuptus@dreptak tuptus]$ kill 21764
Killed by signal 15.
```

Oczywiście takie postępowanie nie rozwiązuje całościowo tematu korzystania z poczty gdyż POP3 to jedynie odbieranie poczty. Do wysyłania wiadomości potrzebny jest serwer SMTP. Możemy do tego celu wykorzystać działający w sieci lokalnej serwer lub zestawić kolejny tunel tym razem na port 25.

Wykorzystanie tunelu ssh na potrzeby Subversion wygląda już znacznie lepiej. Serwer svn wszystkie operacje wykonuje korzystając z portu 3690. Zestawiając tunel otrzymujemy zatem pełną funkcjonalność serwera przeniesioną na lokalny port a ponieważ jest to port "wysoki" to nie ma problemu z wykorzystaniem go przez zwykłego użytkownika. Biorąc pod uwagę założenia jak w poprzednim przykładzie możemy zatem zestawić następujący tunel:

```
[tuptus@dreptak mnt]$ ssh -f -N -L 3690:localhost:3690 host.domena.pl
[tuptus@dreptak mnt]$ netstat -an | grep 3690
tcp        0      0 127.0.0.1:3690          0.0.0.0:*               LISTEN
```

Teraz możemy pobrać repozytorium subversion wpisując komendę:

```
[tuptus@dreptak mnt]$ svn co svn://localhost/Aurox .
[tuptus@dreptak mnt]$
```

Oczywiście repozytorium *Aurox* to tylko przykład, w taki właśnie sposób łącze się z repozytorium zawierającym niniejszy podręcznik.

Jeszcze raz przypominam, że tego typu działania omijają zabezpieczenia ogniomurka i należy je uzgadniać z administratorem sieci.

Zajmiemy się teraz nieco innym rozwiązaniem tunelowania usług a mianowicie udostępnianiem lokalnego portu na zdalnej maszynie. Postępowanie jest analogiczne jak we wcześniejszych przykładach z tą różnicą, że zamiast opcji `-L` stosujemy `-R`.

Uwaga. Udostępniając port z hosta w sieci lokalnej (chronionej) na hoście poza tą siecią może prowadzić do poważnego naruszenia polityki bezpieczeństwa. Rozwiązanie to podaje tylko w celach edukacyjnych. Jeśli wykorzystasz je w praktyce to robisz to na własną odpowiedzialność. **Koniecznie uzgodnij to z administratorem**.

Nie trudno sobie wyobrazić sytuację gdy na maszynie w sieci lokalnej mamy uruchomiony jakiś serwis ale chcielibyśmy aby dostępny był spoza sieci lokalnej. Oczywiście najlepszym rozwiązaniem tego problemu byłaby modyfikacja ogniomurka w taki sposób aby zapytania trafiające na określony port ogniomurka były przekierowywane na nasz host. Jednak takie działanie nie zawsze jest możliwe do zrealizowania. Właściwie moglibyśmy tą formę tunelowania przećwiczyć na Subversion ale ponieważ pracuje ono na wysokich portach to zadanie to byłoby trywialne. Dlatego zajmiemy się udostępnieniem telnetu na hoście poza siecią lokalną. **Pamiętaj, że to potencjalnie niebezpieczne postępowanie.** Aby wykonać to ćwiczenie potrzebować będziemy działający na lokalnym hoście serwis telnetd. Jeśli jeszcze nie masz zainstalowanego tego pakietu to zainstaluj go, wykorzystując polecenie `ntsysv` zaznacz *telnetd* i przeładuj serwis *xinetd* (telnetd nie jest samodzielnym demonem i działa za pośrednictwem superdemona xinetd).

Skoro już mamy przygotowany serwer telnet na lokalnej maszynie to możemy przystąpić do zestawienia tunelu:

```
[tuptus@dreptak tuptus]$ ssh -f -N -R 1123:localhost:23 host.domena.pl
[tuptus@dreptak tuptus]$
```

Jak widzisz kolejność parametrów tunelu nieco się zmieniła. Na pierwszym miejscu mamy podany numer portu, który ma być otwarty na zdalnym hoście (w tym przypadku port 1123) natomiast numer lokalnego portu podajemy na ostatnim miejscu. Teraz wypadałoby sprawdzić czy faktycznie tunel został zestawiony. Logujemy się zatem na *host.domena.pl* i wykonujemy testy:

```
[tuptus@dreptak tuptus]$ ssh host.domena.pl
[tuptus@host tuptus]$ netstat -an | grep :1123
tcp        0      0 127.0.0.1:1123          0.0.0.0:*               LISTEN
[tuptus@host tuptus]$ telnet localhost 1123
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Linux pod opieką Tuptusia
Nieautoryzowany dostęp zabroniony
Wszelkie zdarzenia są logowane!!!
================================


login: tuptus
Password:
Last login: Sun Oct 24 12:33:17 from localhost.localdomain
You have mail.
[tuptus@dreptak tuptus]$
```

Z powyższego wynika, że zestawiony tunel działa poprawnie.

Może zastanawiasz się jaki jest sens zestawiania tunelu do lokalnej maszyny skoro na niej pracujesz. Okazuje się, że jest sens. Możesz w ten sposób wpuścić na swoją maszynę kolegę, który ma konto na *host.domena.pl* a nie ma fizycznego dostępu do maszyny znajdującej się w Twojej sieci lokalnej. Innym pomysłem jest wykorzystanie polecenia tunelującego w skrypcie wywoływanym przez procmaila po otrzymaniu odpowiedniej wiadomości. Możesz w ten sposób połączyć się z hostem w sieci lokalnej (np. w firmie) w czasie gdy jesteś poza tą firmą. Wystarczy, że prześlesz do Twojej maszyny odpowiednią wiadomość e-mail. Mam nadzieję, że teraz rozumiesz jakie niebezpieczeństwa niesie ze sobą korzystanie z tego rodzaju tunelowaniem — korzystaj rozważnie.

Rozdział 14

Serwer pocztowy

Obok stron www serwis pocztowy jest najistotniejszą usługą serwera. W rozdziale tym zajmiemy się konfiguracją takiego serwera. W Auroksie mamy do dyspozycji dwa serwery do wyboru — sendmail i postfix. My będziemy konfigurować sendmaila, jako że jest to klasyka i występuje we wszystkich systemach uniksowych. Wiele osób twierdzi, że sendmail jest bardzo trudnym do skonfigurowania systemem. Jeśli chcemy zgłębić wszystkie tajniki jego działania, to rzeczywiście jest to trudne zadanie. My jednak nie będziemy się zagłębiać w jakieś egzotyczne opcje konfiguracyjne, ale zajmiemy się praktycznym wykorzystaniem tego, co nam udostępnia ten kombajn.

14.1 Podstawowa konfiguracja serwera sendmail

Zaczynamy zatem przygodę z sendmailem. Jeśli jakimś cudem nie został on zainstalowany przy instalacji systemu, to trzeba go będzie zainstalować. Zanim jednak przystąpimy do tego zadania sprawdźmy, czy jest to konieczne.

```
[tuptus@dreptak tuptus]$ rpm -qa | grep sendmail
sendmail-8.12.10-1.1.1
sendmail-cf-8.12.10-1.1.1
[tuptus@dreptak tuptus]$
```

Jak widzisz na mojej maszynie już jest sendmail zainstalowany. Ten drugi pakiet (`sendmail-cf`) potrzebny nam będzie do zmian w konfiguracji sendmaila. Zatem jeśli u Ciebie tych pakietów nie ma, to zainstaluj je najlepiej wykorzystując apta.

Kolejnym krokiem jest uruchomienie demona. Ponownie zaczniemy od sprawdzenia, czy demon działa, a jeśli nie, to go uruchomimy i sprawdzimy czy działa.

```
[root@dreptak root]# service sendmail status
sendmail jest zatrzymany
[root@dreptak root]# service sendmail start
Uruchamianie sendmail:                                     [  OK  ]
Uruchamianie klienta sm:                                   [  OK  ]
[root@dreptak root]#
[root@dreptak mail]# service sendmail status
sendmail (pid 4643 4634) jest uruchomiony...
[root@dreptak mail]# netstat -atn | grep 25
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
[root@dreptak mail]#
```

Jak widzimy demon sendmail został uruchomiony i nasłuchuje na porcie 25 (smtp). Jest tylko jeden mały problem : nasłuchuje na interfejsie loopback. Czyli można do demona wysyłać pocztę

wyłącznie z lokalnych kont. Czasami taka konfiguracja jest nawet pożądana. Nasz host umie teraz wysyłać pocztę lokalnych użytkowników, ale nie przyjmie żadnej przesyłki z innego hosta. Dla stacji roboczej idealne rozwiązanie, tylko że my chcemy mieć serwer pocztowy, a nie stację roboczą.

Przechodzimy zatem do drugiego etapu, czyli konfiguracji serwera. Pliki konfiguracyjne sendmaila znajdują się w katalogu `/etc/mail`. Nas w tym momencie najbardziej interesuje plik `sendmail.mc`.

Uwaga. Przed rozpoczęciem zmian w konfiguracji radziłbym wykonać kopię plików sendmail.mc, sendmail.cf, submit.mc i submit.cf

---

Otwieramy zatem do edycji wspomniany plik konfiguracyjny i ... Wiem, wiem. Format tego pliku może przerażać, ale już za chwilkę będziesz się czuł jak w domku. Pierwsza sprawa to tajemniczy ciąg znaków `dnl` wielokrotnie powtarzający się w tym pliku. Tym ciągiem oznacza się komentarze. Tak jak w większości plików komentarz zaczyna się od `#`, tak tutaj początek komentarza oznacza się `dnl` i obejmuje on cały dalszy tekst aż do końca linii.

Ale zajmijmy się poprawkami. Odszukaj linię :

```
DAEMON_OPTIONS(‘Port=smtp,Addr=127.0.0.1, Name=MTA’)
```

i zakomentuj ją wstawiając na początku `dnl`. Teraz zapisz zmiany i ... Powoli. Plik `sendmail.mc` tak naprawdę nie jest plikiem konfiguracyjnym sendmaila. Jest on swego rodzaju skryptem, który należy przetworzyć programem `m4` do postaci zrozumiałej dla demona i zapisać w pliku `sendmail.cf`. Aby Ci ułatwić to zadanie w katalogu `/etc/mail` umieszczono plik `Makefile`, który zawiera wszystko, co jest potrzebne do aktualizacji plików konfiguracyjnych. Zawsze po wprowadzeniu zmian w którymkolwiek pliku w katalogu `/etc/mail` wykonuj polecenie `make` i dopiero po jego wykonaniu restartuj sendmaila.

```
[root@dreptak mail]# vim sendmail.mc
[root@dreptak mail]# make
[root@dreptak mail]# service sendmail restart
Zamykanie sendmail:                                        [  OK  ]
Zatrzymywanie sm-client:                                   [  OK  ]
Uruchamianie sendmail:                                     [  OK  ]
Uruchamianie klienta sm:                                   [  OK  ]
[root@dreptak mail]# netstat -atn | grep 25
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
[root@dreptak mail]#
```

Wygląda na to, że jest dobrze. Zapis 0.0.0.0:25 oznacza, że nasz serwer nasłuchuje na porcie 25 na wszystkich dostępnych interfejsach.

To jednak nie wszystko. Aby nasz serwer odbierał pocztę przeznaczoną dla niego, to trzeba jeszcze zmodyfikować plik `access` nazywany często *access listą* lub *listą dostępową*. Otwieramy zatem plik do edycji i modyfikujemy go do postaci :

```
localhost.localdomain           RELAY
localhost                       RELAY
127.0.0.1                       RELAY
dreptak.tupward.pl          OK
```

Tutaj trzeba się chwilkę zatrzymać. Pierwsze trzy wpisy dotyczą połączeń lokalnych. Parametr `RELAY` oznacza zezwolenie na przekazywanie poczty. Oznacza to, że poczta wysłana z jednego z tych adresów może być przekazana do innego serwera pocztowego. Temat ten omówimy dokładniej w dalszej części rozdziału. Dwie następne pozycje oznaczają zezwolenie na przyjmowanie

poczty kierowanej na podane adresy. To chyba nie wymaga wyjaśnienia. Istnieją jeszcze inne opcje, ale o nich będziemy mówić przy okazji omawiania problematyki spamu.

Do kompletu konfiguracji należy jeszcze zmodyfikować plik `local-host-names`. Plik ten powinien zawierać wszystkie nazwy, które identyfikują lokalny system. W naszym przypadku plik ten powinien wyglądać następująco :

```
dreptak.tupward.pl
tupward.pl
localhost.localdomain
```

Każda wiadomość przychodząca do naszego serwera i posiadająca w polu adresu host/domenę inną niż wymieniona w tym pliku traktowana będzie jako nielokalna i wymagająca przesłania do innego serwera. W tym momencie następuje sprawdzenie adresu nadawcy i sprawdzenie, czy dla tego nadawcy zezwolono na przekazywanie wiadomości (`RELAY`) i przesłanie jej jeśli test dał wynik pozytywny. W przeciwnym wypadku wiadomość zostaje odrzucona, a nadawca otrzymuje odpowiedni komunikat.

Teraz jeszcze odświeżenie konfiguracji, restart demona i możemy przystąpić do testów. Zaczniemy od wysłania wiadomości do samego siebie. Do tego celu wykorzystamy polecenie `mail`.

```
[tuptus@dreptak tuptus]$ mail tuptus
Subject: test
test
.
Cc:
[tuptus@dreptak tuptus]$
```

Teraz odczekajmy chwilkę i sprawdzamy czy wiadomość została dostarczona.

```
[tuptus@dreptak tuptus]$ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/tuptus": 1 message 1 new
>N  1 tuptus@dreptak.tupwa  Thu Jul 15 23:43  16/673   "test"
& 1
Message 1:
From tuptus@dreptak.tupward.pl  Thu Jul 15 23:43:04 2004
Date: Thu, 15 Jul 2004 23:43:04 +0200
From: tuptus@dreptak.tupward.pl
To: tuptus@dreptak.tupward.pl
Subject: test

test

& d 1
& q
[tuptus@dreptak tuptus]$
```

Jak widać lokalna poczta działa. Teraz należy sprawdzić przesyłanie wiadomości do innych serwerów oraz jej odbieranie. Możemy ponownie wykorzystać polecenie `mail`, przy czym teraz jako adres należy podać pełną nazwę konta np. `tuptus@jakas.domena.pl`. Jeśli wszystko zrobiliśmy poprawnie, to po kilku minutach wiadomość powinna pojawić się na podanym koncie. Jeśli jednak nie będzie się pojawiać, to należy sprawdzić, czy nie otrzymaliśmy jakiegoś "zwrotu". Taka wiadomość zwrotna będzie zawierała informację o powodzie niepowodzenia. W tym wypadku informacja o niepowodzeniu jest jak najbardziej poprawna. Wynika ona z faktu, że internet nic nie wie o domenie *tupward.pl* ani o hoście *dreptak* z tej domeny. Kiedy nasz host łączy się z serwerem pocztowym musi się najpierw "przedstawić", a serwer sprawdzi sobie, czy taki host jest znany przez DNS. Jeśli DNS go nie zna, to odrzuci takie połączenie. My też coś takiego zrobimy, ale za moment. Aby sobie poradzić z tym problemem musimy przekonać sendmaila,

żeby przedstawiał się nazwą internetową, a nie tą, którą zwraca polecenie `hostname`. W tym celu do pliku `sendmail.mc` dopisujemy linijkę :

```
define(‘confDOMAIN_NAME’,‘host81-225.tvk.torun.pl’)dnl
```

Oczywiście ja wpisałem nazwę mojego hosta, ale Ty powinieneś podać nazwę odpowiednią dla Twojego hosta. Teraz jeszcze `make` i restart sendmaila i wysyłanie poczty powinno działać. Kiedy wiadomość dotrze na nasze zdalne konto, to spróbujmy na taką wiadomość odpowiedzieć. Najprawdopodobniej wiadomość dotrze do naszego serwera.

Niby wszystko działa, ale wygląda nieco nieciekawie. Zapewne zauważyłeś, że większość adresów pocztowych ma postać *@domena* tymczasem nasz adres ma postać *@host.domena*. Aby to zmienić będzie potrzebna współpraca z naszym ISP. Po pierwsze musimy mieć własną domenę. Przypuśćmy, że mamy już taką domenę np. *szkola.isp.pl*. Teraz trzeba poinformować świat do jakiego komputera ma trafiać poczta przeznaczona dla tej domeny. Komputer ten nazywany jest *“mail exchanger”*. Oczywiście komputer taki musi być również wpisany do DNS. Nie będziemy się tutaj zajmować modyfikacją strefy DNS, to jest rola ISP. Dla nas istotna jest jedynie informacja, że dla naszej domeny *szkola.isp.pl* musi zostać określony rekord MX wskazujący na nasz serwer pocztowy. Rekord taki będzie wyglądał mniej więcej tak :

```
szkola.isp.pl.  MX  10  dreptak.szkola.isp.pl.
szkola.isp.pl.  MX  20  mail.isp.pl.
```

Najważniejszy jest ten pierwszy rekord. Drugi nie musi istnieć, ale jeśli dobrze żyjesz z ISP, to się z całą pewnością przyda. Czasami może się zdarzyć, że nasz serwer z jakiegoś powodu nie będzie mógł przyjąć poczty. Dzięki temu drugiemu rekordowi poczta przeznaczona dla naszego serwera wyląduje na serwerze ISP, a w momencie, gdy nasz serwer wróci do sieci zostanie przesłana do nas. O kolejności wybierania serwerów pocztowych decyduje liczba występująca za MX — im niższa, tym wyższy priorytet ma dany serwer.

Ale odeszliśmy nieco od tematu. Załóżmy, że z ISP już wszystko mamy uzgodnione i DNS jest zaktualizowany. Zmodyfikujmy zatem konfigurację naszego serwera tak, aby wiadomości wychodzące miały postać *konto@szkola.isp.pl* i wiadomości adresowane w ten sposób były przyjmowane. Pierwszym krokiem jest modyfikacja opcji, którą przed chwilą dopisywaliśmy do `sendmail.mc`.

```
define(‘confDOMAIN_NAME’,‘szkola.isp.pl’)dnl
```

Ten wpis zapewnia wysyłanie poczty w odpowiedniej postaci. Teraz jeszcze otwieramy plik `local-host-names` i wpisujemy tutaj nazwy hostów i domen, dla których będziemy przyjmować pocztę, po jednej pozycji w linii.

```
dreptak.tupward.pl
localhost.localdomain
szkola.isp.pl
dreptak.szkola.isp.pl
```

I dopisujemy odpowiednie linie do pliku access :

```
szkola.isp.pl     OK
```

Teraz jak zwykle odświeżenie konfiguracji, restart demona i możemy przystąpić do testowania naszej konfiguracji.

Możemy już odbierać pocztę i wysyłać, ale tylko z lokalnego hosta czyli z serwera. To nie jest to. Przecież użytkownicy będą chcieli korzystać z klientów pocztowych takich jak kmail, mozilla itp., a te działają na zdalnych maszynach. Zastanówmy się jak dodać taką funkcjonalność. Aplikacje klienckie działają tak jak nasz serwer na samym początku, czyli wysyłają pocztę do

wskazanego serwera SMTP mając nadzieję, ze ten będzie wiedział co dalej z taką przesyłką zrobić. Jeśli taka przesyłka dotrze do naszego serwera, to powinien ją przekazać dalej do serwera właściwego według adresata. Działanie takie nazywa się *RELAY*, dosłownie sztafeta. Oczywiście nie możemy dopuścić do sytuacji, gdy dowolny użytkownik internetu może za pośrednictwem naszego serwera przesyłać pocztę. Serwer działający w ten sposób nazywany jest *open relay* i jest bezwzględnie tępiony. Przyczyna? Serwery tego typu to wspaniała pożywka dla spamerów. Więcej chyba nie trzeba tłumaczyć. Zacznijmy zatem delikatnie. Zezwolimy na wysyłanie poczty klientom z naszej sieci lokalnej. Otwieramy do edycji plik `access` i modyfikujemy go do postaci:

```
localhost.localdomain           RELAY
localhost                       RELAY
127.0.0.1                       RELAY
szkola.isp.pl                   OK
tupward.pl                      RELAY
192.168.1                       RELAY
```

Powyższe zapisy należy rozumieć następująco :

- przyjmujemy pocztę kierowaną do domeny *szkola.isp.pl*
- przekazujemy pocztę pochodzącą z lokalnej maszyny
- przekazujemy pocztę pochodzącą z hostów z domeny *tupward.pl*
- przekazujemy pocztę pochodzącą z hostów, których IP zaczyna się od `192.168.1`

Na dzień dzisiejszy możemy przyjąć taką konfigurację, ale ... Wprawdzie klasa adresowa 192.168... to grupa adresów prywatnych, ale jest to tylko umowa i nic nie stoi na przeszkodzie, żeby jakiś spamer upublicznił taką sieć i wykorzystał nasz serwer do wysyłki. Takie postępowanie ma krótkie nóżki, ale zanim takiemu bandycie odetną dostęp do sieci, to nasz serwer trafi do różnych filtrów i będziemy mieli spore problemy.

Podobnie sprawa wygląda z relayowaniem dla domeny *tupward.pl*. Obecnie domena taka nie istnieje w internecie, ale w każdej chwili może się pojawić i wówczas z naszego serwera będą korzystać wszystkie hosty należące do tej domeny. To również nie jest najlepsze rozwiązanie.

To co pokazałem powyżej to tylko dla testów. W dalszej części opiszę znacznie lepsze rozwiązanie tego problemu. Nie znaczy to jednak, że opcji `RELAY` nie możemy w ogóle stosować. Wręcz przeciwnie, możemy i czasami nawet należy ją stosować, ale nie dla całych sieci tylko dla szczegółowo podanych adresów i to takich, które cały czas są zajęte (hosty dla których robimy relay pracują w trybie 24/7). Wprawdzie nie eliminuje to całkowicie możliwości podszycia, ale utrudnia w takim stopniu, że spamerzy raczej nie będą tego próbować.

Przystąpmy zatem do testów. Odświeżamy konfigurację naszego sendmaila i restartujemy demona. Teraz możemy usiąść do komputera w naszej sieci i skonfigurować jakiegoś klienta pocztowego. Istotą takiej konfiguracji jest podanie naszego serwera *dreptak.tupward.pl* jako serwera SMTP. Pamiętaj, aby nie wybierać autoryzacji przy wysyłaniu poczty — nasz serwer jeszcze nie obsługuje tej funkcjonalności, to będzie w następnym rozdziale. Teraz tworzymy nową wiadomość i próbujemy ją wysłać.

14.2 Autoryzacja SMTP

Jak wcześniej pisałem relay poczty stwarza wiele problemów. Możemy je jednak znacznie ograniczyć stosując autoryzację SMTP. Zacznijmy od wytłumaczenia jak to działa. W dotychczasowej konfiguracji klient wysyłał pocztę do naszego serwera, a ten sprawdzał, czy dany host pasuje do listy relay i pocztę przesyłał dalej lub odrzucał. Teraz wprowadzamy inny mechanizm.

Klient przedstawia się, a następnie podaje nazwę i hasło użytkownika. Nasz serwer sprawdza, czy zna takiego użytkownika i jeśli odpowiedź jest twierdząca przekazuje pocztę dalej, w przeciwnym przypadku przesyłkę odrzuca z odpowiednim komunikatem. Jak widzisz nie ma w tej metodzie sprawdzania listy relay, co eliminuje większość problemów wcześniej opisywanych.

Przystąpmy zatem do konfigurowania tej funkcjonalności. Aby nasz serwer pocztowy mógł wykonać autoryzację potrzebny nam będzie `sasl`, a konkretnie pakiety `cyrus-sasl`.

```
[root@dreptak root]# apt-cache search sasl
cyrus-sasl-gssapi - GSSAPI support for Cyrus SASL.
postfix - Postfix Mail Transport Agent
cyrus-sasl-devel - Files needed for developing applications with Cyrus SASL.
libesmtp - SMTP client library.
cyrus-sasl - The Cyrus SASL library.
cyrus-sasl-md5 - CRAM-MD5 and DIGEST-MD5 support for Cyrus SASL.
cyrus-sasl-plain - PLAIN and LOGIN support for Cyrus SASL.
[root@dreptak root]# apt-get install cyrus-sasl cyrus-sasl-plain cyrus-sasl-md5
Reading Package Lists... Done
Building Dependency Tree... Done
cyrus-sasl is already the newest version.
cyrus-sasl-plain is already the newest version.
cyrus-sasl-md5 is already the newest version.
0 upgraded, 0 newly installed, 0 removed and 0 not upgraded.
[root@dreptak root]#
```

Aurox ma to do siebie, że wiele zadań wykonuje za nas. Jednym z nich jest instalacja wymienionych wcześniej pakietów i w tym przypadku jest to słuszne podejście. Jeśli ktoś decyduje się na uruchomienie serwera smtp, to raczej trudno sobie dzisiaj wyobrazić poprawne działanie takiego serwera bez autoryzacji. Co ciekawe, pakiet `cyrus-sasl` bezpośrednio po zainstalowaniu jest gotowy do wykorzystania i nie wymaga od nas żadnych dodatkowych prac. Warto natomiast wiedzieć, że sposób przeprowadzenia autoryzacji zapisany jest w pliku `/usr/lib/sasl2/Sendmail.conf`. Plik ten pochodzi z pakietu `sendmail`, gdyż twórcy dystrybucji przewidzieli konieczność jego utworzenia.

Kolejnym krokiem jest zmodyfikowanie konfiguracji sendmaila. Przechodzimy zatem do katalogu `/etc/mail`, otwieramy plik `sendmail.mc` i modyfikujemy. Ja proponuję skopiować dwie linie dotyczące autoryzacji i odpowiednio je zmodyfikować :

```
dnl TRUST_AUTH_MECH(‘EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN’)dnl
dnl define(‘confAUTH_MECHANISMS’, ‘EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN’)dnl
dnl
dnl Moje modyfikacje
TRUST_AUTH_MECH(‘LOGIN PLAIN’)dnl
define(‘confAUTH_MECHANISMS’, ‘LOGIN PLAIN’)dnl
```

Jak widzisz w konfiguracji pozostawiłem tylko dwie metody autoryzacji — `LOGIN` i `PLAIN`. Te dwie metody pozwalają na przeprowadzenie autoryzacji na podstawie danych przesłanych tekstowo. Pozostałe metody są znacznie bezpieczniejsze, gdyż wymagają, aby przesyłane hasło było szyfrowane. Mają natomiast jedną poważną wadę — nie wszystkie programy klienckie chcą z nimi współpracować. Możemy wprawdzie próbować wykorzystania SSL do komunikacji pomiędzy klientem i naszym serwerem, ale to również nie będzie działać ze wszystkimi klientami, gdyż część z nich wymaga, aby klucze podpisane były przez zaufane centra autoryzacyjne.

Skoro mamy już poprawioną konfigurację w `sendmail.mc` wykonujemy jak zwykle odbudowę plików konfiguracyjnych i restartujemy sendmaila. Teraz możemy sprawdzić działanie wysyłania poczty. W tym celu musimy zmodyfikować konfigurację klienta pocztowego zaznaczając, że serwer smtp wymaga uwierzytelniania i podać nazwę użytkownika i hasło. Po tych zmianach wykonujemy test wysyłania poczty. Nie przewiduję tutaj problemów, więc możemy przejść do następnego kroku, a mianowicie skasowania z pliku `access` linii odpowiedzialnych za relay —

wszystkich z wyjątkiem trzech pierwszych. I jak zwykle po takich zmianach odświeżamy konfigurację poleceniem `make` i restartujemy sendmaila. Autoryzacja zrobiona.

14.3 Czytanie poczty

Nasz system umie już odbierać i wysyłać pocztę, jednak brakuje mu jeszcze jednej funkcjonalności. Większość użytkowników będzie zapewne korzystać z jakiegoś programu klienckiego i chcieliby z jego poziomu sięgać do swoich skrzynek pocztowych. Sendmail jednak obsługuje jedynie protokół SMTP, a nam potrzebny jest POP3 lub IMAP. Musimy zatem zainstalować dodatkowe oprogramowanie obsługujące te protokoły. W Auroksie mamy dwa pakiety realizujące tą funkcjonalność — `imap` oraz `dovecot`. Do naszych potrzeb wykorzystamy pakiet `imap`, jako że jest wyjątkowo prosty w uruchomieniu. Zatem zabieramy się do pracy.

Zanim przystąpimy do instalacji sprawdźmy, czy przypadkiem nie mamy już zainstalowanego tego pakietu. Skoro `sendmail` był zainstalowany, to jest spora szansa, że `imap` również znajduje się już na dysku :

```
[tuptus@dreptak Aurox-PT]$ rpm -qa | grep imap
imap-2002d-3
php-imap-4.3.6-9.4.aur.1
[tuptus@dreptak Aurox-PT]$
```

Jak miło. Jest zainstalowany, więc przejdziemy do dalszych kroków. Jeśli u Ciebie nie ma tego pakietu, to musisz go zainstalować. Mam nadzieję, że na tym etapie kursu poradzisz sobie samodzielnie. Nawet jestem tego pewien.

Zapewne już zastanawiasz się, gdzie są pliki konfiguracyjne i co trzeba będzie modyfikować. Zaskoczę Cię. Nic nie będziemy konfigurować i nie ma żadnych plików konfiguracyjnych. Jedyne o co musimy zadbać, to automatyczny start serwisu przy starcie systemu. Jednak tutaj musisz się nieco skupić, gdyż uruchomienie wyglądać będzie nieco inaczej niż to robiliśmy do tej pory. Różnica ta wynika z tego, że aplikacje zawarte w pakiecie `imap` nie są samodzielnymi demonami jak sendmail czy apache, ale działają za pośrednictwem "super serwera" `xinetd`. Samodzielne demony mają swoje pliki startowe w katalogu `/etc/rc.d/init.d` i możemy nimi zarządzać wydając polecenie `service <nazwa demona> <polecenie>`. Dla POP3 i IMAP plików takich nie znajdziemy. Znajdują się jednak odpowiednie pliki w katalogu `/etc/xinetd.d` i stanowią one część konfiguracji demona xinetd. Nas interesuje w tej chwili plik `ipop3`, a właściwie jedna linia z tego pliku :

```
        disable = yes
```

Właściwie wystarczy zmodyfikować `yes` na `no` i wykonać restart demona xinetd. Możemy jednak wykorzystać polecenia systemowe lub programy pomagające zarządzać serwisami. Już kilka razy wspominałem o poleceniu `ntsysv`. Uruchamiamy ten program, zaznaczmy serwis `ipop3` i po wyjściu restartujemy demona xinetd. Można wykorzystać również polecenie `chkconfig`. Należy jednak pamiętać, że start usługi ipop3 nie jest związany z żadnym *runlevelem*. Zatem ustawiamy system :

```
[root@dreptak root]# chkconfig ipop3 on
[root@dreptak root]# service xinetd restart
Zatrzymywanie xinetd:                                      [   OK   ]
Uruchamianie xinetd:                                       [   OK   ]
[root@dreptak root]#
```

Teraz wypadałoby jeszcze sprawdzić, czy wszystko działa poprawnie i następny serwis mamy uruchomiony.

14.4 Poczta przez www

Wprawdzie nasz system pocztowy jest już w pełni funkcjonalny, ale do pełni szczęścia brakuje nam jeszcze jednej usługi. Często się zdarza, że nie mamy dostępu do naszego komputera (np. jesteśmy na wczasach lub na szkoleniu), a chcielibyśmy mieć dostęp do poczty. Konfigurowanie klienta pocztowego na obcym komputerze to nie jest najlepszy pomysł. Przecież na tym komputerze zostaną nasze hasła oraz pobrana poczta. Dostęp przez ssh do serwera też nie zawsze jest możliwy. Możliwy jest natomiast dostęp do stron www. Zatem wskazane by było, aby nasz system umożliwiał dostęp do poczty właśnie w ten sposób. Okazuje się, że twórcy dystrybucji zadbali o nasze potrzeby. W dystrybucji mamy aplikację *squirrelmail* nazywaną wiewiórą. Jest to aplikacja napisana w *php*, wymaga więc działającego serwera www obsługującego php. Informacje o uruchomieniu takiego serwisu możesz poczytać w rozdziale "Serwer stron www", tutaj zakładam, że serwer ten jest już uruchomiony.

Przystępujemy zatem do uruchomienia serwisu. Pierwszym krokiem jest oczywiście instalacja pakietu `squirrelmail`, ale z tym sam sobie poradzisz. Ponieważ wiewióra dodaje pewne informacje do konfiguracji serwera apache, to trzeba wykonać restart tego demona poleceniem `service httpd restart`. Ponieważ wiewióra wykorzystuje protokół IMAP musimy uruchomić serwis obsługujący ten protokół. Wykonujemy to analogicznie jak dla POP3.

```
[root@dreptak root]# chkconfig imap on
[root@dreptak root]# service xinetd restart
Zatrzymywanie xinetd:                                      [   OK   ]
Uruchamianie xinetd:                                       [   OK   ]
[root@dreptak root]#
```

Teraz możemy sprawdzić, czy nasz serwis się uruchamia. W przeglądarce wpisujemy *http://dreptak.tupward.pl/webmail* i powinniśmy zobaczyć stronę logowania do wiewióry. Jeśli tak się stanie, to właściwie możemy powiedzieć, że serwis działa.

Niestety system działa domyślnie w języku angielskim — powinniśmy coś z tym zrobić. Pliki konfiguracyjne wiewióry znajdują się w katalogu `/usr/share/squirrelmail/config`. W plikach `config.php` i `config_default.php` radziłbym nic nie zmieniać. Dla nas przeznaczono plik `config_local.php`, to właśnie w tym pliku możemy umieścić odpowiednie wpisy :

```
<?
$squirrelmail_default_language = 'pl_PL';
$default_charset          = 'iso-8859-2';
?>
```

Tylko dlaczego się męczyć z ręczną modyfikacją plików, skoro możemy to zrobić z poziomu przeglądarki? Musimy jednak wcześniej przygotować odpowiednio wiewiórę. Przechodzimy zatem do katalogu `/usr/share/squirrelmail/config` i uruchamiamy `./conf.pl`. Z menu wybieramy pozycje *Plugin* i do zestawu dodajemy *administrator*. Teraz zapisujemy nasze zmiany i kończymy pracę z tym programem. Kolejnym krokiem jest przejście do katalogu `/etc/squirrelmail` i zmiana właściciela pliku `config.php`. Aby mechanizm konfiguracyjny zadziałał, to właścicielem tego pliku musi być użytkownik na uprawnieniach którego pracuje apache — standardowo apache.

```
[root@dreptak root]# cd /etc/squirrelmail/
[root@dreptak squirrelmail]# chown apache config.php
[root@dreptak squirrelmail]#
```

Kolejnym krokiem jest utworzenie pliku zawierającego nazwy użytkowników, którzy będą mogli korzystać z tego pluginu.

```
[root@dreptak squirrelmail]# cd /usr/share/squirrelmail/plugins/administrator/
```

```
[root@dreptak administrator]# touch admins
[root@dreptak administrator]#
```

Teraz otwieramy plik `admins` i wpisujemy login administratora. Dzięki tym manewrom wśród *"Opcji"* wiewióry pojawi się pozycja *Administracja*, ale tylko dla użytkownika, którego podaliśmy w pliku `admins`. Możemy teraz administrować zdalnie naszym serwisem pracując cały czas w oknie przeglądarki.

Dzięki zastosowaniu przeglądarki www możemy nieznacznie podnieść poziom bezpieczeństwa systemu pocztowego. Otóż cały wiewiórczy serwis możemy uruchomić jako HTTPS czyli strona www w tunelu SSL. Nie będę tego teraz omawiał, bo to już temat dotyczący samego serwera www i będzie poruszony w odpowiednim rozdziale.

14.5 Ochrona antywirusowa i antyspamowa

Z problemem spamu i wirusów spotkał się chyba każdy i wszyscy próbują się przed tymi zjawiskami bronić. Większość klientów pocztowych zawiera mechanizmy pozwalające na filtrowanie niechcianej poczty. Do wielu z nich można również dodawać dodatkowe narzędzia zwiększające skuteczność filtrowania. Wszystkie one mają jednak jedną zasadniczą wadę. Aby odfiltrować śmieci trzeba je wcześniej ściągnąć z serwera na stację, a to powoduje niepotrzebny ruch i wydłuża czas dostępu do tych istotnych wiadomości. Rozwiązaniem tego problemu może być zastosowanie filtrów już na serwerze. Jeśli masz na serwerze własny katalog domowy, to możesz wykorzystać narzędzie o nazwie `procmail` i zlecić temu procesowi wykonywanie filtrowania przychodzącej poczty. To rozwiązanie, jakkolwiek dużo lepsze od filtrowania na kliencie, jeszcze nie załatwia wszystkiego. Poczta niechciana nadal będzie wchodzić na serwer, będzie zapychać łącze i obciążać serwer pocztowy. Poza tym każdy użytkownik musiałby indywidualnie ustawiać sobie odpowiednie reguły, a nie jest to sprawa prosta. Jeszcze lepszym rozwiązaniem jest filtrowanie poczty przez sam serwer SMTP, najlepiej na poziomie samego protokołu, gdyż w tym wypadku wiadomości mogą być odrzucane już po przeczytaniu samego nagłówka poczty bez konieczności pobierania treści.

Skoro wiemy, co chcemy osiągnąć, to zastanówmy się co będzie nam potrzebne i co z tego mamy w dystrybucji. Pierwszą sprawą jest serwer SMTP — to już mamy i działa. Kolejna sprawa to jakieś narzędzie do filtrowania spamu. W dystrybucji mamy pakiet *spamassassin*. Jest to skrypt (a właściwie grupa skryptów) napisana w perlu. Narzędzie to można wykorzystać indywidualnie włączając jego wywołania do regułek procmaila. Kolejna sprawa to program antywirusowy. Wprawdzie Linuks nie boi się wirusów, ale przecież z naszego serwera mogą korzystać również programy klienckie pracujące na innych systemach operacyjnych. Poza tym poczta z wirusami to też rodzaj poczty niechcianej i należy ją odfiltrować niezależnie od jej zagrożenia dla naszego systemu. Tutaj z pomocą przychodzi aplikacja *clamAV* dołączona do dystrybucji w wersji 10.

Pozostaje jeszcze problem połączenia wymienionych narzędzi z serwerem SMTP — w naszym przypadku z sendmailem. Tutaj mamy pewien problem. Jednym z najlepszych "łączników" jest *amavis*, ale w momencie pisania tego rozdziału, nie ma tego narzędzia w dystrybucji. W wersji 10 Auroksa narzędzie to znajduje się na piątym krążku wśród pakietów grupy *extra*. Wykorzystamy te pakiety i miejmy nadzieję, że w następnych wydaniach *amavis* wejdzie na stałe do dystrybucji.

Mamy zatem zgromadzone narzędzia, więc zabieramy się do pracy.

14.5.1 Wirusy

Zaczniemy od instalacji oprogramowania antywirusowego. Potrzebne nam będą następujące pakiety:

- clamav — główne narzędzie zestawu
- clamav-lib — biblioteki procedur
- clamav-data — baza znanych wirusów (trzeba będzie ją zaktualizować)
- clamav-update — narzędzie do aktualizacji bazy wirusów

Instalujemy podane pakiety i przechodzimy do aktualizacji bazy wirusów. W tym celu, jako root, uruchamiamy polecenie `freshclam` i po chwili mamy zaktualizowana bazę. Oczywiście aktualizację taką trzeba wykonywać co pewien czas. Nie będziemy tego oczywiście robić ręcznie, od tego mamy demony, żeby takie zadania wykonywały za nas. W pakiecie clamav-update znajduje się plik z opisem dla crona i po instalacji ląduje on w katalogu `/etc/cron.d`. Wystarczy "odkomentować" ostatnią linijkę i wykonać restart demona *crond*, a aktualizacja będzie się wykonywać automatycznie co trzy godziny. Oczywiście częstotliwość tej aktualizacji możesz zmienić, ale obsługę crona już znasz, więc zapewne poradzisz sobie.

14.5.2 Spam

Jak już wcześniej wspomniałem w dystrybucji mamy pakiet *spamassassin*. Ponieważ pakiet ten wymaga wielu innych pakietów, to do instalacji polecam wykorzystanie polecenia `apt-get`. Polecenie to znajdzie wszystkie zależności i zainstaluje wszystko, co jest nam potrzebne. Ale to jeszcze nie koniec instalacji. Od czasu przygotowania pakietu do czasu kiedy go instalujesz upłynęło trochę czasu. Spamerzy to zmyślne bestie i ciągle wymyślają nowe sposoby zasypywania naszych skrzynek pocztowych ofertami powiększania tego i owego. Musimy zatem zadbać o to, aby nasz "system obronny" był również jak najbardziej aktualny — musimy zaktualizować spamassassina. Ponieważ jest to narzędzie perlowe, to aktualizacji najlepiej szukać na serwerach CPAN. W tym celu wykonujemy polecenie :

```
[root@dreptak CPAN]# perl -MCPAN -e shell
```

Jeśli polecenie to wykonujesz pierwszy raz, to moduł CPAN będzie Cię prosił o podanie pewnych danych niezbędnych do ustalenia konfiguracji połączenia. Proponuję zaakceptować domyślne wartości. Jeśli Twoja maszyna stoi za jakimś ogniomurkiem, to może być konieczne podanie adresów serwerów proxy. Jeśli jednak z nich nie korzystasz, to zostaw te pozycje puste. W pewnym momencie dojdziemy do ustalenia naszej lokalizacji. System będzie nas pytał kolejno o kontynent (wybieramy Europę) i kraj. Ponieważ Polska nie zmieściła się na liście, to zgodnie z instrukcją wciskamy SPACE i akceptujemy. Teraz już mamy nasz kraj na liście. Kolejny ekran to lista serwerów lustrzanych. Zalecałbym wybranie wszystkich, a system sam zadba o to, aby w przyszłości korzystać z tego, który w danym momencie będzie "najlepszy".

Teraz otrzymujemy prompt i możemy przystąpić do właściwego procesu instalacji.

```
cpan> install Mail::SpamAssassin
Mail::SpamAssassin is up to date.

cpan> quit
```

Jak widzisz w moim przypadku nie było potrzeby aktualizacji, ale zawsze warto to sprawdzić. Na zakończenie oczywiście opuszczamy perlowy shell.

W tym momencie możemy powiedzieć, że jesteśmy przygotowani do walki ze spamerami i możemy przejść do następnego kroku.

14.5.3 Łączymy części składowe

Proces łączenia wszystkich elementów zaczynamy od zainstalowania *amavisa*. Pakiet ten wymaga wielu dodatkowych pakietów, więc znowu zaprzęgamy do pracy *apta*. Po instalacji pakiet jest już wstępnie skonfigurowany, ale musimy wprowadzić pewne uzupełnienia związane z lokalnymi wymaganiami. Zaczynamy od pliku `/etc/amavis.conf`. W pliku tym musimy zmodyfikować pierwszą linię ustawiając właściwą nazwę domeny :

```
# $mydomain = 'example.com';
$mydomain = 'tupward.pl';
```

Informacja ta jest szalenie istotna dla *amavisa*, gdyż na tej podstawie rozpoznaje pocztę lokalną i zewnętrzną. Na podstawie tej zmiennej budowane są również wiadomości wysyłane przez system np. informacja o znalezieniu wirusa.

Następnym elementem, który musimy ustalić to adresy pocztowe. Naszym zadaniem jest ustawienie adresu, z którego *amavis* będzie wysyłał wiadomości oraz adresów, na które będzie przesyłał raporty. Odszukujemy zatem odpowiednie linie i modyfikujemy (jeśli uznamy to za stosowne) :

```
$virus_admin               = "virusalert\@$mydomain";
$mailfrom_notify_admin     = "virusalert\@$mydomain";
$mailfrom_notify_recip     = "virusalert\@$mydomain";
$mailfrom_notify_spamadmin = "spam.police\@$mydomain";
```

Pierwsza pozycja to adres administratora zajmującego się ochroną antywirusową. Na ten adres wysyłane będą raporty o wykryciu wirusa w poczcie. Proponowałbym dodać jeszcze adres administratora odpowiedzialnego za ochronę antyspamową. Adres ten podstawić należy do zmiennej `$spam_admin`. Kolejne trzy linie określają adresy umieszczane w polu *From:* wiadomości wysyłanej odpowiednio do administratora, adresata wiadomości, administratora zajmującego się spamem. Do adresata raczej bym nie wysyłał informacji o wirusach i spamie, przecież chcemy go właśnie chronić przed poczta śmieciową. Zatem ostatecznie linie te przybiorą postać na przykład taką :

```
$virus_admin               = "vsadmin\@$mydomain";
$spam_admin                = "vsadmin\@$mydomain";
$mailfrom_notify_admin     = "virusalert\@$mydomain";
$mailfrom_notify_recip     = undef;
$mailfrom_notify_spamadmin = "spampolice\@$mydomain";
```

I to by było na tyle w tym pliku. Oczywiście nie wyczerpuje to możliwości konfiguracyjnych *amavisa*. Proponuję zapoznać się z zawartością katalogu `/usr/share/amavisd-new-20040701`, a w szczególności z plikami `amavisd.conf-default` (zawiera wszystkie opcje konfiguracyjne i ich wartości domyślne) i `amavisd.conf-sample`.

Teraz należy zadbać o to, aby programy pocztowe nie narzekały na brak użytkownika. Możemy oczywiście założyć odpowiednie konta w systemie tylko, że takiej potrzeby nie ma. Przecież istnieje system aliasów, który pozwoli przekierować pocztę na jakieś już istniejące konto. Otwieramy zatem plik `/etc/aliases` i dopisujemy odpowiednie linie. U mnie będzie to wyglądać następująco :

```
vsadmin: tuptus
virusalert: tuptus
spampolice: tuptus
```

Teraz jeszcze trzeba poinformować *sendmaila* o zaistniałych zmianach. Wykonujemy to zadanie poleceniem `newaliases`.

Teraz kolej na samego *sendmaila*. Przechodzimy do katalogu `/etc/mail` i otwieramy plik `sendmail.mc`. W trakcie instalacji *amavisa* na końcu tego pliku zostały dodane dwie linie potrzebne do połączenia pracy obu programów. Obecnie linie te traktowane są jako komentarz, kasujemy więc początkowe "`dnl`" w obu liniach i odświeżamy konfigurację sendmaila poleceniem `make`.

Mamy już wszystko przygotowane. Teraz już tylko start odpowiednich serwisów. Wykonujemy zatem `service amavisd start` i restartujemy *sendmaila* i możemy się cieszyć uruchomioną ochroną antywirusowa i antyspamową.

Niestety w tym miejscu troszkę Cię zmartwię. To jest dopiero początek drogi. Aby system antyspamowy działał poprawnie trzeba najpierw nauczyć *spamassassina*, która wiadomość jest spamem, a która nie. Oczywiście *spamassassin* już sporo umie, ale początkowo mogą się zdarzać błędy. Przyjmuje się, że system jest "wykształcony", gdy przeanalizuje ok. 200 wiadomości. Jeśli w tym czasie nie pojawią się błędy, to masz szczęście, ale w przeciwnym wypadku ...

Do nauczania wykorzystujemy polecenie `sa-learn`. Aby jednak rozpocząć naukę musimy mieć wiadomości, które będą podstawą nauki. Wiadomości zakwalifikowane jako spam, a będące wiadomościami poprawnymi możemy znaleźć w katalogu kwarantanny, czyli w `/var/spool/amavis/virusmails`. W katalogu tym znajdują się pliki, których nazwy zaczynają się od `spam` i są spakowane programem *gzip*. Na podstawie raportów jakie otrzymał administrator odszukujemy odpowiedni plik i rozpakowujemy go poleceniem `gunzip <plik>.gz`. Teraz przechodzimy na konto *amavis* i wykonujemy polecenie `sa-learn --mbox --ham <plik>`. Po takiej sesji *spamassassin* już odnotował, że tego typu wiadomość nie jest spamem i będzie łagodniej podchodził do wiadomości podobnych do tej, którą przed chwilą otrzymał do nauki.

W przypadku spamu, który w jakiś sposób prześlizgnął się przez nasze sito musimy postąpić nieco inaczej, gdyż wiadomości tych nie ma na serwerze — znajdują się na stacji klienckiej. Zatem w programie pocztowym wykonujemy zapis wiadomości do pliku na dysku i przenosimy (np. przez ftp) na serwer. Teraz jako root musimy takie pliki przenieść w miejsce dostępne dla użytkownika *amavis* i ustawić uprawnienia tak, aby mógł je swobodnie czytać. Następnie wpisujemy polecenie `sa-learn --mbox -spam <plik>` i czekamy aż *spamassassin* przeanalizuje treść i zaktualizuje swoje bazy.

Innym problemem są adresy, z których przychodzą wiadomości pasujące do wzorców spamu, ale jednak chcemy je otrzymywać. Przykładem takich wiadomości są informacje pochodzące z Allegro oraz różnego rodzaju subskrypcje np. księgarni internetowych. Aby sobie radzić z tego typu adresami tworzymy tzw. whitelisty. Adresy, które wpiszemy na taką listę otrzymują już na wstępie odpowiednie punkty co ułatwia przeprowadzenie ich przez filtry. Zaczątek takiej whitelisty mamy już w pliku `amavisd.conf`. Definicja ta zaczyna się od *"@score_sender_maps ({"*. Jeśli chcemy "pomóc" wiadomościom przychodzącym z adresu np. ksiegarnia@domena.pl przejść przez *amavisa*, to dopisujemy ten adres do listy adresów z ujemnymi punktami. Jeśli jest to nasz pierwszy wpis, to odpowiedni fragment pliku `amavisd.conf` będzie wyglądał następująco :

```
    'clusternews@linuxnetworx.com'           => -3.0,
    lc('lvs-users-admin@LinuxVirtualServer.org')    => -3.0,
    lc('owner-textbreakingnews@CNNIMAIL12.CNN.COM') => -5.0,
    # moje dodatki do whitelisty
    'ksiegarnia@domena.pl'                   => -5.0,

    # soft-blacklisting (positive score)
```

Informacje powyższe to tylko wierzchołek góry lodowej. Jeśli zaczniesz walkę ze spamem i wirusami, to musisz się przygotować na sporo dodatkowej pracy, szczególnie na początku. Również użytkownicy mogą się początkowo nieco denerwować i będziesz zmuszony tłumaczyć im dlaczego wprowadziłeś takie restrykcje. Ale zapewniam Cię, że ten wysiłek się opłaci i po pewnym czasie użytkownicy podziękują Ci za Twój trud.

Rozdział 15

DHCP - dynamiczna konfiguracja hostów

15.1 Trochę teorii

Usługa DHCP jest usługą serwerową pozwalającą na zarządzanie hostami w sieci z jednego miejsca. Rozwiązanie takie pozwala na centralizację zadania zarządzania siecią, a dzięki dynamice przydzielania adresów IP zwiększa bezpieczeństwo sieci.
Komunikacja między serwerem i klientem prowadzona jest protokołem UDP. Serwer korzysta z portu 67, a klient 68. Ponieważ klient nie wie nic o sieci IP wysyła więc wołanie "do wszystkich" i czeka kto odpowie. Jest to zapytanie nazywane `DHCPDISCOVER`. Kiedy serwer otrzyma takie zapytanie odpowiada pakietem `DHCPACK` zawierającym informacje potrzebne klientowi do konfiguracji sieci. W przypadku gdy klient chce zaktualizować swoją dzierżawę wysyła do serwera pakiet `DHCPREQUEST`. Serwer może takie zapytanie zignorować, nakazać klientowi zaprzestanie używania danego adresu wysyłając pakiet `DHCPNAK` lub przesłać pakiet `DHCPACT`.
Skoro klient nie zna sieci IP, to jak następuje komunikacja? Otóż do komunikacji pomiędzy serwerem DHCP i klientem wykorzystywane są adresy sprzętowe kart sieciowych, tzw. MAC adresy. Stanowi to poważne ograniczenie ponieważ komunikacja tego typu odbywać się może wyłącznie w ramach jednego segmentu fizycznego sieci. Pojawia się również drugi problem. Jeśli w jednej sieci fizycznej mamy dwie sieci logiczne, to zapytania klienta docierać będą do hostów w obu sieciach logicznych. Należy pamiętać o tych uwarunkowaniach projektując sieć.

15.2 Instalacja i konfiguracja

Uruchomienie usługi zaczynamy od instalacji pakietu `dhcp` i utworzeniu odpowiednich plików konfiguracynych. Podstawowym plikiem konfiguracyjnym jest plik `/etc/dhcpd.conf`. Pliku tego nie ma w pakiecie, więc zaczynamy od utworzenia tego pliku :

```
[tuptus@dreptak test]$ su -
[root@dreptak ~]# cd /etc
[root@dreptak etc]# touch dhcpd.conf
[root@dreptak etc]#
```

W pliku tym umieszczamy wszystkie dane potrzebne do właściwego funkcjonowania naszego serwera. Należy otworzyć plik `dhcpd.conf` w dowolnym edytorze. Podstawowa konfiguracja może wyglądać tak :

```
option domain-name "domek.pl";
option domain-name-servers 192.168.1.1;
log-facility local0;
ddns-update-style none;
ddns-updates off;

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.10 192.168.1.250;
    option routers 192.168.1.1;
    default-lease-time 600;
    max-lease-time 86400;
}
```

Linia pierwsza ustala nazwę domenową dla hostów otrzymujących adresy z naszego serwera. Druga linia definiuje adres serwera DNS, z którego mają korzystać hosty w sieci. Na liście może być kilka serwerów oddzielonych przecinkiem jeśli występują w naszej sieci. Linie rozpoczynające się od `ddns` dotyczą współpracy z serwerem DNS i omówimy je w odrębnej części. Również linię `log-facility`, dotyczącą logowania zdarzeń serwera DHCP, omówimy osobno.
Dalej następuje definicja dla naszej podsieci. Definicja `range` pozwala na podanie zakresu adresów IP, z którego hosty będą otrzymywać adresy. Dla każdej podsieci musi występować co najmniej jedna deklaracja `range`. Jeśli chcemy wykorzystać kilka zakresów adresów, to podajemy je w kolejnych definicjach `range`. Opcja `routers` pozwala na przekazanie hostom klienckim adresów IP routerów rozdzielonych przecinkami. W naszej sieci mamy tylko jeden router. Definicje `lease-time` opisują czasy dzierżawy adresu IP podane w sekundach. Informacje o przyznanych dzierżawach zostają zapisane do pliku `/var/lib/dhcp/dhcp.leases`. Przykładowe czasy dzierżawy : 600 sekund (dziesięć minut), 3600 sekund (godzina), 86400 (jeden dzień), 604800 (jeden tydzień) i 2592000 (trzydzieści dni).
Po skonfigurowaniu serwera DHCP oraz usług dodatkowych należy wystartować usługę poleceniem `service dhcpd start` i przeprowadzić testy jego funkcjonowania. Jeśli wszystko działa poprawnie, to pozostaje jeszcze zapewnienie automatycznego startu usługi przy starcie systemu. Wykonamy to zadanie poleceniem `chkconfig --level 35 dhcpd on`, które informuje system, że w przypadku startu w "runlevel" 3 i 5 ma być uruchamiany serwer DHCP.

15.3 Dynamiczny DNS

Konfiguracja przedstawiona powyżej działa poprawnie, jednak ma jedną poważną wadę. Wprawdzie przydziela hostom w naszej sieci adresy IP, jednak nie wiadomo kto ma jaki adres. Można oczywiście przeglądać logi, ale to nie jest rozwiązanie. Nie można się również odwoływać do hostów przez ich nazwę, a jedynie przez IP, ale to przecież się zmienia. Rozwiązaniem tego problemu może być dokonanie odpowiednich wpisów w lokalnym serwerze nazw. Nazwę naszej domeny mamy w pliku konfiguracyjnym, adres IP przydziela serwer DHCP, a nazwę hosta przesyła sam klient. Mamy zatem wszystkie elementy niezbędne do dokonania wpisów w odpowiedniej strefie DNS. Ze względu na dynamikę zmian powyższych parametrów należy zorganizować automatyczne modyfikowanie wpisów.

15.3.1 Modyfikacja serwera DNS

Z oczywistych względów automatyczna modyfikacja serwera DNS musi być zabezpieczona. W tym celu wykorzystamy możliwość zastosowania kluczy do komunikacji klienta z serwerem DNS. Zaczniemy od wygenerowanie klucza wykorzystując polecenie `dnssec-keygen` :

```
[root@dreptak root]# dnssec-keygen -a hmac-md5 -b 128 -n USER DHCP_KEY
```

```
Kdhcp_key.+157+52904
[root@dreptak root]# ls -l
razem 176
(...)
-rw-------    1 root  root  50 maj  1 09:32 Kdhcp_key.+157+52904.key
-rw-------    1 root  root  81 maj  1 09:32 Kdhcp_key.+157+52904.private
[root@dreptak root]#
```

Jak widać otrzymaliśmy dwa pliki, nas interesuje ten drugi. Należy go skopiować do katalogu `/etc` i odpowiednio zmodyfikować. W efekcie powinniśmy uzyskać plik `/etc/dhcp.key` o następującej zawartości :

```
key "DHCP_KEY" {
    algorithm hmac-md5;
    secret "HSlW3SamIqnczargtJjSbg==";
};
```

Oczywiście `secret` w Twoim przypadku może być inny, gdyż generowany jest na bazie losowego źródła. Należy jeszcze zadbać, aby serwis `named` miał dostęp do tego pliku :

```
[root@dreptak root]# cd /etc
[root@dreptak etc]# chgrp named dhcp.key
[root@dreptak etc]# chmod 640 dhcp.key
[root@dreptak etc]#
```

Teraz kolej na modyfikację konfiguracji serwera DNS. Modyfikacja polega na dodaniu opcji `allow-update` w definicji strefy naszej domen oraz strefy odwrotnej. Zaczynamy od dodania definicji klucza i dokonujemy odpowiednich wpisów dla stref. W efekcie plik `/etc/named.conf` przyjmuje postać (fragment) :

```
(...)
include "/etc/rndc.key";
include "/etc/dhcp.key";
zone  "1.168.192.in-addr.arpa" {
        type master;
        file  "1.168.192.in-addr.arpa.zone";
        allow-update { key DHCP_KEY; };
};
zone  "domek.pl" {
        type master;
        file  "domek.pl.zone";
        allow-update { key DHCP_KEY; };
};
```

W zasadzie to wszystko, można zresetować serwis named. Warto jednak zadbać o logowanie zdarzeń aktualizacji serwera DNS w osobnych plikach. W tym celu należy dodać do pliku `/etc/named.conf` sekcję logging w postaci podobnej do poniższej :

```
logging {
            channel update_debug {
                 file "/var/log/dhcp/update-dns.log" versions 4;
                 severity  debug 3;
                 print-category yes;
                 print-severity yes;
                 print-time     yes;
            };
            channel security_info    {
                 file "/var/log/dhcp/named-auth.info" versions 4;
                 severity info;
                 print-category yes;
                 print-severity yes;
```

```
                print-time      yes;
            };
            category update { update_debug; };
            category security { security_info; };
        };
```

Poleceniem `touch` tworzmy odpowiednie pliki i ustalamy właściciela odpowiednio do konfiguracji serwera DNS. Warto jeszcze zerknąć do pliku `/var/log/messages` i sprawdzić czy przy restarcie nameda nie pojawiły się jakieś błędy.

Uwaga. W momencie pierwszego odwołania do serwera wymagającego modyfikacji strefy powstaną nowe pliki w katalogu `/var/named`. Będą to pliki o nazwie identycznej z nazwą pliku strefy z dodanym rozszerzeniem `.jnl`. Pliki te nazywane są dziennikami (journal) i zawierają binarny zapis zmiennych elementów strefy. Nie należy ich edytować ręcznie. Jeśli zaistnieje konieczność ręcznej modyfikacji pliku strefy należy :

1. wykonać polecenie `rndc flush`
2. zatrzymać serwis named
3. usunąć pliki dzienników
4. dokonać niezbędnych modyfikacji pliku strefy
5. uruchomić serwis named

15.3.2 Modyfikacja serwera DHCP

Skoro serwer DNS jest już skonfigurowany przyszła kolej na modyfikację serwera DHCP. Zaczynamy od dodania do pliku `/etc/dhcpd.conf` definicji klucza. Wklejamy zawartość pliku `/etc/dhcp.key` do `dhcpd.conf` i kasujemy końcowy średnik :

```
[root@dreptak root]# cd /etc
[root@dreptak etc]# cat dhcp.key >> dhcpd.conf
[root@dreptak etc]# vim dhcpd.conf
```

Teraz dodajemy jeszcze opis stref DNS, które zamierzamy modyfikować oraz modyfikujemy opcje "ddns" i uzyskujemy w efekcie :

```
option domain-name "domek.pl";
option domain-name-servers 192.168.1.1;
log-facility local0;
# ddns-updates off;
ddns-update-style ad-hoc;

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.10 192.168.1.250;
    option routers 192.168.1.1;
    default-lease-time 600;
    max-lease-time 86400;
}
key "DHCP_KEY" {
    algorithm hmac-md5;
    secret "HSlW3SamIqnczargtJjSbg==";
```

```
}
zone domek.pl. {
        primary 127.0.0.1;
        key DHCP_KEY;
}
zone 1.168.192.in-addr.arpa. {
        primary 127.0.0.1;
        key DHCP_KEY;
}
```

Ostatnim krokiem jest oczywiście restart serwisu `dhcpd` i już możemy się cieszyć w pełni funkcjonalnym serwerem.

15.4 Konfiguracja usług pokrewnych

15.4.1 Logi systemowe

Ponieważ hosty w naszej sieci mają dynamicznie przydzielane adresy IP warto wiedzieć, który host i kiedy otrzymał określony adres. Informacja ta jest szalenie istotna, szczególnie w przypadku dochodzenia, kto "rozrabia" w sieci. Pierwszy element konfiguracji już mamy, `log-facility local0` ustawia grupę, według której syslog będzie widział komunikaty napływające od demona dhcpd. Kolejnym elementem jest modyfikacja pliku konfiguracyjnego `/etc/syslog.conf`. Do pliku tego należy dopisać linię :

```
local0.*   /var/log/dhcp/dhcp.log
```

Teraz tworzymy plik `/var/log/dhcp/dhcp.log` i wykonujemy restart sysloga pleceniem :

```
[root@dreptak root]# service syslog restart
```

15.4.2 Rotacja logów

W przypadku większego ruchu w sieci i dużej liczby hostów logi naszego serwera będą bardzo szybko rosły. Niekontrolowany rozrost plików logów stanowi zagrożenie dla stabilności systemu. Przeglądanie dużych plików w celu znalezienia konkretnej informacji jest raczej trudne. Z tych powodów jest wskazane zadbanie o właściwie wykonywaną rotację plików logów. Najprostszym rozwiązaniem tego problemu jest dopisanie pliku dhcp.log do listy plików sysloga. W tym celu należy otworzyć do edycji plik `/etc/logrotate.d/syslog` i w pierwszej linii tego pliku dopisać ścieżkę do naszego pliku `/var/log/dhcp/dhcp.log`.

15.5 Zaawansowane opcje konfiguracyjne

Do tej pory konfigurowaliśmy sieć w taki sposób, że występowały w niej hosty z adresami dynamicznymi oraz statycznymi. Nie uwzględniliśmy jednak kilku problemów, które mogą wystąpić w większych instalacjach.

Pierwszym problemem są hosty ze stałymi adresami IP. Wyobraź sobie, że musisz wykonać jakieś zmiany w konfiguracji sieci, które wymagają dokonania zmian w konfiguracji hostów. W naszej sieci mamy tylko dwa takie hosty, ale już tutaj występuje ten problem. Komputer naszego VIPa jest zajęty przez cały dzień, więc zmian nie możesz dokonać. Po godzinach pracy również istnieje problem z dostępem do tej maszyny ze względu na ograniczone możliwości dostępu do pomieszczenia.

Problem ten możemy jednak rozwiązać wykorzystując serwer DHCP. Pomocna będzie możliwość rezerwacji adresów. Aby z tej funkcjonalności skorzystać musimy wykonać tylko drobną modyfikację pliku konfiguracyjnego `dhcpd.conf`. Zanim jednak zabierzemy się do tego zadania musimy poznać MAC-adres karty sieciowej naszego VIPa. Aby nie zawracać mu głowy zrobimy to bezpośrednio na naszym serwerze. Logujemy się na konto roota i wykonujemy polecenie `arp` umożliwiające dostęp do tablicy MAC adresów znanych przez nasz serwer.

```
[root@dreptak root]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.1.3              ether   00:60:08:0B:7A:1C   C                     eth0
(...)
[root@dreptak root]#
```

Z tej linijki interesują nas kolumny *HWtype i HWaddress* zawierające typ interfejsów i ich MAC adresy dla znanych hostów. Może się zdarzyć, choć w naszym przypadku jest to mało prawdopodobne, że w tablicy arp nie ma w danej chwili informacji o jakimś interfejsie. Wynika to z tego, że tablica ta jest co pewien czas odświeżana i przechowywane w niej są tylko te adresy, z którymi ostatnio był jakiś kontakt. W takim przypadku wystarczy uruchomić zwykły `ping` i informacje pojawią się w tablicy.

Mając takie informacje możemy przystąpić do modyfikacji plik `dhcpd.conf`. W tym celu otwieramy tan plik w trybie edycji i dodajemy poniższą sekcję :

```
host adam {
    hardware ethernet 00:60:08:0B:7A:1C;
    fixed-address 192.168.1.3;
    option routers 192.168.1.1;
}
```

Teraz trzeba jeszcze podejść do tego komputera i zmienić w nim ustawienia sieci. Robimy to tylko raz i już zawsze ten komputer będzie samodzielnie aktualizował sobie te ustawienia przy starcie systemu.

Nieco inaczej sytuacja się przedstawia w przypadku uruchamiania nowego komputera, któremu chcemy przydzielić stały adres. W takim przypadku uruchamiamy ten komputer jako zwykłą stację z dynamicznym adresem IP, odczytujemy jego MAC adres i dalej postępujemy analogicznie do wcześniejszego opisu. Jeśli możemy poznać MAC adres w inny sposób (np. lokalnie na przygotowywanej maszynie), to możemy pominąc pierwszy krok.

Wspomniałem na początku o kilku problemach. Zajmijmy się zatem drugim z nich. Nawet w tak małej sieci jak nasza pracownia występują komputery, na których praca odbywa się w różnych trybach. W jednej pracowni zajęcia trwają godzinę, w innej dwie godziny, w pokoju instruktorów komputer pracuje przez kilka/kilkanaście godzin bez przerwy. Biorąc powyższe pod uwagę należy zastanowić się nad zróżnicowaniem czasu dzierżawy adresu IP w poszczególnych lokalizacjach.

Uwaga. Zakończenie czasu dzierżawy nie jest równoznaczne z koniecznością zmiany adresu IP. W momencie, gdy host kliencki wysyła zapytanie do serwera ten w pierwszej kolejności sprawdza czy dla tego hosta była już przydzielana jakaś dzierżawa i, o ile to możliwe, przyznaje ten sam adres IP.

Aby zróżnicować sposób przydzielania adresów w naszej sieci musimy ją podzielić na tzw. pule adresowe. Aby host otrzymywał adres z danej puli musimy go w jakiś sposób związać z grupą hostów o podobnym przeznaczeniu. Możemy to wykonać wykorzystując mechanizm klas i podklas, a następnie każdej puli przydzielić odpowiednią klasę hostów. My jednak zrobimy to

nieco inaczej wyjaśniając przy okazji pewne pojęcie. Zacznijmy zatem od zmodyfikowania naszego pliku konfiguracyjnego `dhcpd.conf`. Modyfikować będziemy sekcję `subnet` do następującej postaci :

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers 192.168.1.1;
    # pula adresowa trenerow
    pool {
      range 192.168.1.10 192.168.1.20;
      max-lease-time 86400;
      deny unknown clients;
    }
    # pula adresowa w salach
    pool {
      range 192.168.1.21 192.168.1.250;
      max-lease-time 2700;
      allow unknown clients;
    }
}
```

I właściwie wszystko jest jasne poza określeniem *"unknown clients"*. Oznacza ono dosłownie *"nieznany klient"*. W pierwszym przypadku zabraniamy takim klientom korzystania z puli adresowej (`deny`), a w drugim zezwalamy. Jednak które hosty są dla demona `dhcpd` klientami znanymi, a które nie? Odpowiedź jest prosta : klient znany to taki, dla którego istnieje sekcja `host`. Jak pamiętasz z wcześniejszej lektury sekcję tą wykorzystywaliśmy do rezerwowania adresów. Teraz jednak adresów nie będziemy rezerwować. Wystarczy aby dla hosta, który ma być *"znany"* podamy MAC adres. Zatem do naszego pliku konfiguracyjnego dodajemy wpisy dotyczące hostów trenerskich :

```
host psor1 { hardware ethernet 00:60:08:0B:7A:1C; }
host psor2 { hardware ethernet 00:60:08:0B:7B:1D; }
```

Teraz jeszcze restart demona `dhcpd` i już mamy gotowy serwis. Należy jednak pamiętać, że tematy poruszone w tym rozdziale, to tylko wierzchołek góry lodowej. Wspomniałem wprawdzie o kategoriach, ale temat ten jest tak obszerny, że można by z niego utworzyć odrębny rozdział. Nie został również omówiony problem przekazywania wywołań dhcp przez routery. Należy pamiętać, że zapytanie dhcp, to zapytanie typu broadcast, czyli wysyłane do wszystkich, a takie pakiety nie są przekazywane przez routery. Aby to umożliwić potrzebne jest uruchomienie specjalnego programu o nazwie `dhcrelay`, ale to już temat dla bardziej zaawansowanych administratorów sieci teleinformatycznych.

Rozdział 16

Serwer ftp - vsftp

Kolejną usługą sieciową, którą się zajmiemy jest serwer ftp (File Transfer Protocol). Wprawdzie dostępnych jest wiele aplikacji umożliwiających udostępnianie plików ale w obecnie dostępnej wersji Auroksa mamy vsftpd (Very Secuire FTP Daemon) i skoncentrujemy się właśnie na tym demonie.

16.1 Konfiguracja podstawowa

Bezpośrednio po zainstalowaniu pakietu można uruchomić demona *vsftpd* i już mamy działający serwis. Należy jednak zastanowić się czy taka domyślna konfiguracja nam odpowiada. Elementy, na które warto zwócić uwagę to prawa dostępu/logowania z wykorzystaniem protokołu ftp, dopuszczalny transfer, rozmieszczenie katalogów i plików dostępnych przez ftp i jeszcze kilka innych mniej istotnych elementów. Podstawowe opcje konfiguracyjne ustawiamy w pliku `/etc/vsftpd/vsftpd.conf`. Pierwszą decyzją jaką musimy podjąć przy konfigurowaniu demona jest decyzja o udostępnianiu zasobów użytkownikom anonimowym. Opcja `anonymous_enable` ustawiona na "YES" zezwala na dostęp anonimowy i pociąga za sobą konieczność ustawienia jeszcze kilku opcji ale o nich będziemy mówić w następnym rozdziale. Analogicznie wygląda sprawa z użytkownikami lokalnymi. Opcja `local_enable` ustawiona na "YES" umożliwia użytkownikom naszego systemu na korzystanie z serwisu. Skoro zezwalamy lokalnym użytkownikom na korzystanie z ftp to należy również zadbać o to aby pliki wgrywane na serwer otrzymywały odpowiednie uprawnienia. W przypadku kont "shellowych" zastosowalibyśmy polecenie umask, tutaj musimy ustawić opcje `local_umask`. Domyślnie ustawiona jest na wartość 077 (pełne prawa dla właściciela i nic dla pozostałych) ale możesz to zmienić szczególnie jeśli upload'owane pliki miałyby być dostępne również dla innych użytkowników. Jeśli udostępniasz konta służące tworzeniu witryn www i bez dostępu do shella to zmiana taka jest wręcz konieczna bo pliki muszą być dostępne dla użytkownika apache. W takiej sytuacji radziłbym opcję tą ustawić na 022 czyli pełne prawa dla właściciela i prawo czytania dla grupy i innych.

Kolejną opcją, na którą musimy zwrócić uwagę jest `connect_from_port_20`. Ustawienie tej opcji na "YES" umożliwia dokonywanie transferu danych w trybie "normalnym", w przeciwnym przypadku demon będzie działał wyłącznie w trybie pasywnym a to może w niektórych przypadkach stwarzać problemy dla klientów.

Poruszając się po pliku konfiguracyjnym zapewne zauważyłeś kilka opcji zawierających w nazwie `xferlog`. Opcje te dotyczą sposobu logowania aktywności naszego serwera ftp. Wprawdzie opcje te są wstępnie ustawione na poprawne wartości ale warto sprawdzić czy w naszym przypadku będą odpowiadać potrzebom. Nazwa opcji pochodzi od nazwy standardu zapisu logów

rozpoznawanym np. przez programy wykonujące statystki serwera (taka aplikacja w dystrybucji jest webalizer). Zatem warto zadbać aby opcje `xferlog_std_format` i `xferlog_enable` były ustawiona na "YES".

Kolejne dwie opcje, nad którymi powinniśmy się zastanowić przedstawiam poniżej:

```
#idle_session_timeout=600
#data_connection_timeout=120
```

Jak widzisz opcje te są wyłączone co oznacza, że demon przyjmie wartości domyślne (to te, które są w tej chwili wpisane). Pierwsza z tych opcji odpowiedzialna jest za utrzymanie otwartej sesji w czasie gdy użytkownik nie wykonuje, żadnych operacji. Jeśli przez 600 sekund użytkownik nie wykona żadnej akcji to połączenie zostanie przerwane. Druga z tych opcji odpowiedzialna jest za czas połączenia przy przesyłaniu danych. Jeśli po wydaniu polecenie `get` (czytanie danych z serwera) lub `put` (przesyłanie danych na serwer) przez okres 120 sekund nie zostaną przesłane żadne dane to połączenie zostanie uznane za nieaktywne i sesja będzie zamknięta. Obie opcje mają wpływ na wydajność pracy serwera i powinny zapobiec atakowi typu "denial of service".

Skoro już jesteśmy przy kwestiach wydajnościowych to warto zainteresować się opcjami, których w pliku konfiguracyjnym nie ma a mają wpływ na wydajność. Zacznijmy od dwóch podobnych do siebie opcji: `anon_max_rate` oraz `local_max_rate`. Opcje te odpowiadają za maksymalną szybkość przesyłu danych podaną w bajtach na sekundę odpowiednio dla użytkowników anonimowych oraz lokalnych. Pamiętaj jednak, że jest to szybkość dla jednego połączenia a nie sumaryczna. Kolejną ciekawą opcją jest `max_clients` określająca maksymalną ilość połączeń do serwera ftp. Zbliżoną opcją jest `max_per_ip` przy czym nie dotyczy ona wszystkich połączeń lecz ilości połączeń pochodzących z jednego adresu IP. Przy ustawianiu tej opcji musisz jednak pamiętać, że część użytkowników pracuje za NAT'em lub serwerem proxy i przez nasz serwer widziani będą jako połączenia z tego samego IP. Z drugiej jednak strony powszechnym jest stosowanie "akceleratorów" ftp, które z założenia otwierają równocześnie kilka połączeń i pobierają plik w kilku częściach a następnie łączą to co znalazło się na lokalnym dysku. Tak więc ustawiając te opcje musisz brać pod uwagę wiele czynników równocześnie a i tak po pewnym czasie i obserwacji logów będziesz musiał je zmodyfikować.

Jeżeli ustawiłeś już odpowiednie opcje to teraz pozostaje już tylko przeładowanie serwisu i możesz przystąpić do testowania działania serwera.

16.2 Prawa dostępu

Jak już wcześniej wspomniałem w pliku konfiguracyjnym możemy określić czy do zasobów naszego systemu mają mieć dostęp użytkownicy lokalni i anonimowi. Możemy jednak znacznie dokładniej określić kto z lokalnych użytkowników ma prawo do korzystania z serwisu ftp. Interesują nas trzy opcje: `userlist_enable, userlist_deny` oraz `userlist_file`. Zacznijmy od ostatniej z tych opcji. Nie znajdziesz jej w pliku konfiguracyjnym więc demon korzysta z wartości domyślnej a w przypadku Auroksa jest to `/etc/vsftpd.user_list`. Proponowałbym pozostawić ten stan rzeczy bez zmian ale jeśli chcesz ... W pliku tym umieszczamy nazwy kont po jednym w linii natomiast znaczenie tak skonstruowanego pliku zależy od dwóch pozostałych opcji. Opcja `userlist_enable` wydaje się być oczywista. Jeśli ustawimy ją na "YES" to serwer będzie korzystał z wymienionego wcześniej pliku, jeśli ustawimy na "NO" to plik ten zostanie zignorowany. Natomiast ustawienie opcji `userlist_deny` decyduje o sposobie interpretacji listy użytkowników. Jeśli opcję tą ustawimy na "YES" to użytkownicy wymienieni w pliku nie będą mieli prawa korzystania z serwera ftp. Jeżeli jednak ustawimy ją na "NO" to tylko użytkownicy wymienieni w pliku `vsftpd.user_list` będą mieli prawo do logowania. Użytkownicy, którym odmówiono prawa do serwera ftp zostaną odrzuceni już po podaniu nazwy konta i nie będą

pytani o hasło.

Jednak tutaj spotyka nas pewna niespodzianka. Otóż w katalogu `/etc` znajduje się jeszcze jeden plik związany z użytkownikami i serwerem ftp. Plik, o którym mówię to `vsftpd.ftpusers`. Bezpośrednio po instalacji zawartość tych plików jest identyczna więc jaki jest cel tworzenia drugiego pliku? Otóż istnieje opcja `session_support`, która domyślnie ustawiona jest na "YES" a to oznacza, że nasz demon będzie korzystał z systemu PAM przy autoryzacji użytkownika. System ten sprawdzi zawartość pliku `vsftpd.ftpusers` i jeśli znajdzie w nim nazwę użytkownika, który właśnie próbuje się logować to próba taka zostanie uznana za nieudaną i użytkownik otrzyma komunikat o błędzie logowania.

Tak więc system *vsftpd* zapewnia nam podwójny system określania praw dostępu do zasobów serwera i o tym powinieneś pamiętać.

16.3 Dostęp anonimowy

Już wcześniej wspominałem o możliwości udostępnienia naszego serwisu ftp anonimowym użytkownikom. Aby to uczynić trzeba opcję `anonymous_enable` ustawić na "yes". I to jest podstawowa opcja ale w ślad za nią powinniśmy się zainteresować również innymi. Zaczniemy od lokalizacji plików naszego serwisu w strukturze katalogów naszego serwera. Za lokalizację odpowiada opcja `anon_root` i w Auroksie domyślnie ustawiona jest na katalog `/var/ftp`. Oczywiście tą lokalizację możemy zmienić ale wymaga to pewnych dodatkowych zabiegów. Wiadomo, że w systemie nie ma miejsca na anonimowe wyczyny i każde działanie wykonywane jest przez jakiegoś użytkownika posiadającego konto w systemie. W przypadku anonimowego użytkownika przypisywane mu są uprawnienia konta, na które wskazuje opcja `ftp_username` i w Auroksie jest to konto ftp. W momencie zalogowania się użytkownika anonimowego (jako login podajemy *anonymous* a hasło to adres e-mail) zostaje on przeniesiony do katalogu domowego czyli w Auroksie do katalogu `/var/ftp` (sprawdź polecenie `finger ftp`). Mając te podstawowe informacje możemy przystąpić do zmian w konfiguracji. Na potrzeby niniejszego ćwiczenia przyjmuję, że serwis ftp chcemy umieścić na osobnym dysku zamontowanym w katalogu `/data`. Zaczynamy zatem od zmiany w samym systemie:

```
[root@dreptak /]# usermod -d /data ftp
[root@dreptak /]# finger ftp
Login: ftp                              Name: FTP User
Directory: /data                        Shell: /sbin/nologin
Never logged in.
No mail.
No Plan.
[root@dreptak /]# ls -ld data
drwxr-xr-x  2 root root 4096 paź 31 19:45 data
```

Jak widzisz katalog domowy został zmieniony na `data` ale użytkownik ftp nie jest jego właścicielem. I bardzo dobrze. Przecież nie chcemy aby użytkownicy anonimowi mieli możliwość modyfikowania struktury naszego serwisu. Problemem jednak może być umieszczanie plików w naszym serwisie. Przy takich uprawnieniach tylko root może modyfikować zawartość tego katalogu a przecież całkiem prawdopodobne jest, że prowadzeniem serwisu zajmować się będzie osoba specjalnie do tego celu delegowana a nie administrator systemu. Oczywiście możemy temu zaradzić w bardzo prosty sposób. Serfując po internecie zapewne zauważyłeś, że wiele serwisów ftp zawiera katalog `pub` i dopiero w nim znajduje się to czego szukałeś. My zrobimy analogicznie w naszym serwisie:

```
[root@dreptak /]# cd data
[root@dreptak data]# mkdir pub
[root@dreptak data]# chown ftp.ftp pub
```

```
[root@dreptak data]# chmod 2775 pub
[root@dreptak data]# ls -ld pub
drwxrwsr-x  2 ftp ftp 4096 paź 31 19:53 pub
```

Właściwie powyższy ciąg poleceń mógłbym pozostawić bez komentarza ale tytułem przypomnienia... Polecenie `chmod 2775 pub` jest dość interesujące i często zapominane przez administratorów. Część 775 jest zapewne zrozumiała ale co oznacza ta dwójka na początku? Otóż ostawia ona bit sgid (set group id). Ustawienie tego bitu dal katalogu oznacza, że dla nowych plików automatycznie zostanie ustawiona grupa taka jak grupa katalogu. W naszym przypadku uzyskujemy pewność, że grupą dla nowych plików będzie grupa ftp a dzięki temu kilka osób będzie mogło prowadzić serwis wystarczy dodać je do tej grupy. Skoro mamy przygotowane środowisko to możemy teraz przystąpić do modyfikacji pliku `vsftpd.conf`. Aby nasze zmiany zaczęły funkcjonować musimy ustawić `anon_root=/data` i wykonać restart demona vsftpd.

Ale okazuje się, że nawet przy takiej konfiguracji możemy anonimowym użytkownikom ograniczyć dostęp do zasobów. Na ogół pliki w serwisie mają ustawione uprawnienia na 644 (rw-r–r–) i dzięki temu są dostępne dla odwiedzających nasz serwis. Nawet jeśli zmienimy uprawnienia na 640 to (rw-r––) to przecież użytkownik ftp również należy do grupy ftp więc będzie miał dostęp do tych plików mimo, że właścicielem jest ktoś inny (użytkownik prowadzący serwis). Możemy jednak w pliku `vsftpd.conf` ustawić opcję `anon_world_readable_only` na wartość "yes" i w ten sposób odcinamy anonimowych od plików, które nie mają ustawionego bitu r dla "others".

Istnieje również możliwość udostępnienia anonimowym użytkownikom uploadowania plików do naszego serwisu. Za tą funkcjonalność odpowiedzialne są dwie opcje: `anon_upload_enable` oraz `write_enable`. To tylko dwie podstawowe opcje związane z tym tematem. Ponieważ anonimowy upload jest bardzo ryzykownym przedsięwzięciem to temat ten pomijam i stanowczo odradzam uruchamianie takiego serwisu. To jest naprawdę niebezpieczne.

16.4 Środowisko chrootowane

Ostatnim tematem związanym z serwerem ftp jest kolejne zagadnienie ograniczania swobody użytkowników. Korzystając z różnych serwerów ftp zapewne zauważyłeś, że możesz poruszać się jedynie w obrębie katalogów udostępnianych przez ten serwis. W przypadku naszego anonimowego serwisu również takie zjawisko występuje. Jest to efekt działania polecenia `chroot` (od *change root*). Jednak w przypadku użytkowników posiadających konta w naszym systemie już takie ograniczenie nie działa i mogą oni poruszać się po całym drzewie katalogów. Często działanie takie jest pożądane ale w przypadku naszej pracowni może stwarzać pewne zagrożenia więc dobrze by było ograniczyć użytkowników do ich katalogów domowych. Za uruchomienie tej funkcjonalności odpowiedzialne są trzy opcje: `chroot_list_enable, chroot_local_user` oraz `chroot_list_file`. Najważniejsza opcja to `chroot_local_user`, ustawiamy ją na wartość "yes". Teraz kolej na opcję `chroot_list_file`. Tutaj należy podać ścieżkę do pliku zawierającego ... No właśnie, interpretacja zawartości tego pliku zależy od tego jak ustawimy wartość opcji `chroot_list_enable`. Jeśli ustawimy ją na wartość "yes" to użytkownicy, których wymienimy w pliku **nie będą** ograniczani do katalogu domowego natomiast wszyscy pozostali tak. Jeśli ustawimy ją na "no" to ograniczymy tylko tych , których wymienimy z nazwy na liście. Ale wróćmy do pliku listy. W pliku `vsftpd.conf` znajduje się zakomentowana linia, którą z powodzeniem możemy wykorzystać kasując początkowy # uzyskując w efekcie:

```
chroot_list_file=/etc/vsftpd.chroot_list
```

Pliku takiego nie mamy w naszym systemie więc powinniśmy go utworzyć:

```
[root@dreptak etc]# touch vsftpd.chroot_list
[root@dreptak etc]# ls -l vsftpd.chroot_list
-rw-r--r--  1 root root 0 paź 31 22:44 vsftpd.chroot_list
```

Teraz już możesz, w dowolnym edytorze tekstu, wprowadzić listę użytkowników.

Ale do czego to wykorzystać? Najbardziej oczywistym przykładem mogą być konta założone na potrzeby prowadzenia serwisu www. Użytkownikowi zakładamy konto bez dostępu do powłoki ale musimy umożliwić podkładanie plików html, grafik itp. Udostępnienie takiemu użytkownikowi serwisu ftp ograniczonego do jego katalogu domowego jest jak najbardziej poprawnym rozwiązaniem. Wprawdzie takie postępowanie nieco utrudnia nam konfigurację serwera www ale przecież katalog domowy dla takiego konta nie musi znajdować się w katalogu `/home` a to już znacznie upraszcza problem.

Rozważne stosowanie tej funkcjonalności może nam oddać nieocenione usługi.

Rozdział 17

Serwer stron www

Jedną z podstawowych usług, uruchamianych na serwerach internetowych jest serwis www. W rozdziale poniższym nie będziemy się zajmowali budowaniem stron www. Przyglądniemy się natomiast stronie technicznej uruchamiania serwisów www.

17.1 Konfiguracja podstawowa

Instalację serwera zaczynamy od zainstalowania pakietu httpd-<wersja>.i386.rpm. Ponieważ usługa www wiąże się z wieloma innymi cechami systemu warto do instalacji wykorzystać `apt-get`. Poza samym pakietem `httpd` warto zainstalować również :

- mod_ssl — moduł umożliwiający uruchomienie protokołu `https`, czyli szyfrowanej wersji serwisu
- mod_perl — moduł ten umożliwia tworzenie skryptów pisanych w języku perl i uruchamianie ich jako integralnej części serwera
- php — język skryptowy umożliwiający tworzenie dynamicznych stron www, w zależności od potrzeb funkcjonalnych planowanego serwisu warto instalować również pakiety php-mysql i php-imap

Ponieważ w dalszej części podręcznika będziemy zajmować się tematem tworzenia dynamicznych serwisów www, to już w tym miejscu zainstalujemy wszystko, co nam może być potrzebne.

```
[root@dreptak root]# apt-get install httpd mod_ssl php php-mysql
Reading Package Lists... Done
Building Dependency Tree... Done
(...)
```

Prawdopodobnie `apt` zasugeruje zainstalowanie jeszcze kilku pakietów. Przyjmujemy jego propozycje i teraz trzeba tylko czekać na instalację. I to właściwie już wszystko, co potrzebujemy do uruchomienia pierwszego serwisu www. Wystartujmy zatem nasz serwer i sprawdźmy czy działa. Jedna drobna uwaga. Po wykonaniu polecenia `telnet` można wpisać polecenia protokołu `http`. Nie będziemy się aż tak szczegółowo zajmować tym tematem. W tej chwili wystarczy nam informacja, że serwer odpowiedział i czeka na dalsze komendy. Tajemniczy znaczek `^]` oznacza `Ctrl` + `]` .

```
[root@dreptak root]# service httpd start
Uruchamianie httpd:                                        [   OK   ]
```

```
[root@dreptak root]# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
^]
telnet> quit
Connection closed.
[root@dreptak root]# netstat -at | grep http
tcp        0      0 *:http                  *:*                         LISTEN
tcp        0      0 *:https                 *:*                         LISTEN
tcp        0      0 dreptak.tupward.pl:1292 dreptak.tupward.pl:http TIME_WAIT
[root@dreptak root]#
```

Z powyższego wynika, ze serwer wystartował i nasłuchuje na portach 80 (`http`) i 443 (`https`). Czyli mamy to, co jest nam potrzebne. Kolejnym testem jest uruchomienie przeglądarki internetowej i sprawdzenie jaki efekt uzyskamy wpisując adres `http://localhost`. Jeśli wszystko działa, to powinieneś zobaczyć stronę testową.

Skoro mamy już działający serwer, to przyglądnijmy się dokładniej co gdzie jest. Oczywiście pierwszym krokiem jest rpm.

```
[root@dreptak root]# rpm -ql httpd | less
```

Jest kilka plików w tym pakiecie. Ale nie przejmuj się. Omówimy kolejno wszystkie istotne elementy. Zaczynamy od tego, co zapewne najbardziej Ciebie interesuje : gdzie umieścić stronę www. Naszą pierwszą stronkę umieszczamy w katalogu `/var/www/html`. Przechodzimy do tego katalogu i poleceniem `touch` tworzymy plik `index.html`. Plik ten musi być dostępny do czytania dla wszystkich, więc sprawdź czy ma ustawione uprawnienia 644 i ewentualnie popraw je. Teraz w dowolnym edytorze otwieramy plik i wpisujemy kod naszej pierwszej strony.

```
<html>
<head>
  <title>Pierwsza strona</title>
</head>
<body>
  <h1>
    <center>
      Hello world!!!
    </center>
  </h1>
</body>
</html>
```

I sprawdzamy co pokaże się w przeglądarce pod adresem `http://localhost`. Pojawiła się nasza stronka? Wygląda raczej nieciekawie, ale to tylko test funkcjonowania serwera, a nie kurs HTML.

Jak zapewne zauważyłeś do strony odwołujemy się tak jakby była w katalogu `/` mimo, że jest w `/var/www/html`. Za takie działanie odpowiedzialna jest dyrektywa pliku konfiguracyjnego serwera. Pliki konfiguracyjne znajdują się w katalogu `/etc/httpd`. Interesują nas dwa katalogi `conf` (ogólna konfiguracja serwera) oraz `conf.d` (konfiguracja serwerów wirtualnych i modułów). Poznawanie konfiguracji zaczniemy od pliku `conf/httpd.conf`. W pliku tym mamy wiele opcji i jeszcze więcej komentarzy. Postaram się pomóc Ci w poznaniu podstawowych elementów konfiguracyjnych. Otwieramy zatem plik `httpd.conf` i przewijamy go aż dotrzemy do pozycji `User` i `Group`. Pozycje te określają z jakimi uprawnieniami ma działać serwer. Demona `httpd` uruchamia użytkownik root. Ponieważ jednak w samym demonie mogą wystąpić luki bezpieczeństwa, a jeszcze bardziej jest to prawdopodobne w przypadku aplikacji internetowych, to udostępnianie zasobów z uprawnieniami roota byłoby co najmniej nieodpowiedzialne. Dlatego demon `httpd` nie obsługuje bezpośrednio wywołań przychodzących z internetu, tylko uruchamia podprocesy dzia-

łające z uprawnieniami użytkownika i grupy takimi jakie podano w wymienionych wyżej opcjach i dopiero te procesy obsługują nadchodzące wywołania. W przypadku dystrybucji Aurox Linux obie pozycje ustawione są na wartość `apache`. W praktyce możesz się jednak spotkać również z innymi ustawieniami. W bardziej tradycyjnych systemach uniksowych wielkości te ustawiano na `nobody/nogroup`. Cel takiego postępowania jest oczywisty : ograniczyć do minimum możliwość dostępu do systemu z poziomu serwera apache.

Kolejną istotną pozycją jest `ServerName`. Dla opcji tej należy ustawić nazwę kanoniczną oraz port dla jakich serwer będzie odpowiadał. Wpisujemy tutaj nazwę naszego serwera np. `ServerName dreptak.tupward.pl:80`. Jeśli korzystać będziemy z serwerów wirtualnych, to opcję tą należy wyłączyć wpisując `#` na początku linii.

Teraz dochodzimy do tego, co nas tutaj sprowadziło : opcja `DocumentRoot`. Opcja ta podaje katalog w systemie plików, który przez serwer będzie uznawany za początek drzewa katalogów. Wygląda to tak, jakby początek drzewa katalogów został przesunięty do podanego katalogu. Jak widzisz w oryginalnym pliku opcja ta ustawiona jest na `/var/www/html`. W starszych systemach możesz w tym miejscu znaleźć wpis `/home/html`. Jeśli planujesz postawienie większego ośrodka www, to warto się zastanowić nad wydzieleniem osobnego filesystemu na te potrzeby. W takim przypadku będziesz musiał dokonać zmiany właśnie w tym miejscu.

Kolejną istotną opcją jest `DirectoryIndex`. Tutaj podajemy listę plików, które serwer ma traktować jako pliki startowe danego katalogu. Nazwa index może być nieco myląca. Jest to pozostałość bardzo starych wersji serwera, w których trzeba było tworzyć specjalne pliki zawierające indeks plików znajdujących się w danym katalogu. Obecnie znaczenie pliku "indeksującego" jest nieco inne. Wpisując pewien adres w przeglądarce internetowej powinniśmy podawać plik zawierający stronę html. Jeśli nazwy tego pliku nie podamy i poprzestaniemy na katalogu, to serwer nie będzie wiedział co ma udostępnić. Wprawdzie można apacha przekonać do tego, aby w takim przypadku pokazywał listę plików, ale takie postępowanie ułatwia złym chłopcom dotarcie do źródeł naszej strony lub, co gorsze, do aplikacji internetowej. Dlatego właśnie należy deklarować nazwę pliku indeksującego. Serwer przeszukuje podany w wywołaniu katalog w poszukiwaniu pliku z listy plików indeksowych. Przyjęło się, że plik indeksowy to `index.html`, ale z czasem zaczęto stosować również inne pliki. Częściej spotykane to : `index.htm` (DOS pozwala tylko na trzy znaki rozszerzenia), `index.shtml` (plik html zawierający polecenia dla serwera), `default.htm` (pomysł znanej firmy z Redmont). Możesz oczywiście przyjąć jakąś własną zasadę, ale jeśli z Twojego serwera będą korzystać inne osoby może to prowadzić do nieporozumień.

Kolejne dwie opcje są nieco zbliżone. `ErrorLog` i `CustomLog` wskazują położenie plików z logami serwera. Zrozumienie wpisów jakie tu znajdziemy wymaga jednak cofnięcia się do początku pliku konfiguracyjnego, do opcji `ServerRoot`. Opcja ta podaje katalog, względem którego będą określane ścieżki dostępu w dalszej części pliku. Ponieważ opcja ta zawiera wpis `/etc/httpd` to zapis dla `ErrorLog` w postaci `logs/error_log` należy rozumieć jako `/etc/httpd/logs/error_log`. W rzeczywistości pliki logów znajdują się w katalogu `/var/log/httpd`. Wyjaśnienie jest proste :

```
[root@dreptak httpd]# ls -ld /etc/httpd/logs
lrwxrwxrwx  1 root root 19 cze  7 21:49 /etc/httpd/logs -> ../../var/log/httpd
[root@dreptak httpd]#
```

Zwróć uwagę, że opcja `CustomLog` może wystąpić wiele razy. Jako parametry tej opcji podajemy nazwę pliku z logami oraz rodzaj logów jakie mają być gromadzone w danym pliku. Różne rodzaje logów określamy wykorzystując opcję `LogFormat`. Po szczegóły dotyczące formatowania logów odsyłam dociekliwych do dokumentacji serwera.

To już wszystko jeśli chodzi o konfigurację podstawowych parametrów pracy serwera. Pozostaje jeszcze zadbać, aby serwer uruchamiał się automatycznie przy starcie systemu. Zapewne już masz swoją ulubioną metodę konfigurowania serwisów. Ja jednak pozostanę wierny konsoli i

wpiszę :

```
[root@dreptak httpd]# chkconfig --level 35 httpd on
```

17.2 Strony prywatne użytkowników

Powyższy opis przedstawia sytuacje uruchomienia pojedynczego ośrodka www. Jednak w wielu przypadkach na jednej maszynie chcemy uruchomić wiele stron, a w szczególności strony domowe użytkowników naszego systemu. Najprostszym rozwiązaniem jest utworzenie odpowiednich katalogów w katalogu `/var/www/html`. Aby ułatwić użytkownikom dotarcie do odpowiedniego katalogu z plikami stron możemy w jego katalogu domowym utworzyć link.

```
[root@dreptak root]# cd /var/www/html
[root@dreptak html]# mkdir tuptus
[root@dreptak html]# chown tuptus tuptus
[root@dreptak html]# chmod 755 tuptus
[root@dreptak root]# ln -s /var/www/html/tuptus /home/tuptus/strona_www
```

Teraz do stron użytkownika *tuptus* można się odwołać podając adres *http://dreptak.tupward.pl/tuptus* natomiast użytkownik *tuptus* może modyfikować pliki strony tak jakby były w podkatalogu `strona_www` jego katalogu domowego.

Rozwiązanie powyższe ma jednak kilka wad. Po pierwsze użytkownik musi mieć możliwość poruszania się po całym systemie plików. Jeśli na naszym serwerze zakładamy konta dla użytkowników nie do końca zaufanych, to jest to pewne ryzyko. Jeśli jeszcze chcemy, aby użytkownik miał konto bez dostępu do powłoki, a modyfikację plików wykonywał przez FTP, to tym bardziej powinniśmy ograniczyć mu dostęp wyłącznie do jego katalogu domowego. Pewnym wyjściem jest stosowanie polecenia `mount`. W takim przypadku nie tworzymy linku tylko katalog o odpowiedniej nazwie i podmontowujemy katalog ze stroną :

```
[root@dreptak root]# cd /home/tuptus
[root@dreptak tuptus]# mkdir strona_www
[root@dreptak tuptus]# chown tuptus strona_www
[root@dreptak tuptus]# mount --bind /var/www/html/tuptus /home/tuptus/strona_www
```

Teraz użytkownik ma dostęp do plików bez konieczności wychodzenia poza katalog domowy. To jednak nadal nie jest rozwiązanie idealne, gdyż polecenia `mount` dla wszystkich tworzonych w ten sposób witryn należy wpisać do pliku `/etc/rc.d/rc.local`. Dodatkowym problemem może być ilość montowanych filesystemów. `mount` może w pewnym momencie odmówić posłuszeństwa z komunikatem *"to many mounted filesystems"*. Tak więc rozwiązanie to jest dobre, ale dla małej ilości użytkowników.

Kolejnym problemem może być kontrola zajętości filesystemów. Jeśli na naszym serwerze mamy uruchomioną `quote`, to trzeba ją skonfigurować również dla filesystemu, na którym znajdują się pliki stron naszych użytkowników. Więc kolejny element wymagający stałego nadzoru administratora i kolejne zmartwienie.

Najlepszym rozwiązaniem byłoby umieszczanie stron domowych w katalogach domowych użytkowników. Pojawiają się tutaj dwa problemy. Użytkownik `apache` nie ma dostępu do katalogów domowych i lepiej żeby nie miał, bo każdy użytkownik internetu miałby dostęp do prywatnych plików użytkowników. I drugi problem to określenie ścieżki dostępu do plików witryny. Chodzi o to, żeby po wpisaniu adresu *http://dreptak.tupward.pl/tuptus* apache wiedział, że ma sięgnąć do `/home/tuptus/strona_www`, a nie `/var/www/html/tuptus`.

Pierwszy problem możemy rozwiązać następująco : zakładamy odpowiedni katalog w katalogu domowym użytkownika i zmieniamy uprawnienia tak, aby użytkownik `apache` mógł czytać zawartość tego katalogu.

```
[root@dreptak root]# cd /home/tuptus
[root@dreptak tuptus]# mkdir strona_www
[root@dreptak tuptus]# chown tuptus.apache strona_www
[root@dreptak tuptus]# chmod 2750 strona_www
[root@dreptak tuptus]# chmod o+x /home/tuptus
```

Teraz grupa `apache` ma dostęp do zawartości katalogu `/home/tuptus/strona_www`, ale nie może czytać zawartości katalogu domowego użytkownika *tuptus*, może jedynie przez niego przejść. Pozostaje jeszcze poinformować demona `httpd` o naszych zamiarach. Przechodzimy zatem do pliku `/etc/httpd/conf/httpd.conf` i dopisujemy na końcu :

```
Alias /tuptus "/home/tuptus/strona_www"
```

Teraz jeszcze restart demona `httpd` i już użytkownik *tuptus* ma swoją stronę www.

Tak. Tylko dlaczego administrator ma się tak trudzić. Dla kilku użytkowników to zgoda, ale jeśli na serwerze jest pięciuset? Na szczęście twórcy apacha przewidzieli taką sytuację. Otwieramy jeszcze raz plik `/etc/httpd/conf/httpd.conf` i odszukujemy dyrektywę `UserDir disable`. Wystarczy ja zmienić na `UserDir strona_www` i wykonać restart demona `httpd`. Od tej chwili każdy użytkownik naszego systemu może sam sobie stworzyć środowisko potrzebne do uruchomienia strony domowej.

```
[tuptus@dreptak tuptus]# mkdir strona_www
[tuptus@dreptak tuptus]# chmod 2755 strona_www
[tuptus@dreptak tuptus]# chmod o+x /home/tuptus
```

Do tak utworzonej strony możemy się odwoływać podając adres *http://dreptak.tupward.pl/ ∼tuptus*.

Uwaga. Oczywiście nazwa katalogu `strona_www` została przyjęta na potrzeby niniejszego kursu. Przyjęło się, że katalog ze stroną domową nosi nazwę `public_html` i rzadko się zdarza, aby administrator przyjmował inne rozwiązanie.

17.3 Serwery wirtualne

Stawianie komputera na potrzeby tylko jednego ośrodka www to zwykłe marnotrawstwo. Problem ten dostrzeżono już dawno i wymyślono pojęcie serwerów wirtualnych. Początkowo uruchomienie serwerów wirtualnych wymagało posiadania wielu adresów IP i dla każdego adresu uruchamiano osobny ośrodek (do tej pory technika ta stosowana jest w przypadku serwerów ftp). Szybko jednak okazało się, że technika ta ma ogromną wadę — zaczęło brakować adresów IP. W związku z tym, że większość czytelników tego podręcznika ma do dyspozycji tylko jeden adres IP tematem tym nie będziemy się dokładniej zajmować.

Istnieje jednak inna możliwość uruchamiania serwerów wirtualnych. Rozwiązaniem jest stosowanie nazw dla hostów mapowanych przez serwery DNS na adresy IP. Demon `httpd` nasłuchuje na określonym adresie IP, ale jednocześnie umie rozpoznać jaką nazwę serwera podano w jego wywołaniu. Dzięki tej właściwości możemy tworzyć różne ośrodki www udostępniane dla różnych nazw serwera. Rozwiązanie takie nazywane jest *"name-based virtual hosting"* czyli wirtualne hosty bazujące na nazwie.

Wiemy już co chcemy osiągnąć, więc przejdźmy do praktycznej realizacji tego zadania. Pierwszym krokiem powinno być ustawienie odpowiednich wpisów w serwerze DNS. Konfiguracja tego serwera jest opisana w innym rozdziale, tutaj tylko ogólnie. Uruchomienie serwera wirtualnego

wymaga dodania rekordu `CNAME` do pliku strefy. W naszym wypadku należy zmodyfikować plik `tupward.pl.zone`.

```
dreptak                 A       192.168.1.1
www                     CNAME   dreptak
```

Po restarcie demona named odwołanie pod adres *www.tupward.pl* wskazywać będzie na ten sam adres co *dreptak.tupward.pl*, czyli *192.168.1.1*.

Teraz kolej na ośrodek www. Zaczniemy od utworzenia osobnego katalogu na potrzeby naszego nowego ośrodka. W zasadzie możemy to zrobić w dowolnym miejscu systemu plików, ale ja lubię mieć porządek w systemie, więc ośrodek zakładam w katalogu `/var/www` i użytkownika *tuptus* przydzielam do administrowania tym serwisem :

```
[root@dreptak root]# cd /var/www
[root@dreptak www]# mkdir osrodek
[root@dreptak www]# chown tuptus osrodek
[root@dreptak www]# chmod 755 osrodek
```

Teraz użytkownik *tuptus* może w tym katalogu tworzyć strony www.

Kolejnym krokiem jest poinformowanie demona `httpd` o zaistniałych zmianach. Wprawdzie potrzebne wpisy można umieszczać bezpośrednio w pliku `httpd.conf`, ale ja preferuję nieco inne rozwiązanie. W katalogu `/etc/httpd` znajduje się katalog `conf.d`. Wszystkie pliki, których nazwy kończą się ciągiem `conf` i znajdują się w tym katalogu są dołączane do pliku `httpd.conf` w czasie uruchamiania demona. Tworzymy zatem plik `default.conf`, w którym znajdować się będzie konfiguracja serwera domyślnego oraz plik `www.conf`, w którym zapiszemy konfigurację naszego ośrodka www.

Konfigurację zaczynamy od pliku `conf/httpd.conf`. W pliku tym odnajdujemy dyrektywę konfigurującą korzystanie z serwerów wirtualnych i modyfikujemy ją do postaci :

```
NameVirtualHost 192.168.1.1:80
```

Teraz przechodzimy do pliku `conf.d/default.conf` i wpisujemy jego konfigurację :

```
<VirtualHost _default_:80>
    ServerAdmin root@tupward.pl
    DocumentRoot /var/www/html
    ServerName dreptak.tupward.pl
</VirtualHost>
```

I teraz to co nas najbardziej interesuje : plik `conf.d/www.conf` :

```
<VirtualHost 192.168.1.1:80>
    ServerAdmin tuptus@tupward.pl
    DocumentRoot /var/www/osrodek
    ServerName www.tupward.pl
#    ErrorLog logs/osrodek-error_log
#    CustomLog logs/osrodek-access_log combined
</VirtualHost>
```

W powyższym przykładzie przyjąłem, że logi naszego ośrodka będą trafiać do tych samych plików co logi serwera domyślnego. Jeśli chcesz logi umieszczać w osobnych plikach (zalecałbym takie postępowanie), to należy usunąć początkowe znaki `#`.

Teraz należy wykonać restart demona `httpd` i możemy przetestować działanie naszego rozwiązania. Oczywiście podane pliki konfiguracyjne to tylko podstawowe opcje pozwalające na uruchomienie serwera wirtualnego. Do tych opcji warto jeszcze dodać dodatkowe dyrektywy podnoszące bezpieczeństwo serwera oraz modyfikujące prawa dostępu.

17.4 Logi serwera www, Webalizer

17.5 Moduły serwera

17.5.1 Moduł php

17.5.2 Moduł mod_perl

Rozdział 18

Serwer plików

Potęgą systemów informatycznych jest stosowanie szeroko rozumianych rozwiązań sieciowych. Jednym z takich rozwiązań jest możliwość rozproszenia plików na różne maszyny i swobodny dostęp do nich. Dzięki zastosowaniu takiego mechanizmu możemy w łatwy sposób rozmieścić pliki w zależności od ich znaczenia i wymaganego poziomu dostępu. Możemy pliki instalacyjne umieścić na serwerze, z którego użytkownicy będą mogli w łatwy sposób je pobrać. Możemy na serwerze utworzyć specjalny katalog dla ważnego projektu i zapewnić mu odpowiedni poziom bezpieczeństwa, w tym również backup. O ile w pierwszym przypadku doskonale będzie się spisywał poznany już serwer FTP to dla realizacji drugiego zadania trzeba znaleźć inne rozwiązanie. Aby użytkownicy mogli swobodnie pracować na plikach projektu trzeba im umożliwić dostęp do plików w taki sam sposób jak gdyby pliki te znajdowały się na lokalnym dysku. Rozwiązań tego problemu jest kilka, my zajmiemy się tylko dwoma najbardziej rozpowszechnionymi:

- NFS — Network File System czyli sieciowy system plików, rozwiązanie tak stare ja UNIX, bardzo proste w implementacji ale posiadające poważne braki bezpieczeństwa
- Samba — sieciowy system plików stosowany głównie w sieciach Net Bios, oparty na protokole smbfs

Stosowanie tego typu rozwiązań stwarza jednak poważne zagrożenia dla poufności danych udostępnianych w sieci a także dla samych serwerów. Dlatego podstawą dla tego typu instalacji jest wydzielenie odrębnej podsieci fizycznej, w której znajdować się będą zarówno serwer jak i stacje klienckie, niedostępnej dla innych użytkowników. Oczywiście nie można odcinać od świata pracowników przygotowujących dany projekt ale dzięki zastosowaniu filtrowania na routerze możemy te kontakty ograniczyć do niezbędnego minimum a w szczególności odseparować sieciowe systemy plików od dostępu spoza tej wydzielonej sieci. Może to w tej chwili brzmi dość groźnie ale za chwilkę sam się przekonasz, że takie podejście nie jest pozbawione podstaw. Przejdźmy zatem do szczegółów.

18.1 Usługa NFS

Jak już wcześniej wspomniałem jest to usługa bardzo stara i bardzo prosta. Zaczynamy od przygotowania serwera. Potrzebne nam będą pakiety NFS oraz portmap a także kilku innych powiązanych zależnościami. Kernel linuksa też musi być odpowiednio przygotowany ale w przypadku Auroksa ten problem rozwiązali już twórcy dystrybucji.

Zajmiemy się teraz udostępnianiem zasobów na serwerze. Zadanie to jest stosunkowo proste. Potrzebny nam jest plik `/etc/exports`, do którego będziemy wpisywać informacje o udostępnionych zasobach oraz parametry tego udostępniania. Przykładowa linia tego pliku może wyglądać tak:

```
/opt/nfs_share          192.168.1.0/255.255.255.0(rw)
```

Pierwszym elementem w tej linii jest udostępniany zasób, w tym wypadku katalog `/opt/nfs_share`. Kolejnym elementem jest opis komu i w jaki sposób dany zasób udostępniamy. W tym przypadku zasób będzie dostępny dla wszystkich hostów z sieci 192.168.1.0/24 w trybie odczytu i zapisu. Gdybyśmy chcieli ograniczyć prawa dostępu wyłącznie do odczytu wystarczy w nawiasie zmienić zapis *rw* na *ro*.

Jak widzisz nie ma tu żadnego parametru odpowiedzialnego za identyfikację użytkownika. Jak zatem określane są prawa do stępu? Opcja w nawiasie to jedna metoda ale wykorzystywana jest dla zasobu jako całości a to trochę mało. Przecież w linuksie możemy określać prawa dla właściciela, grupy i innych. Otóż NFS również korzysta z tego mechanizmu. Przy każdej próbie dostępu do pliku z *klienta* pobierany jest numer id użytkownika i przekazywany do *serwera*, który na tej podstawie określa uprawnienia. Zauważyłeś już problem? Tak, problem polega na tym, że do poprawnego funkcjonowania mechanizmu uprawnień należy zadbać o to aby w obu systemach użytkownicy mieli te same numery id. Jeśli nie zadbamy o ten szczegół może dojść do poważnych nadużyć.

Jeszcze innym problemem jest konto root. Zwróć uwagę, że root na *kliencie* automatycznie uzyskuje uprawnienia roota na *serwerze*. Czasami właśnie o to nam chodzi ale w większości wypadków byłaby to luka w systemie bezpieczeństwa. Dlatego domyślnym działaniem serwera NFS jest zmiana id=0 na id użytkownika *nobody*. Właściwość ta jest domyślna ale dla przejrzystości warto w nawiasie dodać opcję `root_squash`, która nic nie zmieni ale będzie nam przypominać co właściwie się dzieje w przypadku połączenia dokonanego przez roota. Jeśli jednak chcemy aby właśnie root miał dostęp przez NFS to opcję tą zmieniamy na `no_root_squash`. Oczywiście w takim przypadku radziłbym zasób udostępniać konkretnemu komputerowi a nie całej sieci.

Skoro mamy już przygotowaną konfigurację to możemy wystartować serwer. Robimy to jak zwykle poleceniem `service nfs start`. Jest jednak pewna różnica w stosunku do innych serwerów. Jeśli dokonamy jakichś modyfikacji w pliku `/etc/exports` to nie musimy restartować całego serwisu. Wystarczy wykonać polecenie `exportfs -a` i zmiany wejdą w życie.

Zajmiemy się teraz stroną klienta. Polecenie `mount` już znasz. W przypadku NFS ma ono jednak nieco inną składnię. Jeśli z jakiegoś hosta w naszej sieci chcemy zamontować do katalogu `/mnt/test` zasób udostępniany przez serwer *dreptak.tupward.pl* to wydajemy polecenie:

```
# mount -t nfs dreptak.tupward.pl:/opt/nfs_share /mnt/test
```

Jak widzisz trzeba było podać jawnie typ montowanego zasobu (*-t nfs*) a zamiast urządzenia podajemy *<host>:<path>*. Oczywiście możemy dodać odpowiedni wpis do pliku `/etc/fstab` i użyć skróconej formy montowania. Linia taka wygląda następująco:

```
dreptak.tupward.pl:/opt/nfs_share  /mnt/test  nfs  noauto
```

Oczywiście gdybyś chciał aby ten zasób montował się automatycznie przy starcie systemu to opcję *noauto* zmieniasz na *auto*.

I wszystko by było pięknie gdyby nie dwa drobiazgi utrudniające nap całą zabawę. Po pierwsze tylko root może dokonywać montowania zasobów. Może w tym momencie się nieco zdziwiłeś, przecież użytkownik może montować krążki CD i dyskietki. Tak, może ale zwróć uwagę, że muszą być spełnione pewne warunki:

1. użytkownik musi mieć prawo pisania do katalogu, do którego montuje zasób

2. w pliku fstab musi być ustawiona opcja *owner* (użytkownik musi być właścicielem zasobu) lub *users*

O ile pierwszy z tych warunków jest prosty do spełnienia to z drugim mogą być problemy. Drugim "drobiazgiem" jest ilość zasobów dostępnych w sieci oraz ilość użytkowników chętnych do skorzystania z nich. Wyobraź sobie sytuację, że w sieci jest udostępnione dwadzieścia różnych zasobów i do tego na różnych hostach. Na hoście *kliencie* masz stu użytkowników, z których część chciałaby korzystać z tych zasobów najlepiej montując je w katalogach domowych bo tak wygodniej. Policzyłeś ile wpisów musiałbyś zrobić w pliku *fstab*? Nawet gdybyś zrobił te wszystkie wpisy to następnego dnia musiałbyś je poprawiać bo któremuś użyszkodnikowi zachciało się zmienić punkt montowania. Ale i z tym problemem sobie poradzimy. Wykorzystamy do tego celu polecenie `sudo`.

18.1.1 Polecenie sudo

Polecenie `sudo` pochodzi z pakietu o tej samej nazwie, jeśli jeszcze go nie zainstalowałeś to najwyższa pora abyś to zrobił. Działanie tego polecenia jest podobne do wykonania polecenia `su -c <polecenie>` ma jednak tę przewagę, że można je bardzo dokładnie skonfigurować podając kto może z niego korzystać i w jakim zakresie.

Uwaga. Polecenie sudo zmienia jedynie identyfikator użytkownika (uid) i nie zmienia parametrów środowiska. W szczególności zmienna środowiskowa PATH pozostaje bez zmian. Można to zmienić ustawiając odpowiednie parametry w pliku `/etc/sudoers`.

Ogólna składnia polecenia wygląda następująco:

```
sudo <polecenie>
```

W momencie wywołania takiego polecenia następuje sprawdzenia pliku `/etc/sudoers` i jeśli *polecenie* pasuje do jednego z wzorców znajdujących się w tym pliku zostanie ono wykonane tak jakby wykonywał je użytkownik podany w pliku konfiguracyjnym (domyślnie jest to root). Pliku `/etc/sudoers` jednak nie modyfikujemy bezpośrednio lecz przez polecenie `visudo` (plik konfiguracyjny zostaje otwarty w domyślnym edytorze). Struktura pliku `sudoers` jest stosunkowo prosta choć niektóre reguły mogą mieć dość skomplikowany wygląd. W pierwszej kolumnie umieszczamy informację komu nadajemy dane uprawnienie. Może to być nazwa pojedynczego konta, lista kont oddzielona przecinkami lub nazwa grupy przy czym w tym wypadku nazwa grupy musi być poprzedzona `%`.

Dalej wprowadzamy informacje o tym co danemu użytkownikowi wolno. Zaczynamy od podania nazwy hosta (lub listy hostów) , na których dane polecenie będzie wykonywane. W naszym przypadku zastosujemy alias `ALL`, który jest wbudowany w sudo i przy dopasowywaniu polecenia zawsze zwraca prawdę. Dalej wstawiamy `=` i w nawiasie nazwę konta, na którym dane polecenie ma być wykonane. Jeśli nie podamy konta to domyślnie zostanie przyjęte konto root. Dalej możemy podać opcję `NOPASSWD:`, która zwalnia użytkownika z obowiązku podania hasła przed wykonaniem polecenia. Jako ostatni element podajemy listę pleceń oddzielonych przecinkami. W przypadku podawania poleceń można stosować znaki wieloznaczne takie jak `*,?,+`.

Uzbrojeni w taką wiedzę możemy przystąpić do rozwiązania naszego problemu. Sprecyzujmy zatem co mamy do zrobienia. Chcemy aby użytkownicy mogli samodzielnie montować i odmontowywać zasoby sieciowe NFS w katalogach wewnątrz swoich katalogów domowych. Ponieważ nie wszyscy będą na tyle uświadomieni aby dali sobie radę z podaniem tak skomplikowanej komendy na konsoli to zakres uprawnień ograniczymy do grupy, którą nazwiemy *power_users*. Zaczynamy więc od założenia odpowiedniej grupy i przypisania do niej użytkowników:

```
[root@dreptak root]# groupadd power_users
[root@dreptak root]# usermod -G power_users tuptus
[root@dreptak root]# id tuptus
uid=500(tuptus) gid=500(tuptus) grupy=500(tuptus),510(power_users)
```

Jak widzisz użytkownik *tuptus* został dodany do nowej grupy. Wykorzystując polecenie `usermod` możesz do tej grupy dodać kolejnych użytkowników. Teraz kolej na modyfikację pliku `/etc/sudoers`. Wykonujemy polecenie `visudo` i dodajemy na końcu pliku dwie linie:

```
%power_users    ALL = NOPASSWD: /bin/umount /home/*/*
%power_users    ALL = NOPASSWD: /bin/mount -t nfs * /home/*/*
```

Teraz możemy przetestować działanie naszego rozwiązania:

```
[tuptus@dreptak tuptus]$ mkdir nfs
[tuptus@dreptak tuptus]$ sudo mount -t nfs dreptak.tupward.pl:/opt/nfs_share \
/home/tuptus/nfs
[tuptus@dreptak tuptus]$ mount
dreptak.tupward.pl:/opt/nfs_share on /home/tuptus/nfs type nfs (rw,addr=192.168.1.1)
[tuptus@dreptak tuptus]$ sudo /bin/umount /home/tuptus/nfs
```

Jak widzisz u mnie zadziałało idealnie. Jeśli masz jakieś problemy to dokładnie sprawdź wpisy w pliku `/etc/sudoers`, plik ten jest bardzo wrażliwy na literówki.

18.2 Klient sieci smbfs

18.3 Serwer sieci smbfs

18.4 Serwer w sieci NT

Rozdział 19

Udostępnianie internetu

19.1 Maskarada i przekazywanie pakietów (Lech Ocaya Gamon)

Celem tego rozdziału będzie takie skonfigurowanie komputera spełniającego rolę bramy dostępowej, aby wszystkie komputery z sieci lokalnej przedstawionej na rysunku 1.1 (znajdziesz go w pierwszym rozdziale niniejszego podręcznika) miały dostęp do sieci internet. Zadanie wydaje się być stosunkowo proste i podstawową wersję podziału łącza można wykonać w dosłownie parę minut — wystarczy stworzyć kilka regułek dla programu iptables i gotowe. Nie jest to jednak podejście bezpieczne i rozpatrywanie podziału łącza w oderwaniu od firewall'a może się okazać brzemienne w skutkach dla naszej sieci LAN. Zalecam zatem zapoznanie się z rozdziałem "Zapora ogniowa" przed przystąpieniem do działań opisanych w tym rozdziale.

Ale zacznijmy od początku i odpowiedzmy sobie na pytanie : co to jest udostępnianie łącza, dlaczego się je robi? Jak wiemy informacja przekazywana w sieciach komputerowych jest "kawałkowana", czyli dane są rozbijane na tzw. pakiety. Aby wymiana informacji mogła w ogóle nastąpić konieczne jest odpowiednie zaadresowanie tychże pakietów — muszą zawierać adres IP i numer portu komputera wysyłającego pakiet, jak również adres IP i numer portu komputera, dla którego dany pakiet jest przeznaczony. Oczywiście aby pakiety trafiały do właściwych rąk, konieczne jest aby adresy IP były niepowtarzalne. Warunek ten jest spełniony dla komputerów pracujących w tej samej sieci — czy to będzie sieć internet (komputery z tzw. publicznymi adresami IP), czy sieć lokalna naszego dostawcy usług internetowych, czy też nasz firmowy/domowy LAN. Kłopoty zaczynają się kiedy próbujemy sprawić, aby komunikowały się między sobą komputery należące do różnych, odrębnych sieci — adres IP komputera-nadawcy nie istnieje w sieci komputera-odbiorcy, co w praktyce oznacza, że nawet jeśli komputer-odbiorca otrzyma pakiet, to zwrotne dane prześle w próżnię. Aby uniknąć takiej sytuacji na styku dwóch (lub więcej) sieci stawia się komputer spełniający rolę bramy (gateway) dostępowej dla komputerów z sieci lokalnej i routera pomiędzy sieciami — posiada on przynajmniej dwa interfejsy sieciowe, każdy skonfigurowany tak, aby pracować w innej sieci i niejako reprezentuje komputer-nadawcę w sieci komputera-odbiorcy, czyli w przesyłanym pakiecie adres IP nadawcy jest zamieniany na adres IP zewnętrznego interfejsu routera. Zwrotnie pakiety trafiają na zewnętrzny interfejs routera, gdzie następuje zamiana adresów IP w drugą stronę tzn. zewnętrzny adres IP bramy jest przerabiany na adres IP komputera-nadawcy (który oczywiście w międzyczasie stał się de facto komputerem-odbiorcą). Proste i efektywne, nieprawdaż? W uproszczeniu można zatem powiedzieć, że komputer-router spełnia dwie funkcje :

- tłumaczenie adresów sieciowych lub z angielska Network Address Translation lub krót-

ko NAT

- przekazywanie "cudzych" pakietów pomiędzy sieciami

Jak wspominałem wcześniej podział łącza należy łączyć z tematem ściany ogniowej w naszej sieci i (zakładając, że firewall i router są skonfigurowane na jednym komputerze) do powyższych funkcji należy dodać jeszcze jedną : filtrowanie pakietów. Nie chcemy przecież, aby do naszej sieci lokalnej trafiało co popadnie! Jeśli w naszej sieci nie ma jakiegoś serwera udostępniającego usługi dla "świata zewnętrznego", z reguły oznacza to zezwolenie na nieograniczony ruch pakietów "z wewnątrz" oraz wpuszczanie "z zewnątrz" tylko tych pakietów, które dotyczą komunikacji zainicjowanej "z wewnątrz". No dobrze, poznaliśmy już teorię (w zarysie) i teraz czas na część praktyczną. Aurox domyślnie jest przygotowany do dzielenia łącza, a zatem do dzieła! Na początek należy poinformować system, że ma przekazywać pakiety czyli spełniać rolę routera. W tym celu należy wydać polecenie :

```
[root@dreptak root]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Teraz z kolei zadbamy o ustawienie zgrubnego filtrowania pakietów. W tym celu należy wydać następujące polecenia :

```
[root@dreptak root]# iptables -A FORWARD -i {interfejs_zewn} \
-o {interfejs_wewn} -m state --state ESTABLISHED,RELATED -j ACCEPT
[root@dreptak root]# iptables -A FORWARD -i {interfejs_zewn} \
-s {adres_IP_sieci_LAN} -j DROP
[root@dreptak root]# iptables -A FORWARD -i {interfejs_wewn} \
-o {interfejs_zewn} -j ACCEPT
[root@dreptak root]# iptables -A FORWARD -j DROP
```

Pierwsza linijka informuje program iptables, że ma akceptować (-j ACCEPT) tylko te pakiety przychodzące na interfejs zewnętrzny routera (-i interfejs_zewn) i przeznaczone dla sieci LAN (-o interfejs_wewn), które dotyczą komunikacji zainicjowanej przez sieć LAN (-m state –state ESTABLISHED,RELATED). Druga linijka informuje, że pakiety przychodzące na zewnętrzny interfejs routera i posiadające adres źródłowy należący do zakresu adresów IP sieci LAN (-s adres_IP_sieci_LAN) powinny zostać odrzucone (-j DROP). Ma to za zadanie uniemożliwić hostom spoza naszej sieci wewnętrznej podszywanie się pod hosty należące do tej sieci. Trzecia linijka informuje, że pakiety przychodzące na wewnętrzny interfejs routera i skierowane na jego interfejs zewnętrzny (czyli skierowane na zewnątrz) mają zostać zaakceptowane. Pozwalamy zatem na nieograniczone przekazywanie pakietów na zewnątrz. Czwarta linijka informuje, że domyślnie pakiety przeznaczone do przekazania mają być odrzucane. Jeśli więc pakiet nie spełni reguł zdefiniowanych w linijkach 1-3, to zostanie wobec niego zastosowana ogólna zasada zdefiniowana w linijce czwartej. Oczywiście wykonywanie czwartej linijki ma sens tylko, jeśli nie została ustawiona wcześniej domyślna polityka odrzucająca dla łańcucha FORWARD (polecenie : iptables -P FORWARD DROP).

W tym miejscu warto powiedzieć kilka słów o umieszczaniu informacji o pakietach w logach systemowych. Aby "zrzucać" za pomocą programu iptables informacje do logów należy użyć celu LOG (polecenie : iptables regułka -j LOG). Dla celów dydaktycznych zmienimy regułki zgrubnego filtrowania pakietów przedstawione powyżej tak, aby przed odrzuceniem pakietu informacja o nim była umieszczana w logu.

```
[root@dreptak root]# iptables -N LOGUJ_ODRZUC
[root@dreptak root]# iptables -A LOGUJ_ODRZUC -j LOG --log-prefix \
"NAT Filter - odrzucam" --m limit
[root@dreptak root]# iptables -A LOGUJ_ODRZUC -j DROP
[root@dreptak root]# iptables -A FORWARD -i {interfejs_zewn} \
-o {interfejs_wewn} -m state --state  ESTABLISHED,RELATED -j ACCEPT
[root@dreptak root]# iptables -A FORWARD -i {interfejs_zewn} \
```

```
-s {adres_IP_sieci_LAN} -j LOGUJ_ODRZUC
[root@dreptak root]# iptables -A FORWARD -i {interfejs_wewn} \
-o {interfejs_zewn} -j ACCEPT
[root@dreptak root]# iptables -A FORWARD -j LOGUJ_ODRZUC
```

Pierwsze trzy linijki definiują łańcuch użytkownika `LOGUJ_ODRZUC`, który najpierw umieszcza informację o pakiecie w logu z prefiksem "NAT Filter - odrzucam", a następnie tenże pakiet odrzuca. Piąta linijka zamiast natychmiast odrzucać pakiety podszywające się pod adresy z naszej sieci LAN przekazuje je do łańcucha `LOGUJ_ODRZUC` (który jak wiemy najpierw je loguje, a dopiero później odrzuca). Szósta linijka robi to samo co piąta dla wszystkich pozostałych pakietów.

Teraz nadszedł czas na wydanie polecenia uruchamiającego tłumaczenie adresów. Zanim to jednak nastąpi trzeba powiedzieć kilka słów na temat rodzajów NAT. NAT można podzielić na dwie podstawowe grupy :

- NAT źródłowy lub inaczej SNAT (Source NAT) — w grupie tej podmianie ulega źródłowy adres IP, czyli adres komputera-nadawcy,
- NAT docelowy lub inaczej DNAT (Destination NAT) — w grupie tej podmianie ulega adres IP komputera-odbiorcy.

Oczywiście z punktu widzenia udostępniania łącza internetowego będzie nas interesował wyłącznie SNAT. I tu mamy dwie opcje do wyboru :

- SNAT podstawowy — adres źródłowy jest zamieniany na statycznie przypisany do interfejsu zewnętrznego adres IP
- Maskarada — adres źródłowy jest zamieniany na aktualnie przypisany do interfejsu zewnętrznego adres IP.

Jeśli zatem mamy na interfejsie zewnętrznym statycznie przypisany adres ip, to wydajemy polecenie uruchamiające SNAT podstawowy :

```
[root@dreptak root]# iptables -t nat -A POSTROUTING -o {interfejs_zewn} \
-d 0/0 -j SNAT --to-source {zewn_IP}
```

Linijka ta informuje program iptables, że przed przekazaniem pakietów skierowanych do dowolnej sieci (-d 0/0) na interfejs zewnętrzny ma wykonać NAT źródłowy do statycznie określonego zewnętrznego adresu IP. Jeśli na interfejsie zewnętrznym mamy dynamicznie przypisywany adres IP, to wydajemy komendę :

```
[root@dreptak root]# iptables -t nat -A POSTROUTING -o {interfejs_zewn} \
-d 0/0 -j MASQUERADE
```

Linijka ta informuje program iptables, że przed przekazaniem pakietów skierowanych do dowolnej sieci (-d 0/0) na interfejs zewnętrzny ma wykonać NAT źródłowy (Maskaradę) do dynamicznie określanego zewnętrznego adresu IP obecnie przypisanego do interfejsu zewnętrznego. I to by było na tyle. Jeśli działający w systemie firewall nie będzie kolidował z wymienionymi wyżej regułkami programu iptables (domyślny firewall Auroksa *lokkit* niestety koliduje), nasz komputer-brama powinien już być w stanie rutować pakiety z sieci lokalnej do sieci internet i vice versa. Chciałbym zaznaczyć, że wymieniony wyżej sposób nie jest "jedynym słusznym" i na przykład, jeśli nie odpowiada nam opieranie regułek iptables o nazwy interfejsów sieciowych, to oczywiście zamiast opcji -o oraz -i możemy bazować na opcjach -s (sieć/IP źródłowe) i -d (sieć/IP docelowe), co pozwoli np. na określnenie, które komputery w sieci mogą mieć dostęp do internetu. Chciałbym również przypomnieć, że wymienione wyżej regułki iptables "żyją" tylko

do restartu komputera, więc aby ułatwić sobie życie należy postarać się, żeby trafiły do plików startowych (np. /etc/rc.d/rc.local).

Skoro już wiemy jak skłonić naszego Linuksa do robienia SNAT-a i maskarady, nadszedł czas, aby odnieść się do naszej podręcznikowej sieci (patrz rys. 1.1) i włączyć udostępnianie internetu do pliku rc.firewall stworzonego w rozdziale "Zapora ogniowa". Zmodyfikowany pod kątem udostępniania internetu skrypt znajduje się na końcu tego podrozdziału. Dla wygody dodałem numery linijek - przy tworzeniu działającego skryptu należy je pominąć. Postępujemy dokładnie według procedury opisanej powyżej :

1. Dopisujemy komendę uruchamiającą przekazywanie pakietów w jądrze — patrz linijka 23 w skrypcie
2. Dopisujemy komendę ładującą moduł obsługi połączeń FTP — linia 27
3. Dopisujemy linie czyszczące reguły — linie 32-33
4. Pozwalamy na przekazywanie pakietów wracających — linijki 48-50
5. Odrzucamy pakiety podszywające się pod pakiety z naszej sieci LAN — linijki 52-53
6. Pozwalamy na przekazywanie ruchu wychodzącego — linijki 55-57
7. Nakazujemy maskowanie ruchu wychodzącego (SNAT do statycznie określonego adresu zewnętrznego rutera) — linijki 59-60

I to już koniec - życzę wszystkim powodzenia w udostępnianiu łącza.

```
1  #!/bin/bash
2
3  IPTABLE=/sbin/iptables
4
5  # interfejsy sieciowe
6  INET_IFACE=eth0
7  LAN_IFACE=eth1
8
9  # adresy na serwerze
10 INET_ADR=10.1.1.2/255.255.255.255
11 LAN_ADR=192.168.1.1/255.255.255.255
12 LAN=192.168.1.0/255.255.255.0
13 ADMIN_IP=192.168.1.2
14
15 TCP_LAN=20,21,80,443,25,110
16 UDP_LAN=53,123
17 TCP_INET=20,21,25,80,443
18
19 # konfiguracja jadra
20 echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
21 echo 1 > /proc/sys/net/ipv4/tcp_syncookies
22 echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
23 echo 1 > /proc/sys/net/ipv4/ip_forward
24
25 /sbin/modprobe ip_conntrack
26 /sbin/modprobe ip_conntrack_ftp
27 /sbin/modprobe ip_nat_ftp
28
29 # Wyczysc wszystkie reguly
30 $IPTABLE -F
31 $IPTABLE -X
32 $IPTABLE -t nat -F
33 $IPTABLE -t nat -X
```

```
34
35 # Ustaw policy
36 $IPTABLE -P INPUT DROP
37 $IPTABLE -P FORWARD DROP
38 # $IPTABLE -P OUTPUT DROP
39 $IPTABLE -P OUTPUT ACCEPT
40
41 # Przepusc wszystko na loopbacku
42 $IPTABLE -A INPUT -i lo -j ACCEPT
43
44 # Przepusc pakiety wracajace
45 $IPTABLE -A INPUT -i $INET_IFACE -j ACCEPT -m state --state ESTABLISHED,RELATED
46 $IPTABLE -A INPUT -i $LAN_IFACE -j ACCEPT -m state --state ESTABLISHED,RELATED
47
48 # Przekazuj pakiety wracajace
49 $IPTABLE -A FORWARD -i $INET_IFACE -o $LAN_IFACE -m state --state \
50 ESTABLISHED,RELATED -j ACCEPT
51
52 # Nie pozwalaj na przekazywanie sfalszowanych pakietow
53 $IPTABLE -A FORWARD -i $INET_IFACE -s $LAN -j DROP
54
55 # Pozwalaj na przekazywanie ruchu wychodzacego
56 $IPTABLE -A FORWARD -i $LAN_IFACE -o $INET_IFACE -p TCP -m multiport \
57     --destination-port $TCP_INET -j ACCEPT
58
59 # Maskuj ruch wychodzacy
60 $IPTABLE -t nat -A POSTROUTING -o $INET_IFACE -d 0/0 -j SNAT --to-source $INET_ADR
61
62 # Odpowiada/przyjmuje ping
63 $IPTABLE -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
64
65 # Pytanie o ident
66 $IPTABLE -A INPUT -p tcp --dport 113 -j REJECT --reject-with icmp-port-unreachable
67
68 # Wpuszczam ssh od admina
69 $IPTABLE -A INPUT -i $LAN_IFACE -p tcp -s $ADMIN_IP --dport ssh -j ACCEPT
70
71 # Dopuszczone z LAN'u
72 $IPTABLE -A INPUT -i $LAN_IFACE -p tcp -s $LAN -j ACCEPT -m multiport \
73   --destination-port $TCP_LAN
74 $IPTABLE -A INPUT -i $LAN_IFACE -p udp -s $LAN -j ACCEPT -m multiport \
75   --destination-port $UDP_LAN
76
77 #$IPTABLE -A INPUT -s $LAN -j LOG --log-prefix '[z lanu]'
78
79 # Dopuszczone z Inetu
80 $IPTABLE -A INPUT -i $INET_IFACE -p tcp -d $INET_ADR -j ACCEPT -m multiport \
81   --destination-port $TCP_INET
82
83 #$IPTABLE -A INPUT -s !$LAN -j LOG --log-prefix '[z inetu]'
```

19.2 Serwer proxy

19.3 Podział pasma

Rozdział 20

Baza danych

Jednym z istotniejszych elementów wykorzystania komputerów jest gromadzenie i udostępnianie danych. Do tego celu wykorzystywane są aplikacje nazywane ogólnie bazami danych. W naszej dystrybucji mamy dwie takie aplikacje: MySQL oraz PostgreSQL. Ze względu na dużą popularność oraz stosunkowo prostą obsługę, w podręczniku, zajmiemy się bazą MySQL.

20.1 Instalacja i konfiguracja serwera mysqld

Instalacja nie stwarza większego problemu. Najistotniejszy jest tutaj pakiet mysql-server. Zawiera on aplikacje i skrypty niezbędne do założenia systemu bazodanowego oraz uruchomienia i obsługi serwera. Ponieważ z całą pewnością będziemy chcieli w jakiś sposób dostać się do danych to potrzebujemy jakiejś aplikacji klienckiej. Tutaj z pomocą przychodzi nam pakiet mysql zawierający taka aplikację oraz biblioteki wykorzystywane przez inne aplikacje.

W trakcie instalacji pakietu mysql-server zostanie założony system bazy danych. Pliki tego systemu znajdują się w katalogu `/var/lib/mysql`. Pakiet został przygotowany w taki sposób, że bezpośrednio po instalacji możemy nasz system bazodanowy uruchomić bez potrzeby wykonywania czynności konfiguracyjnych. Takie są założenia ale dbając o bezpieczeństwo naszego systemu musimy jednak kilka czynności wykonać. Pierwszą czynnością jaką powinniśmy wykonać bezpośrednio po pierwszym uruchomieniu serwera mysql jest ustawienie hasła użytkownika *root*.

W tym momencie potrzebne jest małe wyjaśnienie. MySQL, podobnie jak większość baz danych , posiada własny system kont i uprawnień z nimi związanych. System ten jest całkowicie niezależny od kont w systemie operacyjnym. Tak więc konto root w mysqlu nie ma nic wspólnego z kontem root w systemie operacyjnym. Dzięki temu można rozdzielić zadania administracyjne pomiędzy kilku użytkowników. Jeden może zajmować się systemem operacyjnym i o bazie danych nic nie wiedzieć ale potrzebuje dostęp do konta root w systemie. Natomiast inny użytkownik w systemie pracuje jako zwykły użytkownik ale zna hasło root'a w mysqlu i dzięki temu może sprawować funkcje administratora baz danych.

Jak już wcześniej powiedziałem pliki bazy danych znajdują się w określonym miejscu systemu plików (istnieje możliwość zmiany ich położenia). Bezpośrednio po instalacji mamy tam dwie bazy danych: mysql i test. Baza test nas w tym momencie nie interesuje gdyż jest to pusta baza danych przeznaczona do nauki posługiwania się mysqlem. Natomiast baza mysql jest dla nas szalenie istotna gdyż jest to baza wykorzystywana przez sam silnik systemu. W bazie tej zapisane są m.in. uprawnienia dostępu do systemu jako takiego oraz do poszczególnych baz danych. Na tym etapie naszej edukacji nie jest jednak istotne fizyczne rozmieszczenie plików

gdyż są to pliki binarne i nie sięgamy do nich bezpośrednio ale zawsze za pośrednictwem silnika bazy danych.

Wróćmy zatem do hasła użytkownika root. Do zmiany tego hasła wykorzystamy polecenie `mysqladmin`. Na potrzeby niniejszego kursu przyjmuję, że administratorem bazy danych został użytkownik *tuptus* a demon `mysqld` został uruchomiony i działa poprawnie. Przystępujemy zatem do zmiany hasła:

```
[tuptus@dreptak tuptus]$ mysqladmin -u root password 12345678
[tuptus@dreptak tuptus]$
```

Jak widzisz składnia tego polecenia jest prosta. Wpisujemy samo polecenie, opcję `-u`, po której należy podać nazwę konta przez które chcemy się odwoływać do bazy i dalej polecenie do wykonania. W tym przypadku jest to polecenie `password` po którym należy podać nowe hasło użytkownika. Oczywiście podane w powyższym przykładzie hasło to bardzo złe hasło i powinieneś zastosować znacznie trudniejsze do odgadnięcia.

Uwaga. Niestety hasło podajemy otwartym tekstem w linii poleceń a tym samym zostaje ono zapisane do pliku `.bash_history`. Bardziej wrażliwi na bezpieczeństwo użytkownicy powinni po tej operacji "przelogować" się a następnie skasować odpowiednią linię z pliku historii.

Od tego momentu dostęp do bazy przez użytkownika root będzie zawsze wymagał podania hasła. Dlatego też przy każdym poleceniu będziesz musiał podawać opcję `-p` informującą mysqla, że chcemy podać hasło. Przykładowo, zmiana hasła wyglądać będzie następująco:

```
[tuptus@dreptak tuptus]$ mysqladmin -u root -p password 1qaz2wsx
Enter password:
[tuptus@dreptak tuptus]$
```

Hasło roota mamy ustawione ale to jeszcze nie koniec zabezpieczeń, które powinniśmy poczynić. Okazuje się, że do naszego systemu bazodanowego mogą się logować anonimowi użytkownicy bez podawania hasła. Wystarczy wpisać polecenie `mysql` i już jesteśmy w systemie. Wprawdzie taki użytkownik nie ma dostępu do bazy *mysql* ale i tak sytuacja taka stwarza zagrożenie. Sprawdźmy zatem jak to wygląda od środka. W tym celu musimy wejść do mysqla jako root, do bazy *mysql* i wyświetlić dane z tabeli *user*:

```
[tuptus@dreptak tuptus]$ mysql -u root -p mysql
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.0.18-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select host,user,password from user;
+--------------------+------+------------------+
| host               | user | password         |
+--------------------+------+------------------+
| localhost          | root | 360af43f5cd1aa99 |
| dreptak.tupward.pl | root |                  |
| localhost          |      |                  |
| dreptak.tupward.pl |      |                  |
+--------------------+------+------------------+
4 rows in set (0.00 sec)
```

Jak widzisz, ustawione jest tylko hasło użytkownika root logującego się z hosta *localhost*. Pozostałe pozycje to właśnie to zagrożenie, którego powinniśmy się pozbyć. Zatem kasujemy te rekordy:

```
mysql> delete from user where host='dreptak.tupward.pl';
Query OK, 2 rows affected (0.03 sec)

mysql> delete from user where host='localhost' and user='';
Query OK, 1 row affected (0.00 sec)

mysql> select host,user,password from user;
+-----------+------+-------------------+
| host      | user | password          |
+-----------+------+-------------------+
| localhost | root | 360af43f5cd1aa99 |
+-----------+------+-------------------+
1 row in set (0.00 sec)

mysql> \q
Bye
[tuptus@dreptak tuptus]$
```

Czyli osiągnęliśmy to o co nam chodziło, do systemu może się dostać wyłącznie root i to tylko z hosta, na którym działa baza danych.

Zanim przejdziemy do dalszych czynności kilka słów wyjaśnienia do tego co już zrobiliśmy. Zacznijmy od samego wywołania klienta mysql. Znaczenia opcji `-u root -p` zapewne już sam się domyśliłeś więc nie będę się nad tym rozwodził. Zastanawiające może być natomiast końcowe `mysql`. Jest to nazwa bazy danych, do której chcemy się podłączyć bezpośrednio po zalogowaniu. Jest to parametr opcjonalny, nie musimy go podawać ale wówczas musimy określić bazę już w samym kliencie wykorzystując polecenie `use <baza>;`.

Skoro już jesteśmy w środowisku bazy danych to kolej na polecenia SQL. Pierwsze polecenie jakie wykonaliśmy to polecenie `select`. Jest to tak zwane polecenie wybierające. Podstawowa składnia wygląda następująco:

```
SELECT <kolumna>, <kolumna> ...
  FROM <tabela>
  WHERE <warunek wybierania>;
```

Słowa kluczowe podałem wielkimi literami ale dla mysqla nie ma to żadnego znaczenia czy polecenia pisane są wielkimi czy małymi literami. Jak widać składnia jest bardzo zbliżona do języka naturalnego (oczywiście angielskiego) i nie powinna stwarzać większych problemów. Skąd jednak brać nazwy kolumn oraz nazwy tabel? Możemy do tego celu wykorzystać polecenia `show` znajdujące się w mysqlu:

```
mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)

mysql> show tables;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
```

```
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
6 rows in set (0.01 sec)

mysql> show columns from user;
+-------------------+-------------------------+-------+-----+---------+-------+
| Field             | Type                    | Null | Key | Default | Extra |
+-------------------+-------------------------+------+-----+---------+-------+
| Host              | varchar(60) binary      |      | PRI |         |       |
| User              | varchar(16) binary      |      | PRI |         |       |
| Password          | varchar(16) binary      |      |     |         |       |
| Select_priv       | enum('N','Y')           |      |     | N       |       |
(...)
31 rows in set (0.03 sec)

mysql>
```

Kolejnym poleceniem, które wykorzystaliśmy było polecenie `delete`. Należy ono to poleceń modyfikujących dane. Jest to o tyle istotne, że wywołanie polecenia modyfikującego zostaje odnotowane w logach mysqla (ale tylko w sytuacji gdy faktycznie spowodowało modyfikacje danych). Samo polecenie `delete` to nic odkrywczego ale pojawił się tu nowy element - warunki. Jak widzisz z powyższych przykładów ustawianie warunków nie jest zadaniem wyjątkowo trudnym. Oczywiście w naszym przykładzie warunki są bardzo proste ale w większości przypadków takie rozwiązanie w zupełności nam wystarczy. Warunki wybierania rekordów do wyświetlenia czy modyfikacji zawsze zaczyna się od słowa kluczowego `where`, po którym następuje warunek. Piszę w liczbie pojedynczej gdyż zakwalifikowanie danego rekordu do listy przetwarzanych rekordów zależy od spełnienia wszystkiego co następuje po `where`. W pierwszym poleceniu `delete` mamy prosty warunek określający, że obróbce podlegać będą rekordy, w których pole *host* ma określoną wartość. Drugie polecenie jest nieco bardziej złożone, spełnione muszą być dwa warunki równocześnie. Za tą równoczesność odpowiada słowo kluczowe `and`. Aby rekord "zakwalifikował się" pole *host* musi mieć wartość *localhost* i równocześnie pole *user* musi zawierać pusty ciąg znaków.

Podane polecenia to podstawy języka SQL. W trakcie omawiania obsługi mysqla wykorzystywać będziemy jeszcze polecenia:

insert polecenie dodające nowe rekordy do tabeli

update polecenie modyfikujące zawartość pól tabeli

grant i revoke polecenia modyfikujące uprawnienia

Polecam lekturę manuala mysqla lub kursu języka SQL gdyż tutaj podaję jedynie zupełnie podstawowe informacje o stosowanych polecenia a mają one znacznie więcej możliwości.

No dobrze. Zmodyfikowaliśmy "listę dostępu" ale to jeszcze nie koniec. Wprawdzie mysql wpuści obecnie tylko użytkownika root i to tylko w przypadku logowania z *localhost* ale nadal mogą być podejmowane próby logowania z innych hostów a to może być bardzo uciążliwe dla naszego serwera. Przyglądnijmy się temu nieco dokładniej. Z serwerem mysql możemy się łączyć dwoma drogami:

- protokół TCP/IP łącząc się z portem 3306
- wykorzystując mechanizm socketów

Pierwszy sposób nie wymaga specjalnego wytłumaczenia gdyż działa analogicznie do innych serwerów. Drugi sposób jest nieco odmienny i jest wykorzystywany przez wiele aplikacji, które nie potrzebują dostępu do innych hostów. W tym przypadku zamiast adresu hosta i numeru portu mamy do dyspozycji plik specjalny nazywany socketem. Socket to po angielsku kieszeń a to już częściowo daje nam wyobrażenie o działaniu tego mechanizmu. Do tego pliku jedna aplikacja zapisuje dane, które odbiera inna aplikacja i po przetworzeniu zwraca do socketu wyniki, które odbiera aplikacja "pytająca". Przejdź zatem do praktyki. Zacznijmy od sprawdzenia jak działa obecnie nasz demon mysqld:

```
[root@dreptak root]# netstat -a | grep mysql
tcp    0      0 *:mysql                 *:*                         LISTEN
unix  2 [ ACC ] STREAM  LISTENING     102098 /var/lib/mysql/mysql.sock
```

Zgodnie z oczekiwaniem otrzymaliśmy dwie linie, pierwsza informuje nas, że demon nasłuchuje na porcie 3306/tcp a druga pokazuje nasłuchujący socket.

Powinniśmy się teraz zastanowić czy rzeczywiście potrzebujemy możliwości zdalnego dostępu do bazy danych. W większości przypadków taka funkcjonalność nie będzie nam potrzebna ale część aplikacji, nawet pracując na tym samym hoście co mysql, łączy się z nim wykorzystując TCP/IP. Innym przypadkiem wykorzystania zdalnego dostępu do bazy jest rozdzielenie funkcji serwisu www na oddzielne hosty (na jednej maszynie serwer www a baza na osobnej) oraz administrowanie serwerem mysql ze zdalnej maszyny. Jeśli jednak chcemy pozostawić uruchomiony dostęp z wykorzystaniem TCP/IP to powinniśmy utworzyć odpowiednie reguły filtrowania na ogniomurku. Osobiście preferuję wyłączenie portu 3306 ale tutaj zajmiemy się oboma przypadkami.

Zaczniemy od "uszczelnienia" naszego hosta pozostawiając działający port 3306. Pierwszą sprawą jest ogniomurek. Jeśli wykorzystałeś opisany w tym podręczniku przykład to do portu mysqla nie ma dostępu z zewnątrz i to nam wystarczy jeśli jedyną przyczyną potrzeby funkcjonowania łącza TCP/IP jest jakaś aplikacja działająca na serwerze. W takim przypadku w konfiguracji tej aplikacji podajemy jako host *localhost* i port 3306. Jeśli jednak chcemy się do mysqla dostawać zdalnie to musimy dodać odpowiednią regułkę do *iptables*

```
$IPTABLE -A INPUT -i $LAN_IFACE -p tcp -s $ADMIN_IP --dport 3306 -j ACCEPT
```

Zastanawiasz się dlaczego właśnie taką? Zerknij zatem jeszcze raz na skrypt `rc.firewall` i schemat naszej sieci. Dzięki takiej regułce będzie można się łączyć z mysql z maszyny administratora. Ale to jeszcze nie koniec, przecież mysql nas nie wpuści. Musimy zatem dodać odpowiedni wpis do "listy dostępu":

```
[tuptus@dreptak tuptus]$ mysql -u root -p mysql
(...)
mysql> GRANT ALL ON *.* TO root@admin.tupward.pl
    -> IDENTIFIED BY '1qaz2wsx' WITH GRANT OPTION;
Query OK, 0 rows affected (0.05 sec)

mysql> select host,user,password from user;
+------------------+------+------------------+
| host             | user | password         |
+------------------+------+------------------+
| localhost        | root | 360af43f5cd1aa99 |
| admin.tupward.pl | root | 360af43f5cd1aa99 |
+------------------+------+------------------+
2 rows in set (0.00 sec)
```

Ponieważ jako admin bazy danych będziesz często korzystał z tego polecenia to kilka słów wyjaśnienia. Polecenie `GRANT` służy do nadawania uprawnień. Jako pierwszy parametr podajemy listę uprawnień. W naszym przypadku wykorzystaliśmy słowo `ALL` rozumiane przez mysql jako

wszystkie uprawnienia (można w tym miejscu podawać pełną wersję czyli `ALL PRIVILEGES`). Dalej określamy czego dotyczą nadawane uprawnienia podając tą informację w formacie ¡baza danych¿.¡tabela¿. Ponieważ my ustawiamy uprawnienia dla admina to dajemy mu prawa do wszystkich obiektów systemu. Teraz kolej na informację komu nadajemy te uprawnienia. Podajemy tutaj nazwę konta a po @ adres hosta z jakiego będzie dokonywane logowanie. W tym miejscu, podobnie jak wcześniej, możemy stosować znaki wieloznaczne ale to zawsze pewne ryzyko. Kolejny element tego polecenia to hasło. W tym miejscu podajemy hasło otwartym tekstem jednak w trakcie wykonania polecenia mysql wykonuje funkcje `PASSWORD` i do bazy zapisywane jest hasło w formie zakodowanej. Dla zwykłego konta to byłoby wszystko jednak dla admina należy jeszcze dodać `WITH GRANT OPTION`. Mysql sam z siebie nie daje nikomu prawa do zmiany uprawnień nawet jeśli jako listę uprawnień podamy `ALL`. Aby dane konto miało prawo zmieniać uprawnienia musimy dodać opcję jak w przykładzie powyżej.

Teraz wypadałoby jeszcze sprawdzić czy to postępowanie przyniosło oczekiwane efekty. Logujemy się zatem na hoście *adnim.tupward.pl* i wykonujemy polecenie:

```
[tuptus@admn tuptus]$ mysql -h dreptak.tupward.pl -u root -p mysql
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.0.18-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> \q
[tuptus@admn tuptus]$
```

Jak widzisz pojawił się dodatkowy parametr w wywołaniu mysqla. Tak naprawdę to on zawsze tam był tylko, że przez domniemanie jest to `-h localhost` więc do tej pory nie musieliśmy go podawać.

A teraz postąpmy bardziej restrykcyjnie. Wyłączymy obsługę TCP/IP. Tutaj sprawa jest znacznie prostsza niż w poprzednim przypadku, wystarczy w pliku konfiguracyjnym lub w linii poleceń startującej demona mysqld dodać parametr `skip-networking` (w linii poleceń to będzie `--skip-networking`). No właśnie, wystarczy dodać ale gdzie? Parametry konfiguracyjne, które dotyczą mysqla jako całości umieszczamy w pliku `/etc/my.cnf`. Pliku tego nie ma w pakiecie więc najpierw musimy go utworzyć np. poleceniem `touch`. Kiedy już mamy ten plik to wpisujemy do niego następujące linie:

```
[mysqld]
skip-networking
```

Teraz już tylko restart serwisu mysql i do serwera dostęp możliwy jest tylko lokalnie przez socket. Oczywiście warto to sprawdzić pleceniem `netstat` oraz sprawdzić czy działają nasze aplikacje.

20.2 Podstawy administracji serwerem mysql

Skoro mamy już działający serwer możemy się zająć codziennymi obowiązkami administratora. Każdy projekt związany z bazą danych zaczyna się od utworzenia bazy danych oraz nadania odpowiednich uprawnień do tej bazy. Na potrzeby naszego kursu przyjmijmy, że naszym zadaniem jest uruchomienia aplikacji www o nazwie *BIP* opartej o bazę danych. Na razie nie wnikamy w szczegóły instalatora tej aplikacji ani w wewnętrzną strukturę bazy danych. Zadaniem administratora jest przygotowanie środowiska niezbędnego do uruchomienia takiego instalatora.

Zaczynamy zatem od założenia bazy o nazwie *bip* oraz użytkownika *bip_admin* posiadającego pełne prawa w tejże bazie (przypominam, że pełne prawa nie obejmują modyfikacji praw)

```
[tuptus@dreptak tuptus]$ mysqladmin -u root -p create bip
Enter password:
[tuptus@dreptak tuptus]$ mysql -u root -p mysql
Enter password:
(...)
mysql> grant all on bip.* to bip_admin@localhost identified by '1qaz2wsx';
Query OK, 0 rows affected (0.02 sec)

mysql> show databases;
+----------+
| Database |
+----------+
| bip      |
| mysql    |
| test     |
+----------+
3 rows in set (0.00 sec)
```

Tak więc widzimy, że powstała baza danych bip. Istnienie konta *bip_admin* możemy sprawdzić w `mysql.user` Ale jak sprawdzić nadane uprawnienia? Możemy do tego celu wykorzystać polecenie `mysqlaccess`:

```
[tuptus@dreptak tuptus]$ mysqlaccess localhost bip_admin bip -U root -P
mysqlaccess Version 2.06, 20 Dec 2000
By RUG-AIV, by Yves Carlier (Yves.Carlier@rug.ac.be)
Changes by Steve Harvey (sgh@vex.net)
This software comes with ABSOLUTELY NO WARRANTY.
Password for MySQL superuser root:

Access-rights
for USER 'bip_admin', from HOST 'localhost', to DB 'bip'
        +-----------------+---+ +-----------------+---+
        | Select_priv     | Y | | Show_db_priv    | N |
        | Insert_priv     | Y | | Super_priv      | N |
        | Update_priv     | Y | | Create_tmp_table_priv | Y |
        | Delete_priv     | Y | | Lock_tables_priv | Y |
        | Create_priv     | Y | | Execute_priv    | N |
        | Drop_priv       | Y | | Repl_slave_priv | N |
        | Reload_priv     | N | | Repl_client_priv | N |
        | Shutdown_priv   | N | | Ssl_type        | ? |
        | Process_priv    | N | | Ssl_cipher      | ? |
        | File_priv       | N | | X509_issuer     | ? |
        | Grant_priv      | N | | X509_subject    | ? |
        | References_priv | Y | | Max_questions   | 0 |
        | Index_priv      | Y | | Max_updates     | 0 |
        | Alter_priv      | Y | | Max_connections | 0 |
        +-----------------+---+ +-----------------+---+
NOTE:    A password is required for user 'bip_admin' :-(

The following rules are used:
 db    : 'localhost','bip','bip_admin','Y','Y','Y','Y','Y','Y','N','Y','Y',
'Y','Y','Y'
 host  : 'Not processed: host-field is not empty in db-table.'
 user  : 'localhost','bip_admin','360af43f5cd1aa99','N','N','N','N','N','N',
'N','N','N','N','N','N','N','N','N','N','N','N','N','N','','','','',
'0','0','0'
--------------------------------------------------------------------------------

Access-rights
for USER 'ANY_NEW_USER', from HOST 'localhost', to DB 'bip'
```

```
        +-----------------+---+ +-----------------+---+
        | Select_priv     | N | | Show_db_priv    | N |
        | Insert_priv     | N | | Super_priv      | N |
        | Update_priv     | N | | Create_tmp_table_priv | N |
        | Delete_priv     | N | | Lock_tables_priv | N |
        | Create_priv     | N | | Execute_priv    | N |
        | Drop_priv       | N | | Repl_slave_priv | N |
        | Reload_priv     | N | | Repl_client_priv | N |
        | Shutdown_priv   | N | | Ssl_type        | N |
        | Process_priv    | N | | Ssl_cipher      | N |
        | File_priv       | N | | X509_issuer     | N |
        | Grant_priv      | N | | X509_subject    | N |
        | References_priv | N | | Max_questions   | N |
        | Index_priv      | N | | Max_updates     | N |
        | Alter_priv      | N | | Max_connections | N |
        +-----------------+---+ +-----------------+---+
BEWARE:  Accessing the db as an anonymous user.
      :  Your username has no relevance

The following rules are used:
 db    : 'No matching rule'
 host  : 'Not processed: host-field is not empty in db-table.'
 user  : 'No matching rule'

[tuptus@dreptak tuptus]$
```

Z powyższego wynika, że użytkownik *bip_admin* ma pełne prawa do użytkowania bazy *bip* (poza prawami typowo administracyjnymi) natomiast inni użytkownicy nie mają żadnych praw. I o to nam chodziło. Teraz możemy przekazać adminowi aplikacji informacje o nazwie bazy, konta oraz haśle i teraz on może przeprowadzić instalację i konfigurację aplikacji. Tutaj właściwie drogi obu adminów się rozchodzą ale nie do końca. Przecież jednym z obowiązków admina jest dbałość o bezpieczeństwo danych a to zobowiązuje nas do wykonywania kopii bezpieczeństwa bazy danych. Wprawdzie administrator systemu zapewne wykonuje kopie systemu plików ale w przypadku bazy danych to raczej marne zabezpieczenie gdyż w trakcie wykonywania backupu mogły nastąpić jakieś zmiany w bazie i dane na kopii będą niespójne. Jako administrator bazy danych masz do dyspozycji lepsze narzędzia. Zacznijmy od polecenia `mysqldump`. Polecenie to umożliwia nam dokonacie "zrzutu" bazy danych do pliku tekstowego, który następnie może być przeniesiony na zewnętrzny nośnik a w razie potrzeby odtworzony. Ale przejdźmy do zadania, mamy wykonać kopię bazy danych w taki sposób aby w razie awarii można było tą bazę odzyskać. Najprostsze rozwiązanie to wpisanie następującej komendy:

```
[tuptus@dreptak tuptus]$ mysqldump -u root -p bip > bip.sql
Enter password:
[tuptus@dreptak tuptus]$
```

Dzięki temu nasza baza *bip* wylądowała w katalogu bieżącym w pliku `bip.sql` Jeśli teraz zaszłaby potrzeba przywrócenia bazy danych (po jej "zniknięciu") to wykonujemy polecenie:

```
[tuptus@dreptak tuptus]$ mysql -u root -p bip > bip.sql
Enter password:
[tuptus@dreptak tuptus]$
```

Oczywiście takie polecenie zadziała pod warunkiem, że w systemie istnieje **pusta** baza *bip*.

Przedstawione wyżej podejście ma jednak kilka wad. Po pierwsze dla dużej bazy danych taki zrzut musi chwilkę potrwać a w międzyczasie dane mogą ulec zmianie i w efekcie uzyskamy niespójną kopię. Drugi problem to sprawa odtworzenia częściowo zniszczonej bazy. Jeśli w bazie *bip* istnieje jakaś tabela o nazwie identycznej z tą która znajduje się w kopii to otrzymamy komunikat błędu i jeszcze gorszy bałagan niż był do tej pory gdyż część kopii już się odtworzyła a część,

która następuje po tej feralnej tabeli nie. Kolejny problem to znowu wielkość bazy. Standardowym działaniem mysql jest wykonanie najpierw zrzutu do pamięci a dopiero potem do pliku. Postępowanie jak najbardziej słuszne gdyż znacznie przyśpiesza proces odczytu danych z tabel zwiększając szansę na uzyskanie spójnej kopii. Jednak pamięć RAM może tego nie wytrzymać przy większych bazach danych. Na koniec jeszcze jedna uwaga. Czasami nazwy kolumn tabeli posiadają nazwy, które mysql rozpoznaje jako słowa zastrzeżone i mimo, że zrzutu dokona to później nie chce z takiej kopii odtworzyć bazy.

Wszystkim tym problemom możemy zaradzić dodając odpowiednie parametry do polecenia `mysqldump`

```
[tuptus@dreptak tuptus]$ mysqldump -F --lock-tables --add-drop-table -q -e -Q \
> -u root -p --databases bip > bip.sql
Enter password:
[tuptus@dreptak tuptus]$
```

Zamiast opcji `--lock-tables --add-drop-table -q -e` możemy dać jedną, która zawiera w sobie te opcje — `--opt`. Teraz wygląda to znacznie lepiej. Wprawdzie przy wykonywaniu kopii kilku baz danych tą metodą nie mamy zapewnionej spójności danych pomiędzy bazami danych ale to już jest zadanie wykraczające poza nasze potrzeby.

Wiemy już jak wykonać kopię bazy ale przecież nie będziemy tego robić ręcznie. Raz, że możemy zapomnieć a dwa, że przecież nie co dzień jesteśmy obecni przy konsoli, są przecież wakacje, święta itd. Musimy zatem zadbać o automatyzację tego zadania. Ponieważ kopia wykonywana podanym wyżej sposobem zawiera całą bazę danych to właściwie moglibyśmy wykonywać ją zawsze do tego samego pliku. Ja jednak jestem ostrożny i wolę zawsze mieć jeszcze coś w zapasie. Dlatego proponuję przechowywanie kilku ostatnich kopii np. czterech. Kopie te przechowywać będziemy w katalogu `/var/db/dumps`. Katalog `/var/db` z reguły jest pusty więc można z adminem systemu uzgodnić, taką lokalizację. Całość zadania zlecimy demonowi crond, o którym już wcześniej pisałem. Przystępujemy zatem do realizacji. Składnie polecenia wykonującego kopię już mamy ale teraz jeszcze trzeba opracować "otoczenie". Przegotujmy zatem prosty skrypt:

```
#!/bin/sh

DUMP_DIR=/var/db/dump
PLIK=bip.sql
PASSWORD='1qaz2wsx'

cd $DUMP_DIR

# Rotacja pliku kopii
for numer in 3 2 1
do
    numer_1=$(($numer + 1))
    if [ -f $PLIK.$numer ]
    then
        mv $PLIK.$numer $PLIK.$numer_1
    fi
done

if [ -f $PLIK ]
then
    mv $PLIK $PLIK.1
fi

# Wlasciwa kopia
/usr/bin/mysqldump -F --opt -Q -u root -p$PASSWORD --databases bip > $PLIK
```

Plik ten zapisujemy np. jako dbbackup.sh i nadajemy uprawnienia wykonania. Zwróć uwagę, że w plik zapisane jest hasło na konto *mysql/root*. Warto zatem zmienić prawa dostępu do tego pliku tak aby tylko administrator bazy danych miał do niego dostęp.

```
[tuptus@dreptak tuptus]$ chmod 700 dbbackup.sh
[tuptus@dreptak tuptus]$
```

Skoro już mamy narzędzie to teraz pozostaje już tylko zatrudnienie crona.

```
[tuptus@dreptak tuptus]$ crontab -e
00 01 * * * /home/tuptus/dbbackup.sh 2>&1
~
[tuptus@dreptak tuptus]$
```

Oczywiście to tylko przykład. Przy takim zapisie backup będzie się wykonywał codziennie o pierwszej w nocy. Godzinę powinieneś jednak ustalić na podstawie obserwacji czasu wykonywania backupu oraz uzgodnienia z administratorem systemu tak, żeby powstałe pliki trafiły na kopię filesystemów.

Na koniec wrócimy jeszcze raz do problematyki haseł dostępu do bazy danych. Do tej pory hasło ustawiał administrator. Ale czy użytkownik może zmienić swoje hasło? Oczywiście może:

```
[tuptus@dreptak tuptus]$ mysqladmin -u bip_admin -p password 12345678
[tuptus@dreptak tuptus]$
```

Jak widzisz bardzo proste ale zdarza się, że przychodzi użytkownik do admina: *"Zapomniałem hasło"* I co wtedy? Wtedy administrator ma nieco pracy. Wiemy już, że hasła przechowywane są w bazie *mysql* w tabeli *user*. Hasło jednak przechowywane jest w formie zakodowanej i nie ma możliwości odtworzenia go. Jedynym wyjściem jest ustawienie nowego hasła. Nie można tego zrobić poleceniem `mysqladmin` gdyż to pozwala na zmianę tylko swojego hasła. Polecenia `GRANT` to też nie jest rozwiązanie bo przecież nie pamiętamy jakie uprawnienia ma aktualnie nieszczęśliwy użytkownik a odtwarzanie to mordęga. Możemy natomiast zmodyfikować wpisy w tabeli *user* wykorzystując polecenia SQLa modyfikujące bezpośrednio dane w tej tabeli.

```
mysql> update user set Password=PASSWORD('1qaz2wsx')
    -> where User='bip_admin';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>flush privileges;
Query OK, 0 rows affected (0.04 sec)

mysql>
```

Do powyższego potrzebne są dwa wyjaśnienia. W pierwszym poleceniu przyjąłem, że mamy tylko jeden wpis dla użytkownika *bip_admin*. Gdyby dla tego użytkownika było więcej wpisów i jeszcze dla każdego z nich ustawione jest inne hasło a zmienić trzeba tylko jedno to trzeba ustawić nieco ostrzej warunek polecenia `UPDATE`. Ilość wpisów oraz jakie to wpisy możemy sprawdzić poleceniem `SELECT`, które wykorzystywaliśmy wcześniej. Natomiast polecenie `UPDATE` przybierze wówczas postać:

```
mysql> update user set Password=PASSWORD('1qaz2wsx')
    -> where User='bip_admin'
    -> and Host='localhost';
```

co powoduje zmodyfikowanie hasła użytkownika *bip_admin* logującego się z hosta *localhost* natomiast nie zmieni wpisów dla innych rekordów.

Drugą sprawą jest polecenie `FLUSH`. Polecenie to powoduje przeładowanie systemu uprawnień naszej bazy danych. Jeśli tego polecenia nie wykonamy to system nie zauważy wprowadzonych przed chwilą zmian. Teraz możesz podać użytkownikowi nowe hasło.

Ale zdarza się, że z jakiegoś powodu nie mamy dostępu do bazy bo nie znamy hasła roota. Oczywiście, mam na myśli sytuacje np. przejęcia administracji bazą po kimś kto nie zostawił informacji o tym haśle a nie zapomnienie czy tym bardziej próbę włamania. Na szczęście możemy sobie z tym poradzić stosunkowo prosto pod warunkiem, że współpracujemy z adminem systemu operacyjnego. Postępowanie wygląda następująco:

- zatrzymać serwis mysql
- uruchomić ręcznie demona mysql bez obsługi systemu uprawnień
- wejść do mysqla i zmodyfikować wpisy dotyczące roota
- zatrzymać demona mysql
- uruchomić serwis mysql w standardowy sposób

Wygląda to dość prosto, w praktyce jest jednak kilka pułapek, o których najwyraźniej zapomniano w dokumentacji mysqla. Dlatego opiszę krok po kroku całe postępowanie. Zatem logujemy się na konsoli na konto root i zatrzymujemy serwis mysql. Następnie przechodzimy na konto mysql i uruchamiamy ręcznie demona

```
[tuptus@dreptak tuptus]$ su -
Password:
[root@dreptak root]# service mysql stop
Killing mysqld with pid 11551
Wait for mysqld to exit. done
[root@dreptak root]# su - mysql
-bash-2.05b$ /usr/sbin/mysqld --skip-grant-table &
[1] 11607
-bash-2.05b$ 041114 22:04:22  InnoDB: Started
/usr/sbin/mysqld: gotowe do poł?czenia

-bash-2.05b$
```

Tutaj jedna uwaga. Nie można uruchamiać demona mysql z konta root. To znaczy można ale wymaga to dodatkowych zabiegów i jest to niebezpieczne postępowanie więc stanowczo odradzam. Teraz należy zmienić hasło *mysql/root*. W dokumentacji podano polecenie `mysqladmin` jako rozwiązanie. Niestety to nie działa gdyż polecenie to nie ma dostępu do bazy z uprawnieniami. Dlatego należy wejść do mysqla i wykonać polecenie `UPDATE`

```
-bash-2.05b$ mysql -u root mysql
(...)
mysql> update user set password=''
    -> where user='root'
    -> and host='localhost';

mysql> \q
-bash-2.05b$
```

W tym momencie mamy wykasowane hasło roota. Teraz możemy zamknąć demona i uruchomić serwis

```
-bash-2.05b$ kill %1
-bash-2.05b$ 041114 22:15:14  /usr/sbin/mysqld: Standardowe zakończenie działania

041114 22:15:14  InnoDB: Starting shutdown...
```

```
041114 22:15:15  InnoDB: Shutdown completed
041114 22:15:15  /usr/sbin/mysqld: Zakończenie działania wykonane


[1]+  Done                    /usr/sbin/mysqld --skip-grant-table
-bash-2.05b$ logout
[root@dreptak root]# service mysql start
[root@dreptak root]# logout
[tuptus@dreptak tuptus]$
```

Teraz pozostaje już tylko ustawienie nowego hasła i już mamy bazę pod kontrolą.

```
[tuptus@dreptak tuptus]$ mysqladmin -u root password 1qaz2wsx
Enter password:
[tuptus@dreptak tuptus]$
```

To tyle o podstawach użytkowania bazy MySQL. Zachęcam do zapoznania się z bardzo obszerną dokumentacją, która znajduje się w katalogu `/usr/share/doc/mysql-server-4.0.18/` w pliku w formacie html. Możemy ją czytać wykorzystując przeglądarkę internetową wpisując jako adres `file:///usr/share/doc/mysql-server-4.0.18/manual.html`. W katalogu tym znajdują się również przykłady plików `my.cnf` — warto się z nimi zapoznać.

Część VII

Wizytówka pracowni

Rozdział 21

CMS - systemy zarządzania treścią

21.1 Pakiet PHPNuke

21.2 Pakiet tikiwiki

Część VIII

Dodatki

Dodatek A

Filmy w Auroksie

A.1 Instalacja

A.2 Konfiguracja

A.3 Eksploatacja

Dodatek B

Komputerowe biuro - OpenOffice

B.1 Instalacja

B.2 Konfiguracja

B.2.1 Współpraca z MySql

B.3 Składniki pakietu

B.3.1 Edytor tekstu

B.3.2 Arkusz kalkulacyjny

B.3.3 Prezentacje