Wydanie II

Najlepszy podręcznik dla systemu Ubuntu Serwer!

Ubuntu Serwer

Oficjalny podręcznik

- Jak zainstalować Ubuntu Serwer?
- Jak wykonywać typowe zadania administracyjne?
- Jak zwiększyć odporność systemu na awarie?

Benjamin Hill, Kyle Rankin

Helion

Ubuntu Serwer. Oficjalny podręcznik. Wydanie II

Autorzy: Kyle Rankin, Benjamin Hill
Tłumaczenie: Robert Górczyński
ISBN: 978-83-246-3300-5
Tytuł oryginału: Official Ubuntu Server Book, The (2nd Edition)
Format: 170×230, stron: 464

Najlepszy podręcznik dla systemu Ubuntu Serwer!

- Jak zainstalować Ubuntu Serwer?
- Jak wykonywać typowe zadania administracyjne?
- Jak zwiększyć odporność systemu na awarie?

Ubuntu to dystrybucja systemu operacyjnego Linux, która podbiła serca użytkowników domowych. Możliwość zapoznania się z systemem bez ingerencji w dotychczas używany system, banalna instalacja, wyjątkowo atrakcyjny interfejs użytkownika oraz małe wymagania to tylko niektóre atuty tego systemu. Autorzy postanowili pójść za ciosem i przygotowali wersję dla serwerów. Co sprawia, że jest wyjątkowa? Co ją odróżnia od innych dystrybucji? Czy warto ją zainstalować?

Na te pytania znajdziesz odpowiedź w jedynym oficjalnym podręczniku poświęconym dystrybucji Ubuntu w wersji serwerowej. Autorzy prezentują sposób instalacji systemu oraz podstawowe operacje administracyjne. Po szybkim i pełnym konkretnych informacji wstępie zajmiesz się zarządzaniem zainstalowanym oprogramowaniem, usługami dostarczanymi przez Ubuntu, zapewnieniem bezpieczeństwa w systemie oraz tworzeniem kopii zapasowej. Ponadto dowiesz się, jak monitorować pracę serwera, wirtualizować go oraz zwiększać odporność na awarie. Punkt po punkcie autorzy rozwieją wszystkie Twoje wątpliwości związane z systemem Ubuntu w tej wersji. Jeżeli nie jesteś pewien, czy to coś dla Ciebie, chcesz poznać ten system lub jesteś pasjonatem Linuksa – musisz mieć tę książkę!

- Historia projektu Ubuntu
- Instalacja systemu i proces uruchamiania Ubuntu
- System plików i administrowanie systemem
- Zarządzanie pakietami oprogramowania
- Automatyzacja procesu instalacji
- Konfiguracja serwera DNS, WWW
- Uruchomienie serwera poczty elektronicznej
- Konfiguracja serwera baz danych
- Umożliwienie dostępu zdalnego do serwera – OpenSSH
- Bezpieczeństwo w Ubuntu
- Wykrywanie włamań do systemu
- Monitorowanie pracy serwera
- Wirtualizacja – KVM, VMware Server
- Konfiguracja macierzy RAID
- Rozwiązywanie typowych problemów
- Tryb ratunkowy i odzyskiwanie
- Przydatne zasoby Ubuntu Server

Sprawdź możliwości Ubuntu Serwer!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 3

Zarządzanie pakietami

3

- **Wprowadzenie do zarządzania pakietami**
- **Pakiety systemu Debian**
- **Zarządzanie pakietami w Ubuntu**
- **Tworzenie własnych pakietów**

Na początku rozdziału zostaną przedstawione ogólne informacje na temat pakietów. Skoncentrujemy się na podstawowych funkcjach pakietów oraz systemach zarządzania pakietami dostępnych w większości dystrybucji GNU/Linux. Czytelnik dowie się, czym są pakiety oraz jakie są zadania systemów zarządzania pakietami. Wprawdzie przedstawione przykłady pochodzą z Ubuntu, ale analiza będzie dotyczyła ogólnej koncepcji kryjącej się za pakietami. Po solidnym omówieniu podstaw zostaną zaprezentowane pakiety projektu Debian — ten rodzaj pakietów jest stosowany w Ubuntu — a także pokrótce inne, odmienne rodzaje pakietów: kodu źródłowego oraz binarne. Natomiast w pozostałej części rozdziału skoncentrujemy się na zarządzaniu pakietami w Ubuntu za pomocą narzędzi powłoki. Wprawdzie wielu użytkowników biurkowej wersji Ubuntu zna procedurę uaktualnienia systemu, jednak w rozdziale będzie przedstawiona taka procedura, tyle że przeprowadzana bez użycia interfejsu graficznego. Czytelnik pozna podstawowe i bardziej zaawansowane sposoby używania systemów zarządzania pakietami, które wielu administratorów serwerów uzna za przydatne. Wreszcie ostatnie poruszone zagadnienie — tworzenie, modyfikowanie i rozprowadzanie własnych pakietów — będzie szczególnie interesowało zaawansowanych użytkowników i administratorów.

Wprowadzenie do zarządzania pakietami

W systemie Ubuntu — oraz innych środowiskach GNU/Linux — pakiety są podstawowym sposobem tworzenia, implementacji i instalacji oprogramowania. Niemal każda ważniejsza dystrybucja systemu operacyjnego GNU/Linux rozprowadza oprogramowanie w postaci pakietów; dotyczy to zarówno plików binarnych, jak i kodu źródłowego. Wspomniane pakiety są najczęściej w formacie RPM (na przykład w dystrybucji Red Hat) lub DEB (projekt Debian) dla oprogramowania binarnego oraz odpowiadających im formatów RPM i DEB dla kodu źródłowego. Ponieważ Ubuntu jest ściśle powiązane z projektem Debian i kontynuuje stosowanie jego osiągnięć, to jest oczywiste, że Ubuntu stosuje pakiety w formacie DEB.

Upraszczając, pakiety stanowią alternatywę dla procesu pobierania, budowania i instalacji oprogramowania zupełnie od zera. W stosunku do standardowego modelu „budowy na podstawie kodu źródłowego" pakiety oferują pewne istotne zalety pod względem instalacji, usuwania, monitorowania i wzajemnych relacji między oprogramowaniem. Ponieważ stosowanie pakietów nie jest aż tak bardzo rozpowszechnione poza światem GNU/Linux — lub przynajmniej nie jest przedstawiane w takich kategoriach — to przed przejściem do omawiania sposobu implementacji pakietów w Ubuntu warto wcześniej poznać pewne ogólne informacje dotyczące pakietów.

Ogólny opis pakietów

Niemal każdy system operacyjny bazujący na GNU/Linux — Fedora, RHEL, openSUSE, Slackware, Debian i inne — zawiera prawie że w całości taki sam zestaw oprogramowania podstawowego. Z definicji każdy z wymienionych systemów operacyjnych zawiera jądro opracowane przez Linusa Torvaldsa oraz dużą ilość oprogramowania w postaci projektów GNU, czyli aplikacji przeznaczonych dla programistów i użytkowników, a niezbędnych do zbudowania i używania

tego systemu. Większość wymienionych systemów operacyjnych zawiera także oprogramowanie typowo serwerowe, na przykład OpenSSH, Apache, implementację systemu X Window w postaci XFree86 lub X.org oraz bardzo często wyjątkowo obszerną kolekcję narzędzi powłoki i aplikacji graficznych. Warto w tym miejscu wyraźnie powiedzieć, że wymieniona kolekcja oprogramowania nosi nazwę dystrybucji. Ubuntu jest dystrybucją. Kiedy użytkownicy używają pojęcia „Linux" w odniesieniu do systemu operacyjnego, wówczas bardzo często mają na myśli system Linux lub dystrybucję GNU/Linux.

Podstawowym celem wszystkich dystrybucji jest zapewnienie automatycznej instalacji, konfiguracji, usuwania, obsługi i uaktualniania oprogramowania — zarówno poprzez dostarczenie przeznaczonej do tego infrastruktury, jak również utworzenie zmodyfikowanych wersji istniejącego już oprogramowania. Wspomniane modyfikacje istniejącego oprogramowania w taki specjalny sposób noszą nazwę „przygotowywania pakietów" i stanowią większą część zadań wykonywanych przez programistów Ubuntu. W ogromnym stopniu ma to wpływ na konkretną zawartość dystrybucji Ubuntu. Chociaż przygotowywanie pakietów to podstawowe zadanie przeprowadzane przez twórców dystrybucji takich jak Ubuntu, zadanie to może być wykonywane także przez użytkowników dystrybucji lub dostawców oprogramowania. Celem tych pierwszych jest zapewnienie czystej integracji oprogramowania niedostarczanego w pakietach, natomiast celem tych drugich jest ułatwienie użytkownikom procesu instalacji i obsługi danego oprogramowania.

Czym są pakiety?

Utworzenie pakietu — zarówno w Ubuntu, jak i innych dystrybucjach — rozpoczyna się od oprogramowania, które ma być umieszczone w pakiecie. W większości przypadków, ale nie zawsze, oznacza to nabycie odpowiedniego kodu źródłowego. We wszystkich sytuacjach kod trzeba pobrać z oryginalnego kodu źródłowego, co w świecie dystrybucji zwykle nosi nazwę „upstream". Następnym krokiem będzie utworzenie dodatkowych metadanych, w których najczęściej znajdują się następujące dane:

- nazwa programu;
- autor upstreamu oraz osoba tworząca pakiet;
- licencja na dane oprogramowanie;
- położenie upstream danego oprogramowania (lub opis przedstawiający w jaki sposób uzyskano kod źródłowy);
- architektura bądź architektury na których gwarantowane jest działanie danego oprogramowania;
- informacje odnośnie klasyfikacji oprogramowania, najczęściej związane z przeznaczeniem pakietu; podstawowym zadaniem tych informacji jest pomoc użytkownikom przeglądającym pakiety;
- opis oprogramowania w formacie możliwym do przetworzenia przez komputer;
- informacje na temat ważności, czyli „priorytet" pakietu w ramach większego systemu Ubuntu (na przykład wymagany, opcjonalny itp.).

Powyższe informacje będą używane przez system zarządzania pakietami lub dowolny program pozwalający użytkownikowi na wyszukiwanie, sortowanie i pracę z zainstalowanym bądź dostępnym do instalacji oprogramowaniem — to jedno z zadań systemu zarządzania pakietami. Jednak choć przedstawione powyżej metadane są ważne dla użytkowników, ponieważ pozwalają na uzyskanie dalszych informacji na temat danego oprogramowania, najważniejsza grupa metadanych dodawanych do pakietów dotyczy dokumentacji dotyczącej powiązania oprogramowania znajdującego się w pakiecie z innymi pakietami w dystrybucji. Wprawdzie składnia i semantyka różnią się w zależności od dystrybucji, ale zawarte tam informacje są podobne i odwołują się do:

- innego oprogramowania wymaganego do zbudowania danego programu;
- innego oprogramowania wymaganego do instalacji lub konfiguracji programu;
- innego oprogramowania wymaganego do uruchomienia danego programu;
- innego oprogramowania, z którym dany program nie może być jednocześnie zainstalowany bądź używany;
- innego oprogramowania, dla którego dany program może być zamiennikiem;
- innego oprogramowania, które może usprawnić lub rozbudować dany program.

Nowoczesne systemy zarządzania pakietami rejestrują jeszcze większą ilość informacji. Przykładowo w trakcie uaktualniania oprogramowania pliki konfiguracyjne w przeciwieństwie do zwykłych plików nie mogą być po prostu zastąpione nowszą wersją. Z tego powodu systemy zarządzania pakietami mają wbudowane infrastruktury odpowiedzialne za sprawdzanie i przechowywanie podstawowych informacji na temat konfiguracji. Te dane są wykorzystywane podczas uaktualniania pakietów wymagających wprowadzania zmian w plikach konfiguracyjnych. Wreszcie ostatnim zdefiniowanym celem pakietów jest dostarczenie struktury zbudowanej wokół metadanych pakietu — takich jak opisy — które mogą być tłumaczone w celu dostarczenia użytkownikowi interfejsu prowadzącego do oprogramowania zlokalizowanego w jego języku i kulturze. Szczegółowe informacje dotyczące tworzenia i uzyskiwania dostępu do tych wszystkich metadanych w pakietach Ubuntu będą przedstawione w kolejnych podrozdziałach.

Podstawowe funkcje zarządzania pakietami

Szeroka gama funkcji może być uznawana za podstawowe funkcje systemu zarządzania pakietami. Wspomniane funkcje najczęściej są implementowane przez narzędzia lub zestaw narzędzi niskiego poziomu. W przypadku projektów Ubuntu i Debian będzie to skrypt `dpkg` i powiązane z nim inne skrypty. W przeszłości były to podstawowe narzędzia używane przez większość użytkowników do przeprowadzania wszelkich zadań związanych z pakietami. Jednak sytuacja uległa zmianie kilka lat temu, gdy utworzono wysokiego poziomu narzędzia zarządzania pakietami. Wspomniane narzędzia zapewniają interfejs dla narzędzi niskiego poziomu. Większość użytkowników systemów operacyjnych bazujących na pakietach rzadko więc używa już narzędzi niskiego poziomu. Nadal są one stosowane przez programistów bądź administratorów, którzy chcą tworzyć własne pakiety. Ujmując ogólnie i bardzo nieprecyzyjnie, wiele z tych narzędzi niskiego poziomu odpowiada narzędziu `apt` używanemu w Ubuntu i Debianie.

Podstawowym celem pakietów jest automatyzacja kompilacji oprogramowania. Pakiety DEB są dostarczane w dwóch formatach: po jednym dla kodu źródłowego i programów binarnych. Pakiety kodu źródłowego są doskonałym systemem rozprowadzania i kompilacji kodu źródłowego. W systemie Ubuntu oraz innych pakiety zostały zaprojektowane do ich budowy w sposób nieinteraktywny, w przypadku oficjalnych pakietów Ubuntu mogą być budowane automatycznie dla wielu różnych architektur za pomocą oprogramowania nazywanego „autobilder".

Pakiety oferują prostą (zwykle w postaci pojedynczego polecenia) metodę tworzenia oprogramowania, która jest spójna i stosowana we wszystkich pakietach. Problemy związane z konfiguracjami i innymi opcjami są rozwiązywane wcześniej przez twórcę pakietu. Wadą jest przeprowadzanie konfiguracji w trakcie budowy pakietu, natomiast zalety są ogromne, o czym Czytelnik przekona się w pozostałej części rozdziału. Zależności wymagane w trakcie budowy programu są deklarowane w pakiecie, więc ich spełnienie może być zrealizowane automatycznie. Przykładowo pakiety źródłowe przeznaczone dla konkretnej architektury (na przykład pakiety, które muszą być ponownie zbudowane dla każdej architektury) są umieszczane w postaci kodu źródłowego Ubuntu i w większości przypadków bez wprowadzania jakichkolwiek zmian w pakiecie źródłowym są automatycznie budowane we wszystkich architekturach obsługiwanych przez Ubuntu.

Z pojedynczego pakietu danych źródłowych można zbudować dowolną liczbę pakietów binarnych. Możliwość tworzenia wielu pakietów binarnych na podstawie pojedynczego pakietu kodu źródłowego jest szczególnie użyteczna w ogromnych projektach, które wydają ogromne lub monolityczne pakiety kodu źródłowego zawierające szeroką gamę różnorodnego oprogramowania. To także ważne w przypadku ściśle powiązanych ze sobą elementów oprogramowania i (lub) dokumentacji, których rozdzielenie może być niebezpieczne. Przykładem może być tutaj system graficzny XFree86 — obecnie zastąpiony i już zmodularyzowany przez X.org — który był dostarczany w postaci pojedynczego pakietu kodu źródłowego, ale tworzył dziesiątki innych pakietów binarnych. W tym przypadku dostarczenie pakietu pozwala użytkownikom na rozprowadzanie, instalowanie i usuwanie serwera X niezależnie od emulatora terminalu, pakietu biblioteki xlib oraz menedżera okien.

Jak można wywnioskować na podstawie powyższych informacji, kluczową zaletą systemów zarządzania pakietami jest oferowana przez nie pomoc w automatyzacji instalacji oprogramowania. Kiedy pakiet binarny jest zainstalowany, wówczas:

- „Zawartość" tego oprogramowania może być zweryfikowana w celu zagwarantowania spójności pakietu. Pochodzenie oprogramowania można zweryfikować poprzez użycia uwierzytelniania kryptograficznego.
- Istnieje możliwość przeanalizowania zależności danego oprogramowania, a następnie system można sprawdzić pod kątem dostępności wymaganych komponentów innego oprogramowania koniecznych do poprawnego działania danego oprogramowania. Gdy zależności nie będą spełnione, użytkownik zostanie poinformowany o brakach i o tym, jakie oprogramowanie jest wymagane, a instalacja będzie przerwana[1].

[1] Niektóre systemy w takim przypadku proponują instalację brakujących komponentów — *przyp. tłum.*

- W pewnym momencie podczas procesu instalacji pakietu użytkownik może zostać zapytany o opcje konfiguracyjne. Odpowiedzi na takie pytania będą zapisane w systemie, a następnie użyte podczas dostosowywania pliku konfiguracyjnego instalowanego oprogramowania.
- Zawartość pakietu jest przechowywana w systemie.
- Metadane i powiązane z nimi informacje w różnych formach są umieszczane w bazie danych systemu, która zawiera bieżące informacje o zainstalowanych pakietach i ich stanie instalacji (na przykład zainstalowany, choć jeszcze nie skonfigurowany), listę plików należących do poszczególnych pakietów i inne informacje.

Prawdopodobnie najważniejsze jest tutaj sprawdzenie zależności instalowanego pakietu oraz obsługa listy pakietów zainstalowanych w systemie. W przypadku informacji dotyczących zależności użytkownik może dowiedzieć się, jakie oprogramowanie jest wymagane do uruchomienia oprogramowania znajdującego się w instalowanym pakiecie. Z tego powodu osoby tworzące oprogramowanie przeznaczone do umieszczania pakietów mogą bardzo łatwo tworzyć i implementować oprogramowanie z uwzględnieniem bibliotek współdzielonych. Sukces systemów zarządzania oprogramowaniem to jeden z powodów powszechnego wykorzystywania w środowisku GNU/Linux dynamicznie dołączanych bibliotek współdzielonych.

Kiedy użytkownik chce usunąć wybrane oprogramowanie, system zarządzania pakietami wraz ze swoim katalogiem zawierającym listę plików pakietu i działań przeprowadzonych w trakcie instalacji jest doskonale przygotowany do zapewnienia użytkownikowi pomocy w celu zagwarantowania przeprowadzenia procesu pełnego usunięcia danego oprogramowania.

Proces automatycznego uaktualniania oprogramowania jest podobny do instalacji i stanowi kolejny obszar, na którym z powodzeniem można wykorzystać system do zarządzania pakietami. Dzięki temu użytkownik takiego systemu może bezpiecznie i bardzo łatwo przeprowadzić uaktualnienie oprogramowania z jednej wersji do innej. Proces uaktualniania oprogramowania będzie niemal identyczny z procesem jego instalacji. W większości przypadków instalowane oprogramowanie jest zapisywane w miejsce istniejącego pakietu, a pliki, które nie znajdują się w nowszej wersji pakietu, są usuwane. Pliki konfiguracyjne zmodyfikowane przez proces instalacyjny i od tego czasu nie zmieniane przez użytkownika mogą być automatycznie wygenerowane przez użytkownika. Ewentualnie użytkownik może zostać poproszony o przejrzenie i zatwierdzenie zmian.

Informacje dotyczące zależności mogą odgrywać ważną rolę w procesie aktualizacji pakietu wykorzystującego biblioteki współdzielone. W przypadku zmian ABI[2] (ang. *Application Binary Interface*) system zarządzania pakietami poinformuje użytkownika, że proces uaktualnienia pakietu nie będzie mógł być zakończony bez instalacji nowej biblioteki. Ponadto użytkownik będzie mógł zostać ostrzeżony o możliwości ewentualnego uszkodzenia innych pakietów w wyniku tej zmiany. Z tego powodu użytkownicy mogą kształtować uaktualnienia — lub system zrobi to za nich — aby nie dochodziło do nieoczekiwanych uszkodzeń spowodowanych przez API

[2] ABI to zestaw reguł wpływających na współpracę między programami i bibliotekami, oprogramowaniem i systemem operacyjnym lub między różnymi komponentami oprogramowania i dotyczy programów skompilowanych — *przyp. tłum.*

(ang. *Application Programming Interface*, interfejs programowania aplikacji) i ABI. W ten sposób użytkownicy mają gwarancję, że wszystkie pakiety wykorzystujące daną bibliotekę współdzieloną będą mogły być bezpieczne uaktualniane.

Wreszcie w dowolnej chwili użytkownik może sprawdzić podpis cyfrowy pakietu, wyświetlić wartości hash (zwykle sumy kontrolne MD5) plików znajdujących się w pakiecie i tym samym zweryfikować spójność plików w systemie pod kątem ewentualnego uszkodzenia bądź zmodyfikowania w wyniku ataku.

Zaawansowane funkcje systemów zarządzania pakietami

Wprawdzie wymienione funkcje dają ogromny potencjał podczas zarządzania oprogramowaniem w systemie, ale systemy zarządzania pakietami wyposażone *jedynie* w takie funkcje — czyli w zasadzie przedstawiające stan zarządzania pakietami w połowie lat dziewięćdziesiątych ubiegłego stulecia — nakładają istotne ograniczenia. Przeobrażenia API i ABI na dużą skalę wymagają pobrania wielu pakietów oraz wysokiego stopnia koordynacji ze strony użytkownika. Podczas instalacji bądź uaktualniania oprogramowania użytkownicy byli zmuszani do określenia stanu zależności programu, a następnie wyszukiwania, pobierania i jednoczesnej instalacji nowych elementów oprogramowania. W przypadku skomplikowanego oprogramowania z wieloma zależnościami ten proces bardzo często był zbyt męczący.

W wyniku tego większość uaktualnień systemu oraz zmian ABI/API była przeprowadzana za pomocą ogromnych skryptów uaktualnień między kolejnymi wydaniami dystrybucji. Użytkownik oczekiwał, że w trakcie większego uaktualnienia każdy wymagany pakiet zostanie zainstalowany przez skrypt aktualizacji, którego struktura pozwoli na zachowanie właściwej kolejności i odpowiednie obsłużenie zależności. Wprawdzie wspomniane problemy wynikały z ograniczeń samego systemu zarządzania pakietami, jednak większość z tych problemów występowała też poza systemami zarządzania pakietami. Bez systemu zarządzania pakietami biblioteki współdzielone poddawane zmianom API i ABI były akceptowane bardzo rzadko bądź wcale (z powodów bezpieczeństwa lub konieczności zagwarantowania spójności). Ewentualnie podlegały takim samym ograniczeniom, ale bez ostrzeżeń generowanych przez system zarządzania pakietami.

W ostatnich latach jesteśmy świadkami poważnej rewolucji związanej z rozwojem systemów zarządzania pakietami. Wynika to po części z faktu utworzenia w ramach projektu Debian programu o nazwie `dselect` i często chwalonego zestawu narzędzi APT (ang. *Advanced Package Tools*, początkowo o nazwie *deity*), które zaimplementowano w programie o nazwie `apt-get`. Większość z tych narzędzi do rodzaj pewnych form „interfejsu" dla poprzednio omówionych narzędzi niskiego poziomu służących do zarządzania pakietami. Podobnie jak większość innych dystrybucji bazujących na pakietach DEB, również Ubuntu używa `apt-get`, Aptitude, `dselect` oraz graficznego interfejsu Synaptic.

Możliwość śledzenia i katalogowania zależności to prawdopodobnie najważniejszy aspekt każdego systemu zarządzania pakietami. Podstawową funkcją narzędzi zaawansowanych było dostarczenie dodatkowej funkcjonalności istniejącym narzędziom zarządzania pakietami oraz możliwość jednoczesnego operowania wieloma pakietami. Każde z narzędzi zaawansowanych zawiera

dodatkowe bazy danych przechowujące nie tylko opis zainstalowanych pakietów, ale również pakiety *dostępne* jako kandydujące do instalacji za pomocą archiwów pakietów znajdujących się lokalnie, na płytach CD lub (obecnie najczęściej) w repozytoriach sieciowych.

Systemy zarządzania pakietami mają możliwość automatycznego rozwiązywania problemów z zależnościami i kolejnością ich instalowania, pobierania pakietów (w tym również zależności), instalacji w pierwszej kolejności wszystkich zależności, a dopiero następnie instalacji i konfiguracji wybranego pakietu oprogramowania. Cały proces jest przeprowadzany za pomocą narzędzi niskiego poziomu omówionych w poprzedniej sekcji.

Te same narzędzia zaawansowane mogą być również wykorzystywane do usuwania pakietów. Przykładowo: jeżeli użytkownik chce usunąć bibliotekę współdzieloną, wówczas zostanie wyświetlony komunikat informujący o konsekwencjach takiego kroku wraz z listą pakietów, które będą musiały być odinstalowane z powodu niespełnienia zależności po usunięciu żądanej biblioteki. Uaktualnienia obejmujące zmianę zależności (na przykład zastąpione pakiety) również mogą być obsługiwane za pomocą takiego systemu.

Rzeczywiste możliwości tego rodzaju systemów ujawniają się, gdy aspekty zależności pakietu ulegają zmianom w czasie lub jeśli wiele pakietów może być zamiennikami. Pakiet wymagający możliwości wysyłania wiadomości e-mail może wymagać zależności w postaci pakietu wirtualnego „dostarczanego" przez inne pakiety. Nowsze wersje pakietu mogą powodować konflikty mimo deklaracji o „zastępowaniu" innych pakietów lub dostarczaniu funkcjonalności oryginalnego pakietu. Przykładowo: jeżeli kilka pakietów zostanie połączonych w jeden i pozostałe staną się zbędne, zaawansowany system zarządzania pakietami powinien śledzić zmiany w danych dotyczących zależności i właściwie zareagować w trakcie procesu uaktualniania. Oprócz tego zaawansowane systemy zarządzania pakietami dają użytkownikom możliwość przeprowadzania strategicznych „inteligentnych uaktualnień" każdego pakietu w systemie do najnowszej dostępnej wersji przy użyciu danych zadeklarowanych w zależnościach pakietu.

Dla niektórych użytkowników jeszcze bardziej ekscytującą możliwością jest śledzenie opracowywanych właśnie wersji systemu operacyjnego GNU/Linux i jego uaktualnianie każdego dnia do najnowszej dostępnej wersji. System zarządzania pakietami może określić bezpieczny sposób uaktualnienia i następnie je przeprowadzić. W trakcie uaktualniania zmiany wersji ABI i API również mogą być obsłużone automatycznie, ponieważ system odmówi przeprowadzenia pełnej aktualizacji biblioteki, jeśli wszystkie pakiety zainstalowane w systemie i zależne od danej biblioteki współdzielonej nie będą mogły być uaktualnione jednocześnie. System nie musi więc zawierać lub śledzić wielu wersji biblioteki współdzielonej.

Pakiety systemu Debian

Jak już wcześniej wspomniano, projekt Ubuntu jest zbudowany na bazie dystrybucji GNU/Linux Debian. Wśród wielu innych technologicznych spuścizn system Ubuntu odziedziczył także system pakietów Debiana. W rzeczywistości wielu głównych programistów zaangażowanych na

samym początku w projekt Ubuntu preferowało system pakietów Debiana, gdyż stanowił on atrakcyjny punkt wyjścia i wyróżniał się wśród wielu innych dystrybucji GNU/Linux. Z tego powodu niemal wszystkie aspekty zarządzania pakietami — począwszy od formatów aż po narzędzia — są identyczne w projektach Debian i Ubuntu. W wielu przypadkach niezmodyfikowany pakiet Debian może być po prostu zainstalowany w Ubuntu. W niemal wszystkich sytuacjach niezmodyfikowany pakiet kodu źródłowego Debiana może być zbudowany w systemie Ubuntu. Tak więc pierwszym krokiem będzie tutaj dokładne omówienie formatu Ubuntu DEB, aby Czytelnik zrozumiał anatomię pakietu oraz sposób implementacji funkcji przedstawionych w poprzednich sekcjach.

Pakiety kodu źródłowego

Pakiety DEB zawierające kod źródłowy są zwykle wyrażone w postaci formatu składającego się z dwóch lub trzech plików, choć równie dobrze pakiety mogą składać się ze znacznie większej liczby plików. Oznacza to, że sam pakiet zawiera pewną liczbę plików, a pobranie „pakietu kodu źródłowego" może w rzeczywistości wymagać pobrania wielu różnych, małych plików. Pakiety kodu źródłowego mogą być sklasyfikowane jako *rodzime* lub *nierodzime* pakiety DEB. Różnica między nimi polega na tym, że rodzimy DEB to pakiet oprogramowania, gdzie wersje upstream i w pakiecie DEB absolutnie się nie różnią. W większości przypadków rodzime pakiety są charakterystyczne dla Ubuntu, Debiana lub innej dystrybucji bazującej na Debianie. Innymi słowy, rodzimy pakiet DEB nie wymaga wprowadzania żadnych zmian w celu możliwości utworzenia pakietu. Pakiet DEB kodu źródłowego zawsze składa się z archiwum „pierwotnego" kodu źródłowego w formacie skompresowanego przez GNU gzip pliku tar (plik z rozszerzeniem *.tar.gz*) oraz pliku DSC, w którym wymieniono zawartość pakietu i który może być uważany za „jądro" pakietu kodu źródłowego. Przykładowa zawartość pliku DCS dla dostarczanego przez autora programu o nazwie `most` przedstawia się następująco:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Format: 1.0
Source: most
Binary: most
Architecture: any
Version: 5.0.0a-1
Maintainer: Benjamin Mako Hill <mako@debian.org>
Standards-Version: 3.7.3
Build-Depends: debhelper (>= 4), libslang2-dev
Files:
30f2131b67f61716f6fe1f65205da48b 155233 most_5.0.0a.orig.tar.gz
↪07e3eb05ad5524fe6d885f5cdc2eb902 20160 most_5.0.0a-1.diff.gz

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)
```

```
iD8DBQFH4IoAic1LIWB1WeYRAjyOAKCrLCfuZA7b8JcvYTFYeuHrF7r34wCfVTBS
↪/jGUfIrELNq173sM9CorZA4= =/Cia
-----END PGP SIGNATURE-----
```

Powyższy plik jest podpisany za pomocą klucza GnuPGP w celu zapewnienia spójności i identyfikacji jego autora. Jeżeli Czytelnik sprawdzi podpis za pomocą GPG, to przekona się, że plik został podpisany z użyciem klucza PGP autora pakietu. Plik DSC zawiera także informacje dotyczące wersji formatu źródłowego (w tym przypadku to „stary" format 1.0), nazwę pakietu kodu źródłowego, wersję pakietu (podzieloną na wersję kodu źródłowego upstream oraz wersję pakietu po znaku minus), imię i nazwisko oraz adres e-mail twórcy pakietu oraz informacje na temat architektury, w oparciu o którą działa dane oprogramowanie. Ponadto plik zawiera informacje o wersji polityki (oznaczonej jako „Standards") zastosowanej podczas tworzenia oprogramowania, informacje o oprogramowaniu wymaganym do zbudowania tego pakietu, a także listę innych plików znajdujących się w tym pakiecie kodu źródłowego wraz z ich wielkościami oraz sumami kontrolnymi MD5.

W rodzimym pakiecie DEB znajduje się tylko jeden skompresowany plik tar (*.tar.gz*). W przypadku nierodzimych pakietów dostępne będą pliki dodatkowe, przedstawiające wszystkie zmiany wprowadzone w pakiecie. W ten sposób wszystkie zmiany wprowadzone przez twórcę pakietu są w pełni jawne i widoczne. Czasami wynika to z warunków licencji, ale zwykle celem jest to, aby inni użytkownicy dokładnie wiedzieli, co zrobił twórca pakietu oraz co zostało zmienione. W ten sposób osoba odpowiedzialna za pakiet może łatwiej wychwycić ewentualny błąd w pakiecie. Zmiany wprowadzone w pakiecie zazwyczaj są przedstawione w skompresowanym za pomocą gzip pliku diff, w którym wymienione są wszystkie różnice między kodem źródłowym pakietu i pierwotnym kodem źródłowym. W przypadku przedstawionego powyżej pakietu plik diff to *most_5.0.0a-1.diff.gz*. W nowszych wersjach formatu pakietu DEB dla kodu źródłowego dozwolone jest umieszczanie dodatkowych plików tar przedstawiających dodatki lub zmiany w stosunku do pierwotnego kodu źródłowego, o ile będą one wymienione w pliku DSC na znajdującej się tam liście plików.

Po rozpakowaniu i wprowadzeniu wszystkich wymaganych poprawek każdy pakiet DEB kodu źródłowego zostanie rozpakowany do pojedynczego katalogu o nazwie pakietu wraz z numerem wersji (na przykład *most-5.0.0a*) i obowiązkowym podkatalogiem o nazwie *debian*. W większości pakietów niemal wszystkie zmiany w kodzie źródłowym są przeprowadzane właśnie w wymienionym katalogu. Sam katalog zawiera pewną liczbę plików — jest ich znacznie więcej niż miejsca na ich omówienie tutaj. Najważniejsze z nich to pliki *control* i *rules*. Większość informacji o pakiecie kodu źródłowego znajdujących się w pliku *control* można znaleźć także w pliku DSC, który jest generowany automatycznie na podstawie danych pliku *control*; natomiast informacje dodatkowe opisują każdy pakiet binarny. Plik *control* przedstawia więc wszystkie powiązania pomiędzy pakietami, między innymi *Depends*, *Conflicts*, *Provides*, *Replaces*, *Recommends*, *Suggests* oraz *Enhances*. Pierwsze cztery są określane jako „twarde" i są najważniejsze. Natomiast *Suggests* i *Enhances* są rzadko używane przez jakikolwiek program. W pliku *control* można także znaleźć opis programu, zarówno w postaci skróconej do pojedynczego wiersza, jak i dłuższą wersję. Poniżej przedstawiono przykładową zawartość pliku *control* (tutaj dla wspomnianego już wcześniej programu `most`):

```
Source: most
Section: text
Priority: optional
Maintainer: Benjamin Mako Hill <mako@debian.org>
Standards-Version: 3.7.3
Build-Depends: debhelper (>=4), libslang2-dev

Package: most
Architecture: any
Depends: ${shlibs:Depends}
Description: Pager program similar to more and less
```

Dłuższa wersja opisu została usunięta z powyższych danych wyjściowych, ale w rzeczywistości ten tekst znajduje się poniżej pokazanego tutaj ostatniego wiersza, jest wcięty o jedną spację, a każdy akapit jest oddzielony pojedynczą kropką. Jak już wcześniej wspomniano, plik *control* składa się z serii akapitów. Pierwszy zawsze rozpoczyna się od słowa `Source:` i zawiera informacje dotyczące pakietu kodu źródłowego. Każdy kolejny przedstawia opis pojedynczego pakietu binarnego. W omawianym przykładzie jest tylko jeden pakiet binarny, który podobnie jak pakiet kodu źródłowego ma nazwę `most`. Taka sytuacja — pojedynczy pakiet kodu źródłowego tworzący pojedynczy pakiet binarny o takiej samej nazwie — jest bardzo często spotykana.

Plik *rules* to po prostu plik GNU Make makefile. Zawiera wszystkie reguły makefile potrzebne do utworzenia i zbudowania pakietu. Uruchomienie pliku binarnego w rozpakowanym katalogu pakietu kodu źródłowego skutkuje utworzeniem pakietu Debian, który będzie umieszczony w katalogu nadrzędnym (`../`), o ile system ma zainstalowane wszystkie wymagane programy zależne. W większości przypadków oprogramowanie zostanie zbudowane i „zainstalowane" w serii podkatalogów w katalogu *debian*. Następnie te pliki w ich położeniu tymczasowym będą umieszczone w pakiecie jako jego zawartość.

Dodatkowe pliki w katalogu *debian* obejmują pliki z informacjami na temat praw autorskich, zmian wprowadzonych w pakiecie, opcjonalne skrypty uruchamiane przed instalacją lub usunięciem pakietu lub po tych czynnościach, dodatkowe dane konfiguracyjne plus to wszystko, co twórca pakietu chce w nim umieścić.

Pakiety binarne

Pakiety binarne Debiana są bardzo prostym formatem, więc nie trzeba im tutaj poświęcać zbyt dużo miejsca. Co ważniejsze, prawie nigdy nie są modyfikowane ręcznie. Pakiety binarne są jedynie instalowane i usuwane. Aby wprowadzić zmiany w pakiecie binarnym, w pierwszej kolejności trzeba zmodyfikować pakiet kodu źródłowego, a następnie na jego podstawie zbudować nowy pakiet binarny. W systemach Ubuntu i Debian pakiet binarny to pojedynczy plik w archiwum w formacie *ar*. Wymienione archiwum zawiera plik *debian-binary* z serią wierszy rozdzielonych znakami nowego wiersza. W chwili obecnej ten plik przechowuje jedynie numer wersji formatu.

Drugi plik wymienionego archiwum to również archiwum, tym razem o nazwie *control.tar.gz*; zawiera on informacje kontrolne o pakiecie (zgodnie z przedstawionym wcześniej opisem). Trzeci i ostatni element archiwum nosi nazwę *data.tar.gz* i zawiera archiwum systemu plików.

Zarządzanie pakietami w Ubuntu

Administrator każdej instalacji Ubuntu, zarówno serwerowej, jak i biurkowej, musi poznać podstawowe mechanizmy zarządzania pakietami. Ponieważ administrator musi znajdować nowe oprogramowanie w celu rozwiązywania określonych problemów, metadane w systemie pakietów mogą być dobrym punktem wyjścia. Kiedy administrator chce zainstalować nowe oprogramowanie, system zarządzania pakietami jest najlepszym sposobem realizacji takiego zadania. Oferowany w Ubuntu system pozwala użytkownikom na instalację i usuwanie oprogramowania, sprawdzanie dostępności aktualizacji i ich aktualizację — w szczególności dotyczy to aktualizacji bezpieczeństwa. Wreszcie, po wydaniu nowej wersji Ubuntu, system zarządzania pakietami pozwala na przeprowadzenie aktualizacji do nowszego wydania systemu.

Ubuntu oferuje różnorodne narzędzia przeznaczone do zarządzania pakietami. W biurkowej wersji systemu użytkownicy pracują z systemem zarządzania pakietami za pomocą małej ikony widocznej na pulpicie, która informuje o dostępności nowych wersji oprogramowania. Ponadto w systemie znajduje się aplikacja graficzna pozwalająca na instalację i usuwanie oprogramowania, a także graficzny program do zarządzania pakietami, o nazwie Synaptic, który pozwala użytkownikom na przeglądanie archiwów pakietów. Ponieważ wspomniane programy zostały szczegółowo omówione w książce *Ubuntu. Oficjalny podręcznik. Wydanie V* (również wydanej przez Helion), a niniejsza książka dotyczy serwerowego wydania Ubuntu, tutaj skoncentrujemy się na działających w powłoce narzędziach służących do przeglądania pakietów i zarządzania nimi.

Do zarządzania pakietami na wysokim poziomie większość administratorów wykorzystuje przede wszystkim narzędzia z rodziny APT. Oryginalnym narzędziem opracowanym do tego celu było `apt-get`. Aptitude to często stosowana alternatywa dla narzędzia `apt-get`, która oferuje zarówno interaktywny interfejs, jak i większość funkcji znajdujących się w `apt-get`. Wiele poleceń przedstawionych w pozostałej części rozdziału, a wywołujących `aptitude` może być używanych również z `apt-get`. Sposób działania i dane wyjściowe obydwu narzędzi są takie same bądź jedynie z niewielkimi różnicami. Podstawowe różnice między wymienionymi narzędziami dotyczą sposobów rozwiązywania skomplikowanych problemów z zależnościami pakietów, co na pewno nie ma żadnego wpływu na całkiem proste operacje omówione w tym rozdziale.

Zapewnienie aktualności systemu

Każdy system Ubuntu w pliku */etc/apt/sources.list* przechowuje listę repozytoriów pakietów. Wspomniany plik zawiera listę „miejsc", w których systemy zarządzania pakietami — początkowo po prostu APT, a teraz także kilka innych narzędzi — będą szukały uaktualnionych wersji

oprogramowania. Źródła oprogramowania mogą zawierać repozytoria lokalne w systemie plików, na płycie CD umieszczonej w napędzie komputera i — jak ma to miejsce w większości sytuacji — pod określonym adresem sieciowym. W celu uaktualnienia listy pakietów należy wydać polecenie `apt-get update` lub `aptitude update`.

Wymienione polecenia powodują pobranie uaktualnionych list pakietów dla wszystkich repozytoriów wymienionych w pliku */etc/apt/sources.list* i sprawdzają wszystkie podpisy kryptograficzne tych uaktualnień względem kluczy znajdujących się w systemie. W nowym systemie sprawdzane są tylko repozytoria pakietów Ubuntu zawierające uaktualnienia bezpieczeństwa.

Instalacja nowej wersji pakietu jest bardzo prosta i sprowadza się do wykonania polecenia `aptitude safe-upgrade`, które jest odpowiednikiem polecenia `apt-get update` znanego bardziej doświadczonym użytkownikom. Argument `safe-upgrade` próbuje uaktualnić wszystkie zainstalowane pakiety do ich najnowszych dostępnych wersji. Zainstalowane pakiety nie będą usuwane, o ile nie były używane, choć w celu spełnienia wszystkich zależności uaktualnianego pakietu mogą zostać zainstalowane inne.

Narzędzie `apt` może zostać skonfigurowane do automatycznego pobierania i uaktualniania pakietów do ich najnowszych dostępnych wersji. To jest atrakcyjna propozycja dla administratorów, którzy nie chcą logować się do systemu tylko w celu przeprowadzenia jego aktualizacji. Jednak automatyczna aktualizacja jest podatna na błędy wynikające z istnienia określonego oprogramowania w systemie lub nawet wprowadzenia pewnych zmian w konfiguracji. W takich przypadkach proces automatycznej aktualizacji pakietów może zakończyć się pozostawieniem systemu w stanie niestabilnym bądź spowodować brak działania systemu. Dlatego też uaktualnienia automatyczne nie są zalecane przez autorów i nie będą omówione w książce.

Wyszukiwanie i przeglądanie

Wcześniej stosowanym podstawowym sposobem wyszukiwania nowych pakietów było użycie programu o nazwie `dselect`. Obecnie użytkownicy biurkowych wydań Ubuntu stosują graficzne narzędzie do dodawania i usuwania programów[3], a także graficzny menedżer pakietów o nazwie Synaptic. Użytkownicy konsoli mają kilka dodatkowych opcji.

Pierwszą z nich jest prosty program o nazwie `apt-cache`, dostarczający danych statystycznych dotyczących pakietów. Przykładowo: jeżeli użytkownik będzie chciał znaleźć program stronicujący wyświetlane dane, taki jak less, wówczas można wydać następujące polecenie:

```
$ apt-cache search pager less
less - Pager program similar to more
wdiff - Compares two files word by word
console-log - Puts a logfile pager on virtual consoles
gdesklets-data - Applets for gdesklets
```

[3] W nowych wydaniach Ubuntu wprowadzono nowe narzędzie *Centrum oprogramowania Ubuntu* pozwalające na instalację i usuwanie oprogramowania — *przyp. tłum.*

```
jless - A file pager program, similar to more(1) supporting ISO2022
most - Pager program similar to more and less
nagios-plugins-basic - Plugins for the nagios network monitoring
and management system
```

Jak można zobaczyć na powyższej liście, polecenie `apt-cache search` zwróciło osiem „trafionych" wyników wyszukiwania dwóch słów kluczowych *pager* i *less*. Dane wyjściowe są przedstawione w postaci listy nazw pakietów wraz z krótkim opisem. Słowo kluczowe było szukane na pełnej liście pakietów, proces wyszukiwania skoncentrował się na nazwach pakietów, opisach krótkich i długich, choć te ostatnie nie znajdują się w danych wyjściowych polecenia. Użytkownik może dowiedzieć się więcej na temat określonego pakietu, ponieważ po użyciu argumentu `show` narzędzie `apt-cache` wyświetli więcej informacji na temat wskazanego pakietu, na przykład jak przedstawiono poniżej:

```
$ apt-cache show most
Package: most
Priority: optional
Section: universe/text
Installed-Size: 172
Maintainer: Ubuntu MOTU Developers <ubuntu-motu@lists.ubuntu.com>
Original-Maintainer: Benjamin Mako Hill <mako@debian.org>
Architecture: i386
Version: 5.0.0a-1
Depends: libc6 (>= 2.7), libslang2 (>= 2.0.7-1)
Filename: pool/universe/m/most/most_5.0.0a-1_i386.deb
Size: 48092
MD5sum: e089c00005b536e1b8848b7087df2bae
SHA1: 4f4ab395f340be4804732452aa112007916f90cb
SHA256:
ccf50fb49270e7ddf7735da23e699afcd11dcfc8e241973bb17ad03bf49e6f4a
Description: Pager program similar to more and less
Most is a paging program that displays, one windowful at a time, the
contents of a file on a terminal. A status line at the bottom of the
screen displays the file name, the current line number, and the
percentage of the file so far displayed.
.
Unlike other paging programs, most is capable of displaying an
arbitrary number of windows as long as they all fit on the screen,
and different windows could be used to view the same file in
different positions.
.
In addition to displaying ordinary text files, most can also display
binary files as well as files with arbitrary ascii characters.
Bugs: mailto:ubuntu-users@lists.ubuntu.com
Origin: Ubuntu
```

Czytelnik może dostrzec, że całkiem spora ilość informacji przedstawionych powyżej pokrywa się z informacjami umieszczonymi w pakiecie kodu źródłowego oraz w pliku *control* pakietu binarnego (co zostało dokładnie omówione w poprzednim podrozdziale). Oczywiście, w tym miejscu są wyodrębniane metadane zaszyte w pakietach.

Większość danych wyjściowych omawianego polecenia stanowi treść długiego opisu pakietu, która została pominięta we wcześniejszym przykładzie. Dane zawierają jeszcze kilka innych interesujących pól, na przykład *Original-Maintainer* informujące o osobie przygotowującej dany pakiet w systemie Debian lub pole *Maintainer* informujące o osobie bądź grupie, z którą można się kontaktować w sprawach dotyczących pakietu. Ponadto znaleźć można tutaj informacje o wielkości pakietu, sumach kontrolnych (na przykład MD5Sum, SHA1, SHA256) zapewniających sposób identyfikacji danej wersji pakietu, co pozwala na upewnienie się, że pakiet został pobrany prawidłowo i nie został zmodyfikowany przez nieupoważnioną osobę.

Po wywołaniu bez argumentów narzędzie Aptitude dostarcza użytkownikowi interfejs tekstowy bazujący na bibliotece Curses, który pozwala na bardziej interaktywne przeglądanie wszystkich dostępnych pakietów. Użytkownicy znających interfejs programu Synaptic mogą to potraktować jako tekstową wersję interfejsu menedżera Synaptic. W tym trybie działania wiele wyników wyszukiwania można przeglądać za pomocą klawiszy kursora i „zaznaczać" różne programy do instalacji.

Przed kontynuacją przedstawiania kolejnych opcji służących do wyszukiwania i przeglądania pakietów warto wspomnieć o witrynie internetowej *http://packages.ubuntu.com/*. Interfejs zastosowany na wymienionej witrynie pozwala użytkownikom na wyszukiwanie pakietów w podobny sposób jak przy użyciu omówionych tutaj narzędzi, choć oferuje kilka dodatkowych, użytecznych opcji. Wymieniona witryna internetowa pozwala więc użytkownikowi na wyszukanie określonego pliku w *dowolnym* pakiecie Ubuntu. Zwykle użytkownicy mogą wyszukiwać pliki jedynie w „posiadanych" pakietach, czyli pakietach zainstalowanych w systemie. Przykładowo: jeśli użytkownik potrzebuje określonego pliku nagłówkowego lub biblioteki współdzielonej, a zna jedynie nazwę pliku, wówczas może przeszukać wszystkie pakiety dostępne w archiwum Ubuntu, używając wymienionej witryny internetowej pod kątem znanej nazwy pliku.

Instalacja i usuwanie

Instalacja i usuwanie pakietów to kolejne proste zadanie, które administrator będzie często wykonywał. W celu instalacji pakietu można wywołać polecenie `apt-get` lub `aptitude` w sposób podobny do przedstawionego wcześniej. Jednak instalacja, w przeciwieństwie do wyszukiwania, wymaga uprawnień użytkownika root. Zalecanym sposobem uzyskania tych uprawnień jest użycie polecenia `sudo`. Ponieważ poprzedzanie każdego wydawanego w tej sekcji polecenia poleceniem `sudo` byłoby żmudne, przyjęto założenie, że bieżący użytkownik to root. Warto pamiętać, że logowanie się jako użytkownik root nie jest uznawane za najlepsze rozwiązania. W celu instalacji programu `most` wystarczy wydać poniższe polecenie (trzeba pamiętać o uprawnieniach użytkownika root):

```
# aptitude install most
Czytanie list pakietów... Gotowe
Budowanie drzewa zależności
Odczyt informacji o stanie... Gotowe
Odczyt dodatkowych informacji o stanie
Inicjalizacja stanów pakietów... Gotowe
```

```
Następujące NOWE pakiety zostaną zainstalowane:
  libslang2{a} most
0 pakietów aktualizowanych, 2 instalowanych, 0 do usunięcia i 0
↪nieaktualizowanych.
Do pobrania 509kB archiwów. Zajęte po rozpakowaniu: 1323kB.
Kontynuować? [Y/n/?] y
Zapisywanie dodatkowych informacji o stanie... Gotowe
Zaznaczenie poprzednio niezaznaczonego pakietu libslang2.
(Odczytywanie bazy danych ... 125915 plików i katalogów obecnie
↪zainstalowanych.)
Rozpakowanie libslang2 (z .../libslang2_2.1.3-3ubuntu1_i386.deb) ...
Konfigurowanie libslang2 (2.1.3-3ubuntu1) ...
Zaznaczenie poprzednio niezaznaczonego pakietu most.
(Odczytywanie bazy danych ... 125915 plików i katalogów obecnie
↪zainstalowanych.)
Rozpakowanie most (z .../most_5.0.0a-1_amd64.deb) ...
Przetwarzanie wyzwalaczy dla man-db...
Konfigurowanie most (5.0.0a-1) ...

Czytanie list pakietów... Gotowe
Budowanie drzewa zależności
Odczyt informacji o stanie... Gotowe
Odczyt dodatkowych informacji o stanie
Inicjalizacja stanów pakietów... Gotowe
Zapisywanie dodatkowych informacji o stanie... Gotowe
```

Z powyższych danych wyjściowych wynika, że wraz z programem `most` została zainstalowana biblioteka `libslang2`. W tym konkretnym przypadku narzędzie `aptitude` „zauważyło" brak wymaganej biblioteki S-Lang. Narzędzie poprosiło więc użytkownika o potwierdzenie instalacji pakietu dodatkowego (na co użytkownik się zgodził), następnie pobrało oba wymagane pakiety, zainstalowało i skonfigurowało je w systemie.

Usunięcie pakietu również zalicza się do prostych operacji. Jeżeli użytkownik zdecyduje się na usunięcie programu `most` z systemu, wówczas wystarczy po prostu wydać polecenie:

```
# aptitude remove most
```

Po wydaniu powyższego polecenia biblioteka `libslang2` *nie* zostanie usunięta (ponieważ użytkownik nie poprosił o jej usunięcie). Natomiast w przypadku próby usunięcia samej biblioteki `libslang2` narzędzie `aptitude` poinformuje użytkownika o konieczności usunięcia również wszystkich pakietów zależnych od wymienionej biblioteki. W tym konkretnym systemie będzie to tylko program `most`, ale w przypadku innych pakietów lub w innym systemie konieczne może być usunięcie większej liczby innych pakietów. Taki rodzaj zarządzania zależnościami oznacza, że użytkownik nie powinien (i tak łatwo nie może) usuwać pakietów podstawowych czy istotnych dla funkcjonowania systemu. Pakiety „nieużywane" mogą być usunięte za pomocą polecenia `apt-get autoremove`.

Wprawdzie w powyższych przykładach użyto narzędzia `aptitude`, ale instalację i usuwanie pakietów można przeprowadzać również za pomocą narzędzia niższego poziomu o nazwie `dpkg`. W rzeczywistości w tle narzędzie `aptitude` po prostu wywołuje polecenie `dpkg` i używa go do wykonywania zadań. Narzędzie `aptitude` — a także `apt-get` — zawsze w pierwszej kolejności pobierze pakiety i rozwiąże wszelkie problemy z zależnościami i dopiero wtedy zleci odpowiednie zadanie narzędziu `dpkg`. W przypadku posiadania już zainstalowanych pakietów zależnych pakiet DEB można zainstalować bezpośrednio za pomocą `dpkg`, używając opcji `-i`, a następnie nazwy pliku jako argumentu. Przykładowo pakiet DEB z programem `most` można zainstalować, wydając poniższe polecenie:

```
$ dpkg -i most_5.0.0a-1_i386.deb
```

Narzędzie `dpkg` sprawdzi zależności; w przypadku ich niespełnienia zostanie wygenerowany komunikat błędu. Jednak narzędzie nie będzie automatycznie pobierało lub instalowało pakietów, ponieważ nie ma wbudowanych do tego funkcji. Usunięcie pakietu `most` za pomocą `dpkg` następuje po wydaniu polecenia `dpkg -r most`.

Operacje na zainstalowanych pakietach

Narzędzie `dpkg` zapewnia dziesiątki metod sprawdzania, wyszukiwania i przeprowadzania operacji na zainstalowanych pakietach. Zawiera bazę danych informacji o pakietach zainstalowanych w systemie. W celu pobrania ogólnych informacji o pakiecie można wydać następujące polecenie:

```
$ dpkg -l most
Wybór=U=Nieznany/I=Instalacja/R=Usunięcie/P=Wyczyszczenie/H=Zatrzymanie
| Stan=N=Brak/I=Zainst./C=Skonfig./U=Rozpak./F=Nieskonfig./H=Wpół-zainst.
↪/W=Wyzw-czek/T=Wyzw-zapl
|/ Błędy?=(brak)/H=Wstrzym./R=Do przeinst. (Stan,Błędy:wielk.lit.=źle)
||/ Nazwa               Wersja              Opis
+++-===================-===================-
=====================================
ii  most                5.0.0a-1            Pager program similar to more and
↪less
```

Wydanie powyższego polecenia bez żadnych argumentów (`dpkg -l`) spowoduje wyświetlenie informacji podstawowych dotyczących stanu instalacji, nazwy, wersji i opisu *każdego* pakietu znajdującego się w systemie.

Kolejne proste zadanie to pobranie listy plików znajdujących się w pakiecie. W przypadku posiadania niezainstalowanego pakietu DEB listę plików pakietu można wyświetlić po wydaniu polecenia `dpkg --contents`, jak pokazano poniżej:

```
$ dpkg --contents /var/cache/apt/archives/most_5.0.0a-1_i386.deb
drwxr-xr-x root/root         0 2008-05-06 12:06 ./
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/bin/
-rwxr-xr-x root/root    59940 2008-05-06 12:06 ./usr/bin/most
```

```
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/share/
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/share/man/
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/share/man/man1/
-rw-r--r-- root/root      5912 2008-05-06 12:06 ./usr/share/man/
man1/most.1.gz
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/share/doc/
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/share/doc/most/
-rw-r--r-- root/root      2989 2007-09-09 12:14 ./usr/share/doc/
most/changelog.gz
-rw-r--r-- root/root      5544 2008-05-06 12:06 ./usr/share/doc/
most/copyright
-rw-r--r-- root/root      3335 2007-09-06 10:15 ./usr/share/doc/
most/README
-rw-r--r-- root/root      1386 2006-05-01 13:51 ./usr/share/doc/
most/lesskeys.rc
-rw-r--r-- root/root       492 2006-05-01 13:51 ./usr/share/doc/
most/most-fun.txt
-rw-r--r-- root/root      3086 2006-05-01 13:51 ./usr/share/doc/
most/most.rc
-rw-r--r-- root/root      2028 2008-05-06 12:06 ./usr/share/doc/most/
changelog.Debian.gz
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/lib/
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/lib/mime/
drwxr-xr-x root/root         0 2008-05-06 12:06 ./usr/lib/mime/
packages/
-rw-r--r-- root/root        94 2008-05-06 12:06 ./usr/lib/mime/
packages/most
```

Pobranie podobnych informacji, ale dotyczących zainstalowanych pakietów jest możliwe po wydaniu polecenia `dpkg -L`. Z drugiej strony, jeśli użytkownik ma plik i chciałby dowiedzieć się, do którego pakietu należy dany plik, może użyć polecenia `dpkg -S` w celu wykonania zapytania do bazy danych zawierającej informacje o wszystkich pakietach. Przykładowo:

```
dpkg -S /usr/bin/most
most: /usr/bin/most
```

Nie powinno być zaskoczeniem, że plik binarny */usr/bin/most* należy do pakietu o nazwie *most*. Ponieważ powyższe polecenie powoduje przeszukanie listy plików każdego pakietu, zakończenie operacji może zająć nieco czasu.

Operacje na repozytoriach

Najlepszym sposobem instalacji oprogramowania w „sposób Ubuntu" na pewno nie jest po prostu pobranie pakietu DEB i jego „ręczna" instalacja za pomocą narzędzia `dpkg`. Jednak narzędzie `apt-get` zawiera informacje o pakietach, które zna. Podczas gdy narzędzie `dpkg` pracuje z pakietami, narzędzie `apt-get` działa z repozytoriami pakietów zawierających informacje na temat wielu różnych pakietów, ich wersji i zależności. Dlatego też w celu zarządzania pakietami

za pomocą `apt-get` do systemu trzeba dodać nie sam pakiet, ale zawierające go repozytorium. To zadanie jest wykonywane poprzez dodanie lub edycję listy „źródeł oprogramowania". Wprawdzie biurkowa wersja Ubuntu zawiera narzędzie graficzne przeznaczone do operacji na repozytoriach, ale zadanie można bardzo łatwo wykonać ręcznie; w większości tak też będzie wykonywane.

Wspomniany już kilkakrotnie w tym rozdziale plik *sources.list* znajduje się w katalogu */etc/apt/* w każdym systemie Ubuntu i Debian. Wymieniony plik składa się z serii następujących wierszy:

```
deb http://us.archive.ubuntu.com/ubuntu/ lucid main universe
deb-src http://us.archive.ubuntu.com/ubuntu/ lucid main universe
```

Na samym początku wiersza będzie znajdował się znak # oznaczający komentarz lub słowo `deb`, lub `deb-src`. W ten sposób można rozróżnić repozytoria kodu źródłowego i pakietów binarnych. Drugi element wiersza to adres podany w postaci URI. Trzeci element to nazwa dystrybucji, którą dokładniej można określić jako wersję dystrybucji. W powyższym przykładzie wersja dystrybucji to `lucid`, co odnosi się do wydania Ubuntu 10.04 Lucid Lynx (Świetlisty Ryś). Pozostałe argumenty tworzą listę komponentów. Komponenty dostarczane w podstawowych repozytoriach Ubuntu zostaną dokładnie omówione w dalszej części rozdziału.

Proces dodawania repozytorium najlepiej zilustrować na przykładzie. Jeżeli Czytelnik chce zawsze posiadać oprogramowanie Bazaar[4] w najnowszej wersji, instalację trzeba przeprowadzić z pominięciem standardowych repozytoriów Ubuntu, ponieważ w ich przypadku oprogramowanie jest uaktualniane zgodnie z cyklami wydawania systemu. Na szczęście programiści tworzący Bazaar dostarczają własne „osobiste repozytorium pakietów" — ten temat będzie omówiony na końcu tego rozdziału. Na swojej witrynie podają wiersze, które trzeba dodać do pliku *sources.list*:

```
deb http://ppa.launchpad.net/bzr/ubuntu lucid main
deb-src http://ppa.launchpad.net/bzr/ubuntu lucid main
```

Po dodaniu powyższych wierszy i pierwszym uaktualnieniu listy repozytoriów na ekranie zostanie wyświetlony komunikat błędu informujący o braku odpowiednich kluczy kryptograficznych, za pomocą których można potwierdzić, że wskazane repozytorium naprawdę należy do programistów oprogramowania Bazaar:

```
W: Błąd GPG: http://ppa.launchpad.net lucid
Release: Następujące podpisy nie mogły zostać zweryfikowane z powodu
braku klucza publicznego: NO_PUBKEY FE8956A73C5EE1C9
```

Powyższy błąd można bardzo łatwo naprawić poprzez pobranie wskazanego klucza z zaufanego źródła, na przykład witryny internetowej dostawców PPA (ang. *Personal Package Archives*), zapisanie go w pliku (w omawianym przykładzie ten plik to */tmp/plik_klucza*), sprawdzenie poprawności klucza, a następnie jego dodanie do bazy danych kluczy za pomocą polecenia:

```
apt-key add - < /tmp/plik_klucza
OK
```

[4] Bazaar to sponsorowany przez firmę Canonical projekt GNU systemu kontroli wersji — *przyp. tłum.*

W podręczniku polecenia `apt-key` można znaleźć szczegółowe informacje na temat zarządzania kluczami repozytoriów za pomocą wymienionego polecenia.

Domyślne repozytoria Ubuntu

Większość pakietów, których użytkownik będzie potrzebował, została już przygotowana dla Ubuntu. Wynika to z faktu, że będąc kontynuatorem Debiana, Ubuntu zapewnia dostęp do większości najpopularniejszego oprogramowania w postaci pakietów umieszczonych w repozytoriach Ubuntu.

Te dziesiątki tysięcy pakietów podzielono na różne sekcje i komponenty. Użytkownik może je włączać i wyłączać przez odpowiednie ustawienie listy komponentów w pliku *sources.list*. Ponieważ ma to istotny wpływ na oprogramowanie, które będzie można zainstalować w systemie, warto poznać poszczególne komponenty i samemu zdecydować, z jakich źródeł będzie się pobierać oprogramowanie. W Ubuntu Serwer dostępne są następujące komponenty oprogramowania: *main* (oprogramowanie open source obsługiwane przez firmę Canonical), *restricted* (oprogramowanie open source obsługiwane przez społeczność), *universe* (własnościowe sterowniki dla urządzeń) oraz *multiverse* (oprogramowanie ograniczone prawami autorskimi lub problemami natury prawnej). Poniższe opisy poszczególnych komponentów powstały na bazie informacji dostępnych na witrynie internetowej Ubuntu:

- **Main**

  Podstawowe komponenty dystrybucji zawierające wolne oprogramowanie, które może być bez przeszkód rozpowszechniane i jest w pełni obsługiwane przez zespół Ubuntu. Tutaj znajdują się najpopularniejsze i najsolidniejsze aplikacje typu open source, z których pewna część jest już zainstalowana w systemie. Oprogramowanie znajdujące się w main obejmuje ręcznie wybraną listę aplikacji, które programiści Ubuntu, społeczność i użytkownicy uważają za ważne, a zespoły odpowiedzialne za bezpieczeństwo i dystrybucję Ubuntu chcą obsługiwać. Instalując oprogramowanie z main, użytkownik ma pewność, że będzie ono obsługiwane, a także uaktualniane w postaci poprawek bezpieczeństwa.

- **Restricted**

  Komponent restricted jest zarezerwowany dla oprogramowania, które jest bardzo często używane i tym samym obsługiwane przez zespół Ubuntu, nawet jeśli nie jest rozpowszechniane na w pełni otwartej licencji. Warto pamiętać, że zespół Ubuntu może nie być w stanie zapewnić pełnej obsługi tego oprogramowania na przykład z braku możliwości poprawienia kodu źródłowego. W takim przypadku Ubuntu może przekazać informacje o problemach do rzeczywistych autorów oprogramowania.

- **Universe**

  W komponencie universe można znaleźć niemal każde oprogramowanie open source i inne dostępne na mniej otwartych licencjach. Znajdujące się tutaj oprogramowanie jest budowane automatycznie na podstawie różnych publicznych źródeł. Kompilacja tego oprogramowania odbywa się z uwzględnieniem bibliotek i narzędzi z komponentu main,

a więc powinno być bez przeszkód instalowane obok oprogramowania pochodzącego z main. Warto pamiętać, że w przypadku oprogramowania z universe *nie ma żadnej gwarancji dostarczania poprawek bezpieczeństwa i zapewnienia wsparcia technicznego*.

- **Multiverse**

  Komponent multiverse zawiera oprogramowanie niebędące wolnym, co oznacza, że warunki licencji tego oprogramowania nie spełniają polityki licencji komponentu main. Sprawdzenie warunków licencji dostępnego tutaj oprogramowania jest obowiązkiem użytkownika. To oprogramowanie nie jest obsługiwane i zwykle nie są dla niego dostarczane poprawki bądź uaktualnienia. Użytkownik używa tego oprogramowania na własną odpowiedzialność.

Używanie innych repozytoriów

Jak Czytelnik mógł się przekonać wcześniej podczas dodawania repozytorium Bazaar, użytkownicy mają możliwość używania różnych pakietów pochodzących z innych repozytoriów, nie tylko z dostarczanych przez Ubuntu. Przykładowo użytkownik może chcieć instalować nowe wersje wybranych programów lub bibliotek z opracowywanych wersji Ubuntu, ale równocześnie może nie chcieć uaktualniania wszystkich pakietów do najnowszych wersji.

Półoficjalne repozytoria Ubuntu zawierające „backporty[5]" to doskonałe źródło oprogramowania. Mogą one zawierać wersje oprogramowania z opracowywanych właśnie wydań Ubuntu, które zostały przystosowane do instalacji w stabilnych wydaniach systemu. Backporty można dodać do systemu poprzez ręczną instalację pakietu DEB za pomocą narzędzia `dpkg` lub po dodaniu dodatkowych wpisów do pliku *sources.list*. Więcej informacji na ten temat można znaleźć na witrynie internetowej Ubuntu pod adresem *https://help.ubuntu.com/community/UbuntuBackports*.

Jednym z powodów, dla którego wielu użytkowników wybiera właśnie taką metodę — to znaczy ręczne pobieranie pakietu, a następnie jego instalację za pomocą `dpkg` — zamiast używania repozytorium, jest ograniczenie w sposobie działania `apt-get`. Chodzi o to, że *apt-get oraz inne narzędzia zarządzania pakietami zawsze będą instalowały najnowszą dostępną wersję każdego pakietu, który jest domyślnie dostępny*. Dlatego też po dodaniu do pliku *sources.list* repozytorium z backportami lub opracowywanymi właśnie wersjami oprogramowania podczas procesu aktualizacji zainstalowane zostaną najnowsze wersje *wszystkich* pakietów z takiego repozytorium. W przypadku małych repozytoriów (takich jak opisane wcześniej Bazaar PPA zawierające jedynie oprogramowanie Bazaar i ściśle powiązane z nim pakiety) nie stanowi to żadnego problemu. Jednak po dodaniu ogromnego repozytorium z wieloma pakietami (na przykład repozytorium z backportami bądź opracowywanymi właśnie wydaniami Ubuntu), efekt będzie inny od pożądanego (to znaczy zaktualizowane zostaną wszystkie pakiety z repozytorium), mimo że użytkownik chciał uaktualnić tylko kilka pakietów.

[5] W użytym tutaj znaczeniu *backport* to przeniesienie oprogramowania z nowszego wydania systemu do starszego — *przyp. tłum.*

Ogólnie rzecz biorąc, rozwiązaniem takiego problemu jest zastosowanie tzw. techniki „pinning" lub inaczej „apt pinning". Technika pinning oferuje niezwykle potężne możliwości, ale w swej zaawansowanej postaci może być bardzo skomplikowana. Dlatego też pełne omówienie techniki pinning wykracza poza temat niniejszej książki. Warto jednak zapoznać się z poniższym przykładem, gdzie użytkownik ma zainstalowaną wersję Karmic systemu, ale chce, by narzędzie `apt-get` preferowało pakiety wersji Lucid. W tym celu należy utworzyć plik */etc/apt/preferences.d* zawierający następujące sekcje:

```
Package: *
Pin: release a=karmic
Pin-Priority: 700

Package: *
Pin: release a=lucid
Pin-Priority: 600
```

Każdy akapit opisuje jedno wydanie, a gwiazdka umieszczona na początku oznacza zastosowanie reguł względem wszystkich pakietów. W ostatnim wierszu każdego akapitu znajduje się priorytet wskazujący względne położenie (w przedstawionym tutaj przykładzie wydanie Karmic jest bardziej preferowane niż Lucid) oraz wagę każdego z wydań. Wagę można dostosować do własnych potrzeb, a więc pakiety będą instalowane bądź nie na podstawie ustalonej wagi, oczywiście poza sytuacjami specjalnymi. Więcej informacji na temat tej techniki można znaleźć na stronach podręcznika `apt_preferences`, a także w kilku miejscach dokumentacji wiki Ubuntu i Debian.

Uaktualnienie całego systemu

Ostatnim zadaniem, które każdy administrator będzie musiał kiedyś przeprowadzić, jest pełne uaktualnienie systemu do nowszego wydania. W przypadku biurkowego wydania Ubuntu domyślnym sposobem uaktualnienia systemu do nowszego wydania jest użycie *Menedżera aktualizacji*. Jednak ten program został zaprojektowany specjalnie z myślą o środowisku graficznym. Ponieważ proces uaktualnienia można bardzo łatwo przeprowadzić także z poziomu powłoki, takie rozwiązanie prawdopodobnie będzie odpowiedniejsze dla większości serwerów.

W przeszłości aktualizacja systemu do nowszego wydania była procesem składającym się z dwóch etapów. W pierwszym administrator uaktualniał listę repozytoriów (zostało to dokładnie przedstawione już wcześniej), tak aby odwołania do starszego wydania były zastąpione odwołaniami do nowszego wydania. Przykładowo podczas uaktualniania z wydania Hardy Heron do Gutsy Gibbon w pliku *sources.list* wszystkie wystąpienia słowa *hardy* należało zastąpić słowem *gutsy*. Następnie wydanie polecenia `aptitute update` (zgodnie z przedstawionym wcześniej opisem) powodowało uaktualnienie lokalnej bazy danych zawierającej metadane. Na tym etapie wspomniana baza danych posiadała informacje o wszystkich pakietach nowej dystrybucji.

Drugi etap to wydanie polecenia `aptitude full-upgrade`, które w przeciwieństwie do omówionego wcześniej `safe-upgrade` spowodowało uaktualnienie wszystkich pakietów do ich najnowszych wersji oraz w zależności od potrzeb usunięcie bądź instalację pakietów dodatkowych. Opcja

`full-upgrade` jest mniej konserwatywna niż `safe-upgrade` i oznacza większe niebezpieczeństwo wystąpienia niepożądanych efektów ubocznych. Jednak ma możliwość uaktualnienia pakietów, których nie uaktualni opcja `safe-upgrade`. Z powodu możliwości wystąpienia niepożądanych efektów podczas uaktualniania systemu z jednego wydania do drugiego użycie opcji `full-upgrade` było zalecanym sposobem przeprowadzania aktualizacji. W chwili obecnej obie wymienione metody nie są już obsługiwane.

W bieżących wydaniach Ubuntu prawidłowym sposobem przeprowadzenia aktualizacji systemu do nowszego wydania jest użycie programu `do-release-upgrade`. Wymieniony program to po prostu skrypt automatyzujący przedstawiony powyżej proces i inteligentnie obsługujący pewne kłopotliwe sytuacje. W tej chwili to jedyny obsługiwany sposób uaktualnienia Ubuntu Serwer do nowszego wydania.

Tworzenie lustrzanej kopii systemu

Jednym z zadań często wykonywanych przez wielu administratorów systemów jest utworzenie lustrzanej kopii systemu w innym komputerze. Ponieważ całe oprogramowanie w domyślnej instalacji Ubuntu pochodzi z pakietów, to dzięki systemowi zarządzania pakietami wspomniane zadanie jest łatwe do wykonania. Za pomocą narzędzia `dpkg` można utworzyć listę wszystkich pakietów zainstalowanych w danym komputerze:

```
# dpkg --get-selections > lista_pakietów
```

Powyższe polecenie spowoduje utworzenie listy pakietów, która następnie zostanie zapisana w pliku o nazwie *lista_pakietów*. Kolejny krok to przeniesienie tego pliku do drugiego komputera i użycie jako listy pakietów do instalacji:

```
# dpkg --set-selections < lista_pakietów
```

Ostatni krok to instalacja wskazanych pakietów w systemie docelowym. W tym celu należy wydać poniższe polecenie:

```
# apt-get dselect-upgrade
```

`dselect-upgrade` odwołuje się do poprzednika `apt-get`, czyli programu `dselect`. Jego zadaniem jest po prostu uaktualnienie pakietów w systemie oraz instalacja wszelkich nowych pakietów „oznaczonych” przez `dpkg --set-selections` do uaktualnienia.

Tworzenie własnych pakietów

Potęga systemu zarządzania pakietami tkwi w możliwości śledzenia zależności między pakietami, rozwiązywania ewentualnych konfliktów między nimi, przeprowadzania automatycznego uaktualniania oraz śledzenia każdego pliku w systemie i oprogramowania, do którego dany plik

należy. Instalacja za pomocą pakietu jest znacznie łatwiejsza niż pobranie kodu i samodzielna budowa programu. Jednak system zarządzania pakietami pokazuje pełnię swoich możliwości w trakcie usuwania lub aktualizacji oprogramowania. W przypadku instalacji oprogramowania z kodu źródłowego należące do niego pliki mogą znajdować się w wielu miejscach systemu plików. Natomiast po instalacji oprogramowania z pakietu jego usunięcie jest proste i sprowadza się do wydania polecenia `apt-get remove`.

Z tego powodu wielu odpowiedzialnych administratorów systemu uznało za wygodne upewnienie się, że *całe* oprogramowanie w systemie jest instalowane z pakietów. To brzmi nieźle, ale czasami konieczne oprogramowanie — lub jego konkretna wersja — nie jest dostępne w postaci pakietu lub dla używanego wydania Ubuntu. W takim przypadku zachodzi potrzeba samodzielnego zbudowania pakietu w dowolny sposób. Pozostała część rozdziału to ogólny opis takiego procesu. Zapewnia dobry punkt wyjścia dla administratorów systemu, którzy chcą wyjść poza używanie pakietów i pragną stać się również twórcami pakietów.

Przebudowanie pakietu

Jak już wcześniej wspomniano, wielu użytkowników chce przebudowywać istniejący pakiety, co jest częścią procesu tworzenia backportów, czyli dostosowania oprogramowania dostępnego w nowszym wydaniu Ubuntu lub Debian do starszego wydania. Czasami w przypadku wprowadzenia zmian w ABI oprogramowanie nie będzie działało w innym wydaniu Ubuntu, ponieważ zostało skompilowane z użyciem bibliotek, które nie są dłużej dostępne. Najłatwiejszym sposobem rozwiązania tego problemu jest po prostu pobranie kodu źródłowego brakujących komponentów i ponowne zbudowanie pakietu z uwzględnieniem nowych wersji bibliotek. Ten proces zostanie dokładnie pokazany w tym podrozdziale.

W pierwszej kolejności potrzebny jest pakiet. Jak Czytelnik pamięta z wcześniejszej części rozdziału, pakiet kodu źródłowego składa się z pliku DSC oraz przynajmniej jednego innego pliku. Takie pakiety można pobrać z witryny *http://packages.ubuntu.com* tak samo jak zwykłe pliki. Pakiet trzeba rozpakować za pomocą polecenia `dpkg-source -x` *`nazwapliku.dsc`* lub może być automatycznie zainstalowany, gdy użyje się do tego polecenia `apt-get source` *`nazwapakietu`*.

Jeżeli użytkownik chce pobrać i skompilować pakiet z określonej dystrybucji — jak to często bywa — wówczas dystrybucję można wyraźnie podać za pomocą opcji `-t`. W tle wymieniona opcja domyślnie powoduje ustawienie bardzo wysokiej wartości PIN dla danej dystrybucji poprzez użycie wysokiego priorytetu (w rzeczywistości 990):

```
$ apt-get -t jaunty source --compile most
```

Powyższe polecenie pobierze i rozpakuje pakiet źródłowy programu `most` przeznaczony dla wydania Jaunty — oczywiście przy założeniu, że w pliku */etc/apt/sources.list* znajduje się niezbędny wiersz `deb-src`. Rozpakowany kod źródłowy będzie umieszczony w podkatalogu o nazwie pakietu i wersji. Wspomniany podkatalog znajdzie się w katalogu bieżącym. W omawianym przykładzie podkatalog ma nazwę *most-5.0.0a*, ponieważ pobrano program `most` w wersji 5.0.0a. Dodanie

opcji `--compile` do powyższego polecenia `apt-get` powoduje automatyczne zbudowanie pakietu binarnego, nawet jeśli program został utworzony w języku interpretowanym i rzeczywista kompilacja nie zostanie przeprowadzona. Jeżeli opcji kompilacji nie wybrano podczas wydawania polecenia `apt-get`, kompilację można przeprowadzić później na wiele różnych sposobów. Jednym z najprostszych jest przejście do podkatalogu z kodem źródłowym programu, a następnie wydanie polecenia `dpkg-buildpackage`, jak pokazano poniżej:

```
$ cd most-5.0.0a
$ dpkg-buildpackage -us -uc -rfakeroot
```

Powyższe polecenie powoduje utworzenie niepodpisanego pakietu (opcje `-us` i `-uc` odwołują się do niepodpisanego pliku *sources* i *changelog*); nie są wymagane uprawnienia użytkownika root (fakeroot to program pozwalający na zbudowanie pakietu bez uprawnień użytkownika root). Oczywiście pakiet może wymagać również spełnienia pewnych zależności, które nie są zainstalowane przed wydaniem polecenia w następującej postaci:

```
# apt-get -t jaunty build-dep most
```

Opcja `build-dep` polecenia `apt-get` powoduje automatyzacją procesu instalacji całego oprogramowania wymaganego do zbudowania danego pakietu. Kiedy pakiet pochodzi z repozytorium, powyższe polecenie powinno być wykonane jako pierwszy krok w trakcie pierwszej przebudowy pakietu.

Po udanym przebudowaniu danego oprogramowania katalog będzie zawierał zestaw pakietów binarnych z rozszerzeniem *.deb*, utworzonych na bazie danego pakietu kodu źródłowego. W omawianym przykładzie został utworzony pojedynczy pakiet o nazwie *most 5.0.0a-1 i386.deb*. Liczba *-1* znajdująca się po numerze wersji odwołuje się do wersji pakietu i może być zwiększana po zbudowaniu każdej kolejnej wersji pakietu. Część i386 oznacza architekturę, dla której został zbudowany dany pakiet binarny. W tym przykładzie pakiet utworzono dla komputera z procesorem Intel. W przypadku wielu użytkowników będzie to amd64, gdyż ta platforma zyskuje coraz większą popularność. Dla większości programów interpretowanych platforma będzie oznaczona jako *all*, ponieważ takie programy działają w każdej architekturze.

Nowe wersje upstream

Utworzenie nowej wersji upstream pakietu jest nieco bardziej skomplikowane niż po prostu przebudowanie istniejącego pakietu bez wprowadzania jakichkolwiek modyfikacji. Instalacja pakietu `devscripts` oferuje użytkownikowi program o nazwie `uupdate`, pomagający w tym procesie. W celu użycia programu `uupdate` trzeba w pierwszej kolejności pobrać pakiet kodu źródłowego za pomocą polecenia takiego jak `apt-get source most`. Etap kompilacji na chwilę pomijamy. Następnie trzeba pobrać nową wersję upstream w postaci archiwum tarball. Nie ma żadnego powodu do rozpakowywania wspomnianego archiwum, opcjonalnie można zmienić jego nazwę na postać `nazwa-wersja.tar.gz`. Kolejny krok to przejście do katalogu zawierającego *stary* kod źródłowy pakietu i wydanie polecenia `uupdate` wraz z argumentem w postaci nowego archiwum tarball dla upstream:

```
$ cd most-5.0.0a
$ uupdate ../most-5.0.1.tar.gz
```

Program `uupdate` zwykle prawidłowo określi numer wersji archiwum tarball zawierającego upstream, a następnie wprowadzi w starej wersji kodu źródłowego wszystkie zmiany wprowadzone w nowszej wersji. Jeżeli program `uupdate` nie będzie mógł określić numeru wersji, trzeba go podać jako drugi argument programu.

Dane wyjściowe programu `uupdate` powinny wyjaśnić cały zachodzący proces, którego zwieńczeniem będzie opis wskazujący na położenie nowego, zmodyfikowanego źródła. W omawianym przykładzie zmiana na *../most-5.0.1* spowodowała przejście do nowego „uaktualnionego" katalogu pakietu. Dobrym rozwiązaniem jest sprawdzenie wszystkiego i upewnienie się o prawidłowości przeprowadzonego procesu. Szczególnie warto sprawdzić podkatalog *debian/* i przyjrzeć się znajdującym się tam plikom *control* i *changelog*. Drugi z wymienionych zostanie uaktualniony automatycznie, ale prawdopodobnie będzie wymagał drobnych korekt. Akapit na początku informuje o nowym wydaniu i będzie zawierał informacje o zmianach wprowadzonych w pliku. Gdy użytkownik jest zadowolony z uzyskanych efektów, pozostaje zbudowanie pakietu za pomocą polecenia `dpkg-buildpackage` w sposób omówiony we wcześniejszej części rozdziału.

Tworzenie pakietu zupełnie od początku

Tworzenie pakietu zupełnie od początku jest znacznie bardziej skomplikowane i wymaga pewnej wiedzy dotyczącej wewnętrznej budowy pakietów Debiana. To oczywiście wykracza poza zakres niniejszej książki. Warto jednak w tym miejscu wspomnieć o możliwości niemal łatwego utworzenia pakietów za pomocą programu `dh_make`, który jest wywoływany z nierozpakowanego kodu źródłowego tarball z wersji upstream przygotowanej przez pierwotnego twórcę oprogramowania. W przypadku wielu prostych pakietów program `dh_make` wykonuje większość ciężkiej pracy związanej z utworzeniem działającego pakietu.

Więcej informacji na temat tworzenia pakietów dla Ubuntu oraz dokładne omówienie procesu tworzenia pakietu zupełnie od początku można znaleźć na stronie *https://wiki.ubuntu.com/PackagingGuide*.

Warto w tym miejscu wspomnieć o jednej kwestii związanej z dokumentacją Ubuntu. Dokument znajdujący się na wymienionej wyżej stronie koncentruje się na tworzeniu pakietów przeznaczonych do umieszczania w repozytoriach Ubuntu. Jeżeli użytkownik tworzy pakiety przeznaczone do instalacji jedynie we własnym systemie, potencjalna szkodliwość pakietu jest znacznie mniejsza, a wiele wskazówek w wymienionym dokumencie można tak właśnie potraktować — zwłaszcza w pierwszej wersji pakietu. To jest różnica między pakietami działającymi i spełniającymi politykę zgodności.

Kiedy użytkownik zamierza tworzyć pakiety, dzielić się nimi z innymi i wreszcie umieszczać je w repozytoriach Ubuntu, wówczas *bardzo* dobrym pomysłem będzie dokładne zastosowanie się do wskazówek zamieszczonych w wymienionym dokumencie i używanie programów takich

jak `lintian`, które sprawdzą pakiet pod kątem wielu najczęściej występujących błędów, co jest użyteczne w wielu sytuacjach. Jeżeli pakiet ma jedynie działać, prawdopodobnie wystarczające będzie zapoznanie się z dokumentacją programu `dh_make` i jego użycie.

Hosting własnych pakietów

Ostatnim etapem w procesie tworzenia własnych pakietów jest ich hosting w miejscu, które inni użytkownicy mogą po prostu „dodać do swojego pliku *sources.list*" w sposób przedstawiony we wcześniejszej części rozdziału. Istnieje tutaj kilka rozwiązań. Najłatwiejsze i najczęściej spotykane w świecie Ubuntu polega na użyciu Launchpad — czyli infrastruktury zbudowanej przez firmę Canonical i intensywnie wykorzystywanej podczas prac nad kolejnymi wydaniami Ubuntu — i umieszczeniu pakietu w tak zwanym osobistym archiwum pakietów (ang. *Personal Package Archive*, PPA).

Dzięki PPA programista może po prostu przekazać pakiet kodu źródłowego do Launchpad, a pakiet następnie zostanie zbudowany dla wielu różnych architektur i umieszczony w repozytorium PPA. PPA działa dokładnie w taki sam sposób, w jaki są prowadzone prace nad Ubuntu. Z tego powodu to cenne doświadczenie pozwalające przekonać się, jak wygląda cały proces, gdy użytkownik zdecyduje się na przekazanie pakietu do Ubuntu i zaangażowanie w prace po tamtej stronie. We wcześniejszej części rozdziału pokazano, że podczas dołączania repozytorium z pakietami Bazaar do listy pakietów nastąpiło użycie Bazaar PPA. Więcej informacji na temat PPA można znaleźć na stronach *https://help.launchpad.net/Packaging/PPA* i *https://launchpad.net/ubuntu/+ppas*.

Alternatywnym rozwiązaniem jest hosting pakietów we własnym repozytorium utworzonym w ramach posiadanego serwera. Do tego celu można wykorzystać wiele różnych narzędzi. Wprawdzie klasycznym narzędziem jest tutaj pakiet o nazwie `apt-ftparchive`, ale prawdopodobnie lepszym rozwiązaniem będzie użycie nowszego projektu o nazwie `reprepro`. Dobrym punktem wyjścia będzie instalacja pakietu o podanej nazwie i zapoznanie się z dostarczoną dokumentacją.

Ubuntu to dystrybucja systemu operacyjnego Linux, która podbiła serca użytkowników domowych. Możliwość zapoznania się z systemem bez ingerencji w dotychczas używany system, banalna instalacja, wyjątkowo atrakcyjny interfejs użytkownika oraz małe wymagania to tylko niektóre atuty tego systemu. Autorzy postanowili pójść za ciosem i przygotowali wersję dla serwerów. Co sprawia, że jest wyjątkowa? Co ją odróżnia od innych dystrybucji? Czy warto ją zainstalować?

Na te pytania znajdziesz odpowiedź w jedynym oficjalnym podręczniku poświęconym dystrybucji Ubuntu w wersji serwerowej. Autorzy prezentują sposób instalacji systemu oraz podstawowe operacje administracyjne. Po szybkim i pełnym konkretnych informacji wstępie zajmiesz się zarządzaniem zainstalowanym oprogramowaniem, usługami dostarczanymi przez Ubuntu, zapewnieniem bezpieczeństwa w systemie oraz tworzeniem kopii zapasowej. Ponadto dowiesz się, jak monitorować pracę serwera, wirtualizować go oraz zwiększać odporność na awarie. Punkt po punkcie autorzy rozwieją wszystkie Twoje wątpliwości związane z systemem Ubuntu w tej wersji. Jeżeli nie jesteś pewien, czy to coś dla Ciebie, chcesz poznać ten system lub jesteś pasjonatem Linuksa — musisz mieć tę książkę!

- Historia projektu Ubuntu
- Instalacja systemu
- Proces uruchamiania Ubuntu
- System plików
- Administrowanie systemem
- Zarządzanie pakietami oprogramowania
- Automatyzacja procesu instalacji
- Konfiguracja serwera DNS, WWW
- Uruchomienie serwera poczty elektronicznej
- Konfiguracja serwera baz danych
- Umożliwienie dostępu zdalnego do serwera — OpenSSH
- Bezpieczeństwo w Ubuntu
- Wykrywanie włamań do systemu
- Monitorowanie pracy serwera
- Wirtualizacja — KVM, VMware Server
- Konfiguracja macierzy RAID
- Rozwiązywanie typowych problemów
- Tryb ratunkowy i odzyskiwanie
- Przydatne zasoby Ubuntu Serwer

Cena 79,00 zł
ISBN 978-83-246-3300-5
9 788324 633005