

APT HOWTO

Gustavo Noronha Silva <kov@debian.org>
polskie tłumaczenie: Paweł Tęcza <ptecza@debianusers.pl>

1.8.5 - lipiec 2003

Streszczenie

Ten dokument ma na celu dobrze wyjaśnić użytkownikowi zasadę działania narzędzia do zarządzania pakietami Debiana jakim jest APT. Jego celem jest uczynienie łatwiejszym życia nowym użytkownikom Debiana oraz pomoc tym jego użytkownikom, którzy chcieliby lepiej rozumieć administrowanie systemem pakietów. Został on stworzony w ramach projektu Debian, aby pomóc w polepszeniu wsparcia dla użytkowników tej dystrybucji.

Prawa autorskie

Copyright © 2001, 2002, 2003, 2004 Gustavo Noronha Silva

This manual is free software; you may redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details.

A copy of the GNU General Public License is available as `/usr/share/common-licenses/GPL` in the Debian GNU/Linux distribution or on the World Wide Web at the GNU General Public Licence. You can also obtain it by writing to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Spis treści

1	Wprowadzenie	1
2	Podstawowa konfiguracja	3
2.1	Plik <code>/etc/apt/sources.list</code>	3
2.2	Jak używać APT-a lokalnie	4
2.3	Podjęmowanie decyzji, który serwer lustrzany najlepiej umieścić w pliku <code>sources.list</code> : <code>netselect</code> , <code>netselect-apt</code>	5
2.4	Dodawanie CD-ROM-u do pliku <code>sources.list</code>	6
3	Zarządzanie pakietami	9
3.1	Aktualizacja listy dostępnych pakietów	9
3.2	Instalacja pakietów	9
3.3	Usuwanie pakietów	11
3.4	Aktualizacja pakietów	12
3.5	Aktualizacja do nowego wydania	13
3.6	Usuwanie niepotrzebnych plików z pakietami: <code>apt-get clean</code> i <code>autoclean</code>	15
3.7	Używanie APT-a razem z programem <code>dselect</code>	16
3.8	Jak utrzymywać “wymieszany” system	17
3.9	Jak uaktualniać pakiety do nowszej wersji z określonej wersji Debiana	18
3.10	Jak zachować określone wersje zainstalowanych pakietów	19
4	Bardzo użyteczni pomocnicy	21
4.1	Jak zainstalować lokalnie skompilowane pakiety: pakiet <code>equivs</code>	21
4.2	Usuwanie nieużywanych plików lokalizacyjnych: pakiet <code>localpurge</code>	23
4.3	Jak się dowiedzieć, które pakiety mogą być uaktualnione do nowszej wersji	24

5 Pobieranie informacji o pakietach	25
5.1 Odnajdywanie nazw pakietów	25
5.2 Używanie programu dpkg do dowiadywania się nazwy pakietów	28
5.3 Jak zainstalować pakiety “na żądanie”	28
5.4 Jak sprawdzić, do którego pakietu należy plik	29
5.5 Jak otrzymywać informacje o zmianach w pakietach	30
6 Praca z pakietami źródłowymi	31
6.1 Pobieranie pakietów źródłowych	31
6.2 Pakiety potrzebne do kompilowania pakietów źródłowych	32
7 Jak radzić sobie z błędami	35
7.1 Najczęstsze błędy	35
7.2 Gdzie mogę znaleźć pomoc	36
8 Które dystrybucje wspierają APT-a?	37
9 Podziękowania	39
10 Nowe wersje tego podręcznika	41

Rozdział 1

Wprowadzenie

Na początku był plik `.tar.gz`. Użytkownicy musieli kompilować każdy program, który chcieli używać w swoich systemach GNU/Linux. Kiedy stworzono Debiana, uznano za niezbędne włączenie do systemu metody zarządzania pakietami zainstalowanymi na komputerze pracującym pod jego kontrolą. Ten system zarządzania pakietami nazwano `dpkg`. Tak więc słynne pojęcie ‘pakietu’ weszło do systemów GNU/Linux zanim firma Red Hat zdecydowała się stworzyć swój własny system ‘rpm’.

Szybko uwagę twórców systemu GNU/Linux zajął nowy dylemat. Potrzebowali oni bardzo szybkiego, praktycznego i sprawnego sposobu instalowania pakietów, który automatycznie zarządzałby zależnościami między pakietami i troszczył się o ich pliki konfiguracyjne podczas aktualizacji. I znów to właśnie Debian wprowadził ten sposób i tak oto narodził się APT (Advanced Packaging Tool), który został przystosowany do współpracy z pakietami rpm przez firmę Conectiva, a następnie wykorzystany także przez inne dystrybucje.

Ten podręcznik nie opisuje `apt-rpm`, jak nazwano wersję APT-a zmodyfikowaną przez firmę Conectiva, ale “łaty” do tego dokumentu, które by to robiły są mile widziane.

Informacje zawarte w tym podręczniku dotyczą następnego stabilnego wydania Debiana, które nosi nazwę kodową `Sarge`.

Rozdział 2

Podstawowa konfiguracja

2.1 Plik `/etc/apt/sources.list`

Do części swoich operacji APT używa pliku zawierającego listę 'źródeł', z których mogą być pobierane pakiety. Tym plikiem jest `/etc/apt/sources.list`.

Wpisy w tym pliku mają następujący format:

```
deb http://witryna.http.org/debian dystrybucja sekcja1 sekcja2 sekcja3
deb-src http://witryna.http.org/debian dystrybucja sekcja1 sekcja2 sekcja3
```

Oczywiście powyższe wpisy są fikcyjne i nie powinny być używane. Pierwszym słowem w każdej linii jest `deb` lub `deb-src`, które wskazuje typ archiwum. Mówi ono czy archiwum zawiera pakiety binarne (`deb`), czyli skompilowane już pakiety, których zazwyczaj używamy czy pakiety źródłowe (`deb-src`), którymi są oryginalne źródła programów wraz z plikiem zawierającym opis (`.dsc`) oraz plikiem przechowującym zmiany potrzebne do 'zdebianizowania' programu (`.diff.gz`).

Zwykle plik `sources.list` zawiera domyślnie następujące wpisy:

```
# See sources.list(5) for more information, especially
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib no
```

Linie te są potrzebne do podstawowej instalacji Debiana. Pierwsza linia ze słowem `deb` wskazuje na oficjalne archiwum, druga na archiwum non-US, a trzecia na archiwum z uaktualnieniami pakietów związanych z bezpieczeństwem Debiana.

Dwie ostatnie linie są wykomentowane (za pomocą znaku `#` na początku linii), więc `apt-get` po prostu je zignoruje. To są linie ze słowem `deb-src`, które wskazują na pakiety źródłowe Debiana. Jeśli często pobierasz źródła programów do testowania ich lub rekompilacji, to odkomentuj te linie.

Plik `/etc/apt/sources.list` może zawierać kilka typów linii. APT wie jak sobie radzić z archiwami typu `http`, `ftp`, `file` (lokalne pliki, np. katalog zawierający zamontowany system plików ISO9660) i `ssh`.

Nie zapomnij wydać polecenia `apt-get update` po zmodyfikowaniu zawartości pliku `/etc/apt/sources.list`. Jest to konieczne, aby APT pobrał listy dostępnych pakietów ze źródeł wymienionych w tym pliku.

2.2 Jak używać APT-a lokalnie

Czasami masz dużo pakietów `.deb`, do których instalacji chciałbyś/chciałabyś użyć APT-a, aby zajął się zależnościami.

Aby to zrobić stwórz najpierw katalog i umieść w nim pliki `.deb`, które chcesz zindeksować, np.:

```
# mkdir /root/debs
```

Możesz zmodyfikować zapisywane w plikach kontrolnych ustawienia pakietów znajdujących się w Twoim repozytorium przy pomocy pliku unieważniającego. Wewnątrz tego pliku możesz zdefiniować jakieś opcje unieważniające inne, które były dostarczone wraz z pakietem. Odbywa się to w następujący sposób:

```
pakiet priorytet sekcja
```

gdzie `'pakiet'` jest nazwą pakietu, `'priorytet'` może przybierać wartości `'low'`, `'medium'` lub `'high'`, a `'sekcja'` jest nazwą sekcji, do której należy pakiet. Możesz nazwać ten plik jest tylko chcesz, będziesz musiał później podać jego nazwę jako argument programu `dpkg-scanpackages`. Jeśli nie masz ochoty na tworzenie pliku unieważniającego, to zamiast niego możesz użyć urządzenia `/dev/null` przy uruchamianiu programu `dpkg-scanpackages`.

Wciąż w katalogu `/root` wydaj polecenie:

```
# dpkg-scanpackages debs plik | gzip > debs/Packages.gz
```

W powyższej linii plik to plik unieważniający, zaś komenda tworzy plik `Packages.gz`, zawierający różne informacje na temat pakietów, które są używane przez APT-a. Aby móc używać pakietów, w pliku `/etc/apt/sources.list` dodaj linię:

```
deb file:/root debs/
```

Teraz już możesz używać poleceń APT-a tak, jak zwykle. Możesz także stworzyć repozytorium źródeł. W tym celu musisz wykonać taką samą procedurę jak wcześniej. Pamiętaj jednak, że będziesz potrzebować w katalogu plików `.orig.tar.gz`, `.dsc` i `.diff.gz`, oraz że należy użyć nazwy `Sources.gz` zamiast `Packages.gz`. Również program do jego tworzenia jest inny. Nazywa się on `dpkg-scansources`. Linia komend powinna wyglądać następująco:

```
dpkg-scansources debs | gzip > debs/Sources.gz
```

Zauważ, że program `dpkg-scansources` nie potrzebuje pliku unieważniającego. Do pliku `/etc/apt/sources.list` powinniśmy wpisać:

```
deb-src file:/root debs/
```

2.3 Podejmowanie decyzji, który serwer lustrzany najlepiej umieścić w pliku `sources.list`: `netselect`, `netselect-apt`

Często zadawanym pytaniem, zwłaszcza przez nowych użytkowników Debiana, jest: “który serwer lustrzany z pakietami powinienem umieścić w pliku `sources.list`?”. Jest wiele sposobów pozwalających wybrać serwer lustrzany. Eksperci zapewne używają skryptów mierzących obciążenie różnych serwerów. Istnieje jednak program, który zrobi to dla nas, a który nazywa się `netselect`.

Aby zainstalować program `netselect`, jak zwykle wydajemy polecenie:

```
# apt-get install netselect
```

Uruchomienie programu `netselect` bez podania parametrów spowoduje wyświetlenie pomocy ekranowej. Gdy jednak uruchomimy go podając jako argument listę, oddzielonych spacją, stacji (serwerów lustrzanych), to otrzymamy w wyniku jedną z podanych maszyn. Wynik ten zależy od przybliżonego czasu powrotu wysyłanych do stacji pakietów IP (tzw. ping) oraz od ilości stacji pośrednich (przez które te pakiety “przechodzą” zanim dotrą do celu) i jest odwrotnie proporcjonalny do przybliżonej prędkości pobierania (zatem, im mniejszy, tym lepiej). Zwrócona w wyniku stacja ma najmniejszy wynik (pełna lista wyników będzie widoczna po dodaniu opcji `-vv`). Spójrz na poniższy przykład:

```
# netselect ftp.debian.org http.us.debian.org ftp.at.debian.org download.unes  
365 ftp.debian.org.br  
#
```

Oznacza to, że spośród serwerów lustrzanych podanych jako parametry programu netselect, najlepszym okazał się `ftp.debian.org.br`, który uzyskał wynik 365 (Uwaga! Ten test został wykonany na moim komputerze a trasa, którą pokonują pakiety IP zależy od miejsca, do którego podłączony jest nasz komputer. Uzyskana wartość niekoniecznie będzie prawdziwą prędkością na innych komputerach).

Teraz umieść najszybszy serwer lustrzany znaleziony przez netselect w pliku `/etc/apt/sources.list` (patrz na sekcję 'Plik `/etc/apt/sources.list`' na 3 stronie) i zastosuj porady z sekcji 'Zarządzanie pakietami' na 9 stronie.

Uwaga: listę serwerów lustrzanych można zawsze znaleźć w pliku http://www.debian.org/mirror/mirrors_full.

Poczynając od wersji 0.3, pakiet netselect zawiera skrypt `netselect-apt`, który automatyzuje opisany powyżej proces. Po prostu jako parametr skryptu podaj dystrybucję (domyślnie jest nią dystrybucja stabilna), a wygeneruje on plik `sources.list` z najlepszym serwerem lustrzanym zawierającym pakiety z sekcji main i non-US i zapisze go w bieżącym katalogu. Poniższy przykład generuje plik `sources.list` dla stabilnej dystrybucji.

```
# ls sources.list
ls: sources.list: File or directory not found
# netselect-apt stable
(...)
# ls -l sources.list
sources.list
#
```

Pamiętaj: plik `sources.list` zostanie wygenerowany w bieżącym katalogu i musi być przeniesiony do katalogu `/etc/apt`.

Następnie przejdź do porad z sekcji 'Zarządzanie pakietami' na 9 stronie.

2.4 Dodawanie CD-ROM-u do pliku `sources.list`

Jeśli raczej używasz napędu CD-ROM do instalowania pakietów lub automatycznego uaktualniania Twojego systemu przy pomocy APT-a, to możesz go umieścić w pliku `sources.list`. Aby to zrobić, możesz użyć programu `apt-cdrom` w poniższy sposób:

```
# apt-cdrom add
```

z umieszczoną w napędzie CD-ROM płytą z pakietami Debiana. Polecenie te zamontuje napęd i poszuka na płycie informacji na temat pakietów. Jeśli konfiguracja Twojego CD-ROM-u jest trochę nietypowa, możesz także użyć następujących opcji:

```
-h          - Pomoc ekranowa do programu
-d katalog - Katalog, w którym montowany jest CD-ROM
-r          - Zmiana nazwy rozpoznanego CD-ROM-u
-m          - Bez montowania
-f          - Tryb szybki, bez sprawdzania plików z pakietami
-a          - Tryb dokładnego przeszukiwania
```

Na przykład:

```
# apt-cdrom -d /home/ala/mojcdrom add
```

Możesz także zidentyfikować swój CD-ROM, bez dodawania go do listy:

```
# apt-cdrom ident
```

Zwróć uwagę, że ten program będzie działał tylko wtedy, gdy Twój CD-ROM jest właściwie skonfigurowany w systemowym pliku `/etc/fstab`.

Rozdział 3

Zarządzanie pakietami

3.1 Aktualizacja listy dostępnych pakietów

System zarządzania pakietami używa własnej bazy danych, w której przechowuje informacje o tym, które pakiety są zainstalowane, które nie i które są dostępne do instalacji. Program `apt-get` używa właśnie tej bazy danych, aby dowiedzieć się jak zainstalować pakiety na życzenie użytkownika i jakie dodatkowe pakiety są potrzebne, aby te wybrane przez użytkownika działały prawidłowo.

Aby uaktualnić tę bazę danych należy użyć komendy `apt-get update`. Przeszuka ona archiwa umieszczone w pliku `/etc/apt/sources.list`; zobacz sekcję 'Plik `/etc/apt/sources.list`' na 3 stronie, aby uzyskać więcej informacji na temat tego pliku.

Dobrym nawykiem jest uruchamiać tę komendę dosyć regularnie, aby stale informować swój system o możliwych uaktualnieniach pakietów, zwłaszcza tych związanych z jego bezpieczeństwem.

3.2 Instalacja pakietów

W końcu proces, na który wszyscy czekaliśmy! Gdy już mamy plik `sources.list` z listą źródeł pakietów i wiemy jakie pakiety chcemy uaktualnić, to wszystko co musimy zrobić, to uruchomić program `apt-get`, który pobierze pakiety, które zamierzamy zainstalować. Dla przykładu możesz go wywołać następująco:

```
# apt-get install xchat
```

APT poszuka w swojej bazie danych najnowszej wersji pakietu i pobierze go z odpowiedniego archiwum, które podano w pliku `sources.list`. Gdyby okazało się, że pakiet ten zależy od innego, tak jak w powyższym przykładzie, to APT sprawdzi zależności i zainstaluje potrzebne pakiety. Spójrz na ten przykład:

```
# apt-get install nautilus
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1 not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

Pakiet `nautilus` zależy od kilku wymienionych bibliotek współdzielonych, dlatego APT pobierze je z archiwum. Gdybyś podał nazwy tych bibliotek jako parametr wywołania programu `apt-get`, to APT nie zapytałby Cię czy chcesz kontynuować, gdyż automatycznie założyłby, że chciałeś je zainstalować.

Oznacza to, że APT tylko wtedy pyta Cię o potwierdzenie, gdy z powodu zależności między pakietami konieczne jest zainstalowanie także innych pakietów niż te podane przez Ciebie w linii komend.

Użyteczne mogą być następujące opcje programu `apt-get`:

```
-h Pomoc ekranowa do programu
-d Tylko pobranie pakietu, BEZ instalacji lub rozpakowania archiwum
-f Próba kontynuowania, nawet jeśli sprawdzenie integralności kończy się nie
-s Bez akcji. Wykonuje symulację polecenia
-y Zakłada odpowiedź 'Tak' na wszystkie zapytania i nie pyta o potwierdzenie
-u Wyświetla również listę uaktualnionych pakietów
```

W jednej linii można podać wiele pakietów do instalacji. Pliki pobrane z sieci są umieszczane do późniejszej instalacji w katalogu `/var/cache/apt/archives`.

Możesz także w tej samej linii wymienić pakiety, które chcesz usunąć. Po prostu umieść znak `'-'` bezpośrednio za nazwą pakietu do usunięcia, tak jak pokazano poniżej:

```
# apt-get install nautilus gnome-panel-
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1 not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Aby dowiedzieć się więcej na temat usuwania pakietów, zobacz sekcję ‘Usuwanie pakietów’ na tej samej stronie.

Gdybyś w jakiś sposób uszkodził zainstalowany pakiet, albo po prostu chciał ponownie zainstalować jego najnowszą dostępną wersję, to możesz użyć opcji `--reinstall`, jak pokazano poniżej:

```
# apt-get --reinstall install gdm
Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 1 not
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

3.3 Usuwanie pakietów

Jeśli nie chcesz już dłużej używać jakiegoś pakietu, możesz go usunąć ze swojego systemu za pomocą APT-a. Aby to zrobić, po prostu wpisz: `apt-get remove pakiet`. Na przykład:

```
# apt-get remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
0 packages upgraded, 0 newly installed, 4 to remove and 1 not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Jak widać na powyższym przykładzie, APT troszczy się także o usunięcie pakietów, które zależą od pakietu, który chcesz odinstalować. Za pomocą APT-a nie można usunąć pakietu bez usunięcia tych pakietów, które zależą od niego.

Uruchamiając program `apt-get` w taki sposób jaki pokazano powyżej spowodujemy usunięcie pakietów, ale jeśli posiadały one pliki konfiguracyjne, to pozostaną one nadal w systemie. Aby całkowicie usunąć pakiet uruchom `apt-get` w następujący sposób:

```
# apt-get --purge remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
0 packages upgraded, 0 newly installed, 4 to remove and 1 not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Zwróć uwagę na znak '*' na końcu nazwy pakietów. Sygnalizuje on, że pliki konfiguracyjne każdego z pakietów także zostały usunięte.

Analogicznie jak dla metody `install`, możesz z opcją `remove` użyć symbolu, który odwróci jej znaczenie. W przypadku usuwania pakietów, jeśli za nazwą pakietu umieścisz znak '+', to zostanie on zainstalowany, a nie usunięty.

```
# apt-get --purge remove gnome-panel nautilus+
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1 not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Zauważ, że program `apt-get` wyświetla dodatkowe pakiety, które zostaną zainstalowane (tzn. te pakiety, których instalacja jest potrzebna do poprawnego działania pakietów, które chce zainstalować użytkownik), pakiety które zostaną usunięte i te, które zostaną zainstalowane (ponownie włączając dodatkowe pakiety).

3.4 Aktualizacja pakietów

Możliwość aktualizacji pakietów to wielkie osiągnięcie systemu APT. Odbywa się to za pomocą pojedynczej komendy: `apt-get upgrade`. Możesz jej użyć zarówno do aktualizacji pakietów z tej samej dystrybucji, jak i by zaktualizować całą dystrybucję, chociaż do tego celu jest raczej preferowana komenda `apt-get dist-upgrade`; zobacz sekcję 'Aktualizacja do nowego wydania' na następnej stronie, jeśli chcesz się więcej dowiedzieć na ten temat.

Warto uruchomić tą komendę z opcją `-u`. Spowoduje ona, że APT wyświetli kompletną listę pakietów, które zostaną zaktualizowane. Bez niej nie będziesz wiedział jakie pakiety aktualizujesz. APT pobierze najnowsze wersje każdego pakietu i zainstaluje je w odpowiedniej kolejności. Bardzo ważne jest uruchamianie `apt-get update` zawsze przed wydaniem polecenia `apt-get upgrade`. Zobacz sekcję 'Aktualizacja listy dostępnych pakietów' na 9 stronie, aby dowiedzieć się więcej szczegółów na jego temat. Spójrz także na poniższy przykład:

```
# apt-get -u upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
```

```
cpp gcc lilo
The following packages will be upgraded
  adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp indent
  ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0 libesd0-dev
  libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
  libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procs psmisc
29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 5055B/5055kB of archives. After unpacking 1161kB will be used.
Do you want to continue? [Y/n]
```

Proces uaktualniania pakietów jest bardzo prosty. Zauważ, że w pierwszych kilku liniach APT informuje, które pakiety zostaną pozostawione nietknięte. Oznacza to, że istnieją nowsze wersje tych pakietów, ale z jakichś powodów nie zostaną zainstalowane. Przyczynami takiego stanu mogą być zepsute zależności (pakiet, od którego zależy nie ma dostępnej wersji do pobrania) lub nowe zależności (pakiet w ostatniej wersji zależy od nowego pakietu).

W pierwszym przypadku nie ma prostego rozwiązania. W drugim wystarczy uruchomić komendę `apt-get install`, aby zainstalować specyficzny pakiet wraz z zależnościami. Jeszcze prostszym rozwiązaniem jest użycie komendy `dist-upgrade`. Zobacz sekcję ‘Aktualizacja do nowego wydania’ na bieżącej stronie, aby dowiedzieć się więcej na ten temat.

3.5 Aktualizacja do nowego wydania

Ta zdolność APT-a pozwala Ci uaktualnić cały system Debian za jednym razem, zarówno poprzez sieć Internet jak i z nowej płyty CD (nabytej lub pobranej w postaci obrazu ISO).

Jest ona również używana, gdy ulegają zmianie relacje pomiędzy zainstalowanymi pakietami. Dzięki poleceniu `apt-get upgrade`, pakiety te pozostaną w spokoju (pozostawione nietknięte).

Dla przykładu, przypuśćmy że używamy wydania nr 0 stabilnej wersji Debiana i kupujemy płytę CD z wydaniem nr 3. Możemy użyć APT-a do aktualizacji systemu z tej nowej płyty CD. W tym celu najpierw użyj programu `apt-cdrom` (więcej szczegółów w sekcji ‘Dodawanie CD-ROM-u do pliku `sources.list`’ na 6 stronie), aby dodać tę płytę do pliku `/etc/apt/sources.list`, a następnie uruchom program `apt-get dist-upgrade`.

Jest ważną rzeczą zauważenie, że APT zawsze szuka najnowszych wersji pakietów. Dlatego, jeśli Twój plik `/etc/apt/sources.list` zawiera listę archiwów z nowszymi wersjami pakietów niż na płycie CD, to APT nie pobierze ich z płyty, lecz właśnie stamtąd.

W przykładzie pokazanym w sekcji ‘Aktualizacja pakietów’ na poprzedniej stronie widzieliśmy, że część pakietów została pozostawiona nietknięta. Teraz możemy rozwiązać ten problem właśnie za pomocą metody `dist-upgrade`:

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
```

```
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common ftp gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procs psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
Do you want to continue? [Y/n]
```

Zauważ teraz, że pakiety zostaną zaktualizowane i że zostaną też zainstalowane nowe pakiety (nowe zależności tych pakietów). Zauważ również, iż lilo wciąż pozostaje nietknięte. Przyczyną tego problemu jest zapewne coś więcej niż nowa zależność. Odnajdziemy ją uruchamiając:

```
# apt-get -u install lilo
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be used.
Do you want to continue? [Y/n]
```

Jak widać powyżej, lilo ma nowy konflikt z pakietem `debconf-tiny`, który uniemożliwia jego instalację (lub uaktualnienie) bez usunięcia pakietu `debconf-tiny`.

Aby przekonać się, co było przyczyną pozostawienia pakietu nietkniętym lub usunięcia pakietu możemy użyć poniższego polecenia:

```
# apt-get -o Debug::pkgProblemResolver=yes dist-upgrade
```

```
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
  Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
```

W ten sposób można łatwo sprawdzić, że pakiet `python1.5-dev` nie może zostać zainstalowany z powodu niespełnienia zależności przez pakiet `python1.5`.

3.6 Usuwanie niepotrzebnych plików z pakietami: `apt-get clean` i `autoclean`

W czasie instalowania pakietu APT pobiera niezbędne pliki z serwera wskazanego w pliku `/etc/apt/sources.list`, umieszcza je w lokalnym repozytorium (`/var/cache/apt/archives/`), a następnie instaluje w sposób opisany w sekcji ‘Instalacja pakietów’ na 9 stronie.

Z czasem lokalne repozytorium może się rozrosnąć i zająć dużą ilość przestrzeni dysku. Na szczęście APT jest wyposażony w narzędzia, które umożliwiają zarządzanie lokalnym repozytorium. Należą do nich metody `clean` i `autoclean` programu `apt-get`.

`apt-get clean` usuwa wszystkie pliki z wyjątkiem plików “blokujących” (ang. lock files) z katalogów `/var/cache/apt/archives/` i `/var/cache/apt/archives/partial/`. Jednakże, jeśli będziesz chciał ponownie zainstalować pakiet, to APT jeszcze raz pobierze niezbędne pliki.

`apt-get autoclean` usuwa tylko te pliki, które nie będą mogły być później pobrane z Sieci.

Poniższy przykład pokazuje w jaki sposób działa metoda `autoclean`:

```
# ls /var/cache/apt/archives/logrotate* /var/cache/apt/archives/gpm*
logrotate_3.5.9-7_i386.deb
```

```
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

W katalogu `/var/cache/apt/archives` znajdują się dwa pliki z pakietem `logrotate` i jeden z pakietem `gpm`.

```
# apt-show-versions -p logrotate
logrotate/stable uptodate 3.5.9-8
# apt-show-versions -p gpm
gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

Program `apt-show-versions` informuje, że plik `logrotate_3.5.9-8_i386.deb` zawiera najnowszą wersję pakietu `logrotate`, a zatem plik `logrotate_3.5.9-7_i386.deb` jest już bezużyteczny. Także plik `gpm_1.19.6-11_i386.deb` nie będzie dłużej potrzebny, gdyż dostępna jest już nowsza wersja pakietu `gpm`.

```
# apt-get autoclean
Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

Jak widać powyżej, `apt-get autoclean` usuwa tylko stare pliki. Więcej informacji na temat programu `apt-show-versions` można znaleźć w sekcji ‘Jak uaktualniać pakiety do nowszej wersji z określonej wersji Debiana’ na 18 stronie.

3.7 Używanie APT-a razem z programem `dselect`

`dselect` jest programem, który pomaga użytkownikom Debiana wybrać pakiety, które chcą zainstalować. Jest uważany za nieco skomplikowany i dość nieciekawym, ale jeśli nabierzesz wprawy, to będziesz się swobodnie posługiwał jego konsolowym interfejsem.

Jedną z cech programu `dselect` jest to, że informuje o tym, które jeszcze pakiety Debiana są “zalecane” i “proponowane” do zainstalowania. Aby użyć tego programu, jako root uruchom `'dselect'`. Wybierz `'apt'` jako metodę dostępu. Nie jest to w rzeczywistości niezbędne, ale jeśli nie używasz CD-ROM-u i chcesz pobierać pakiety z Internetu, to jest to najlepszy sposób używania `dselecta`.

Aby jeszcze lepiej zrozumieć w jaki sposób używać `dselecta`, przeczytaj dokumentację do niego, którą można znaleźć na stronie <http://www.debian.org/doc/ddp>.

Po wybraniu pakietów przy pomocy `dselecta`, użyj:

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

tak, jak to pokazano na przykładzie poniżej:

```
# apt-get -u dselect-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  lbxproxy
The following NEW packages will be installed:
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0 not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

Porównaj to z tym, co widzisz, gdy uruchamiasz `apt-get dist-upgrade` w tym samym systemie.

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

Zwróć uwagę, iż wiele z powyższych pakietów będzie instalowana, ponieważ są “zalecane” lub “proponowane” przez inne. Pozostałe pakiety będą instalowane lub usunięte (np. w przypadku pakietu `lbxproxy`) zgodnie z naszym wyborem, którego dokonaliśmy przy użyciu `dselect`. `Dselect` może być naprawdę potężnym narzędziem, gdy jest używany razem z APT-em.

3.8 Jak utrzymywać “wymieszany” system

Czasami ludzie chcą używać jednej z wersji Debiana jako ich głównego systemu i jednego lub kilku pakietów z innej wersji.

Aby określić, która wersja Debiana jest Twoją wersją główną, w pliku `/etc/apt/apt.conf` powinieneś umieścić linię:

```
APT::Default-Release "wersja";
```

gdzie *wersja* jest wersją Debiana, którą chcesz używać jako swojej głównej dystrybucji. Wersjami, których możesz używać są *stable* ('stabilna'), *testing* ('testowa') i *unstable* ('niestabilna'). Aby zainstalować pakiet z innej wersji musisz użyć APT-a w następujący sposób:

```
# apt-get -t dystrybucja install pakiet
```

Aby polecenie zadziało, potrzebujesz przynajmniej jednej linii w pliku `/etc/apt/sources.list` ze źródłem pakietów dla dystrybucji, z której pakiet chcesz zainstalować i źródło te musi zawierać ten pakiet.

Możesz również dokładnie określić wersję instalowanego pakietu używając składni:

```
# apt-get install pakiet=wersja
```

Dla przykładu, poniższa linia zainstaluje wersję 2.2.4-1 pakietu `nautilus`:

```
# apt-get install nautilus=2.2.4-1
```

WAŻNE: 'niestabilna' wersja Debiana to wersja, do której w pierwszej kolejności trafiają najnowsze wersje pakietów `.deb`. W tej dystrybucji widoczne są wszystkie zmiany pakietów, od tych drobnych do tych bardzo poważnych, które wpływają na wiele innych pakietów, a nawet cały system. Z tego powodu ta wersja dystrybucji *nie* powinna być używana przez niedoświadczonych użytkowników oraz tych, którzy potrzebują stabilnego systemu.

Wersja 'testowa' dystrybucji jest trochę lepsza od 'niestabilnej' pod względem stabilności, ale systemy produkcyjne powinny używać wersji 'stabilnej'.

3.9 Jak uaktualniać pakiety do nowszej wersji z określonej wersji Debiana

Program `apt-show-versions` umożliwia użytkownikom "wymieszanych" dystrybucji bezpieczne uaktualnienie swoich systemów bez ryzyka obniżenia ich dotychczasowej stabilności. Dla przykładu, istnieje możliwość uaktualnienia do nowszej wersji Twoich niestabilnych pakietów po zainstalowaniu pakietu `apt-show-versions`:

```
# apt-get install `apt-show-versions -u -b | grep unstable`
```

3.10 Jak zachować określone wersje zainstalowanych pakietów

Możesz mieć okazję modyfikować coś w pakiecie i nie mieć czasu lub chęci, aby dokonywać tych zmian w nowych wersjach programów. Możesz też dla przykładu uaktualnić swoją dystrybucję Debiana do wersji 3.0, ale chcesz wciąż używać pewnych wersji pakietów z Debiana 2.2. Możesz “przyszpilić” wersję pakietu, którą zainstalowałeś i która nie będzie uaktualniona.

Używanie tego sposobu jest proste. Po prostu musisz wyedytować plik `/etc/apt/preferences`.

Jego format jest także prosty:

```
Package: <pakiet>
Pin: <definicja "przyszpilenia">
Pin-Priority: <priorytet "przyszpilenia">
```

Na przykład, aby zachować pakiet `sylpheed`, który zmodyfikowałem, po to by móc używać opcji “reply-to-list” w wersji 0.4.99, dodałem:

```
Package: sylpheed
Pin: version 0.4.99*
```

Zauważ, że użyłem znaku `*` (gwiazdka). To jest tzw. “znak zastępczy”, który mówi, że chcę “przyszpilić” wszystkie wersje pakietów rozpoczynające się od numeru 0.4.99. Tak jest z powodu wersji tych pakietów Debiana i “debianowych korekt” i nie chcę uniknąć instalacji tych korekt. Zatem, dla przykładu, wersje 0.4.99-1 i 0.4.99-10 zostaną zainstalowane, gdy tylko będą dostępne. Zwróć uwagę, że jeśli zmodyfikowałeś pakiet, nie będziesz chciał robić tych rzeczy w ten sposób.

Pole `Pin-Priority` jest opcjonalne; jeśli nie zostało określone, to domyślnie przyjmuje wartość 989.

Przyjrzyjmy się jak działa priorytet “przyszpilenia”. Priorytet mniejszy niż 0 wskazuje, że pakiet nigdy nie powinien być zainstalowany. Priorytety o wartościach od 0 do 100 wskazują pakiety, które nie są zainstalowane i które nie mają dostępnych wersji. Nie będą one brać udziału w procesie wyboru wersji. Priorytet równy 100 jest priorytetem przypisanym do zainstalowanego pakietu, aby zainstalowana wersja pakietu została zastąpiona przez inną wersję, wersja ta musi mieć priorytet większy niż 100.

Priorytety powyżej 100 wskazują, że pakiet powinien być zainstalowany. Zwykle zainstalowana wersja pakietu jest zmieniana tylko w czasie aktualizacji do nowszej wersji. Priorytety o wartościach pomiędzy 100 i 1000 (włącznie) wskazują te typowe zachowanie. Pakiet z takim priorytetem nie zostanie umniejszony do dostępnej wersji o mniejszym numerze. Na przykład jeśli mam zainstalowaną wersję 0.5.3 pakietu `sylpheed` i zdefiniuję “przyszpilenie” wersji 0.4.99 tego pakietu z priorytetem 999, to wersja 0.4.99 *nie* zostanie zainstalowana, aby spełnić “przyszpilenie”. Aby uczynić pakiet “zmniejszonym” i spełnić “przyszpilenie” musisz ustawić priorytet na wartość większą niż 1000.

Przyszpilenie można określić na wersji, wydaniu lub pochodzeniu pakietu.

Przyszpilenie wersji, tak jak widzieliśmy, wspiera dosłowne numery wersji, jak i znaki zastępcze w celu określenia kilku wersji jednocześnie.

Opcja wydanie zależy od pliku Release z repozytorium APT-a lub z płyty CD. Opcja ta może nie być użyteczna zawsze jeśli używasz repozytoriów pakietów, które nie dostarczają tego pliku. Możesz zobaczyć zawartość plików Release, które masz w pliku `/var/lib/apt/lists/`. Parametrami dla wydania są `a` (archiwum), `c` (komponenty), `v` (wersja), `o` (pochodzenie) and `l` (etykieta).

Przykład:

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Pin-Priority: 1001
```

W tym przykładzie wybraliśmy wersję Debiana 2.2* (którą może być dla przykładu korekta 2.2r2 albo 2.2r3 - określa ona tzw. "punkt wydania", który zwykle zawiera nowe poprawki bezpieczeństwa i inne bardzo ważne uaktualnienia), repozytorium `stable`, sekcję `main` (w przeciwieństwie do sekcji `contrib` lub `non-free`) oraz pochodzenie i etykietę `Debian`. Pochodzenie (`o=`) definiuje kto stworzył plik Release, a etykieta (`l=`) - nazwę dystrybucji, na przykład `Debian` dla Debiana, `Progeny` dla Progeny. Oto przykładowy plik Release:

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i386
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

Rozdział 4

Bardzo użyteczni pomocnicy

4.1 Jak zainstalować lokalnie skompilowane pakiety: pakiet `equivs`

Czasami użytkownicy Debiana chcą używać specyficznej wersji jakiegoś programu, który nie posiada pakietu `.deb` i jest dostępny tylko w postaci kodu źródłowego. Instalacja takich programów może jednak spowodować problemy w działaniu systemu pakietów. Wyobraźmy sobie na przykład, że chcemy skompilować nową wersję naszego serwera pocztowego. Wszystko poszło świetnie, tylko że zapomnieliśmy o małym szczególe: w Debianie wiele pakietów zależy od agenta pocztowego MTA (ang. Mail Transport Agent), a system pakietów niestety nic nie wie o programie, który sami skompilowaliśmy ze źródeł i zainstalowaliśmy!

W takich właśnie przypadkach z pomocą przychodzi nam pakiet `equivs`. Aby móc z niego skorzystać, należy zainstalować pakiet o takiej samej nazwie. Cóż on takiego robi? Tworzy pusty pakiet, który potrafi w pełni spełnić zależności i sprawia, że system pakietów jest przekonany o tym, iż zależności są spełnione.

Zanim pokażemy jak używać tego pakietu, należy jeszcze przypomnieć, że istnieją bezpieczniejsze sposoby kompilacji programu, który posiada już pakiet Debiana i że nie należy używać pakietu `equivs` do zastępowania zależności, gdy nie wiemy jak to zrobić. Więcej informacji na ten temat można znaleźć w sekcji 'Praca z pakietami źródłowymi' na [31](#) stronie.

Kontynuujmy nasz przykład z agentem pocztowym MTA. Powiedzmy, że właśnie skompilowałeś i zainstalowałeś nową wersję serwera `postfix`, a teraz zamierzasz zainstalować pakiet `mutt`. Nagle okazuje się, że `mutt` chce zainstalować innego agenta MTA, a Ty już masz przecież swój.

Przejdź wtedy do jakiegoś katalogu (np. do `/tmp`) i wydaj komendę:

```
# equivs-control nazwa
```

gdzie *nazwa* to nazwa pliku kontrolnego, który chcesz stworzyć. Plik ten będzie miał następującą zawartość:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: <enter package name; defaults to equivs-dummy>
Version: <enter version here; defaults to 1.0>
Maintainer: <your name and email address; defaults to username>
Pre-Depends: <packages>
Depends: <packages>
Recommends: <packages>
Suggests: <package>
Provides: <(virtual)package>
Architecture: all
Copyright: <copyright file; defaults to GPL2>
Changelog: <changelog file; defaults to a generic changelog>
Readme: <README.Debian file; defaults to a generic one>
Extra-Files: <additional files for the doc directory, comma separated>
Description: <short description; defaults to some wise words>
    long description and info
    .
    second paragraph
```

Teraz musimy tak go zmodyfikować, żeby robił to, co chcemy. Spójrzmy na format pól i ich opisy. Nie ma oczywiście potrzeby objaśniać każdego z nich. Zróbmy tylko to, co jest niezbędne:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

Tak, to naprawdę wszystko, czego nam potrzeba! Pakiet `mutt` zależy od pakietu wirtualnego `mail-transport-agent`, którego dostarczają wszystkie agenty MTA. Mogłem też nazwać pakiet po prostu `mail-transport-agent`, ale wolę używać schematu pakietów wirtualnych, który wykorzystuje pole 'Provides'.

Teraz tylko musimy zbudować pakiet:

```
# equivs-build nazwa
dh_testdir
touch build-stamp
dh_testdir
dh_testroot
```

```
dh_clean -k
# Add here commands to install the package into debian/tmp.
touch install-stamp
dh_testdir
dh_testroot
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installddeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package `nazwa' in `../nazwa_1.0_all.deb'.
```

The package has been created.

Attention, the package has been created in the current directory,

i zainstalować powstały pakiet `.deb`.

Jak widać, pakietu `equivs` można używać na różne sposoby. Za jego pomocą można nawet stworzyć pakiet `moje-ulubione`, który zależy od programów, które zwykle instalujesz. To jak go wykorzystasz zależy tylko od Twojej wyobraźni, ale bądź ostrożny!

Na koniec jeszcze jedna ważna uwaga: przykłady plików kontrolnych znajdują się w katalogu `/usr/share/doc/equivs/examples`. Przyjrzyj się im.

4.2 Usuwanie nieużywanych plików lokalizacyjnych: pakiet `localepurge`

Wielu użytkowników Debiana korzysta tylko z jednych plików lokalizacyjnych. Dla przykładu, brazylijscy użytkownicy Debiana przez cały czas korzystają z plików `pt_BR` i do niczego im nie są potrzebne pliki `es`.

Pakiet `localepurge` jest bardzo użytecznym narzędziem dla takich właśnie użytkowników. Pozwala on zaoszczędzić wiele miejsca na dysku, dzięki zachowaniu tylko tych plików lokalizacyjnych, których naprawdę używasz. Aby go zainstalować, po prostu wydaj komendę `apt-get install localepurge`.

Pakiet ten jest bardzo łatwy w konfiguracji, `debconf` przeprowadzi Cię przez ten proces krok po kroku. Bądź jednak bardzo ostrożny w czasie odpowiadania na zadawane Ci pytania, gdyż zła odpowiedź może usunąć wszystkie pliki lokalizacyjne, nawet te, których używasz! Jedynym sposobem na odzyskanie ich wtedy jest powtórna instalacja wszystkich pakietów, które dostarczają tych plików.

4.3 Jak się dowiedzieć, które pakiety mogą być uaktualnione do nowszej wersji

Programem, który potrafi wskazać pakiety mogące być uaktualnione do nowszej wersji oraz dostarczyć wielu innych użytecznych informacji jest program `apt-show-versions`. Do wyświetlenia listy pakietów, które mogą być uaktualnione służy opcja `-u`:

```
$ apt-show-versions -u
libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

Rozdział 5

Pobieranie informacji o pakietach

Jest wiele programów-nakładek dla systemu APT, które bardzo ułatwiają pobieranie listy pakietów dostępnych do instalacji i już zainstalowanych, pozwalają dowiedzieć się do jakiej sekcji należy pakiet, jaki jest jego priorytet, jaki ma opis, itd.

Ale... naszym celem jest nauczyć się posługiwania się czystym APT-em. Zatem w jaki sposób dowiedzieć się nazwy pakietu, który chcemy zainstalować?

Mamy wiele środków do tego zadania. Zacznijmy od programu `apt-cache`. Jest on używany przez system APT do zarządzania jego bazą danych. Przyjrzymy się krótko jego bardziej praktycznym zastosowaniom.

5.1 Odnajdywanie nazw pakietów

Dla przykładu przypuśćmy, że chcemy powspominać o starych, dobrych czasach Atari 2600. Najpierw chcemy użyć APT-a do instalacji emulatora Atari, a następnie do pobrania kilku gier dla niego. Możemy to zrobić następująco:

```
# apt-cache search atari
atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

Znaleźliśmy kilka pakietów powiązanych z tym, czego szukamy, wraz z ich krótkimi opisami. Aby uzyskać więcej informacji na temat określonego pakietu, możesz wtedy użyć:

```
# apt-cache show stella
```

```
Package: stella
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60falc4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++. You can play most Atari 2600 games with it. The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

Otrzymaliśmy w wyniku wiele szczegółów na temat pakietu, który chcemy (lub który nie chcemy) zainstalować, razem z jego pełnym opisem. Jeśli pakiet jest już zainstalowany w systemie i jest tam jego nowsza wersja, to zobaczysz informacje na temat obu wersji. Na przykład:

```
# apt-cache show lilo
Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (<<1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text screen)
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
Priority: important
```

```
Section: base
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
.
You can use Lilo to manage your Master Boot Record (with a simple text screen)
or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

Zauważ, że pierwszy z pakietów na liście jest pakietem dostępnym, a drugi już zainstalowanym. Aby otrzymać więcej ogólnych informacji na temat pakietu, możesz użyć:

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-1 (/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_main
Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0) libso
Provides:
1.4.5-1 -
Reverse Provides:
```

I aby dowiedzieć się od jakich pakietów on zależy:

```
# apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

Podsumowując, mamy wiele sposobów, których możemy użyć, aby dowiedzieć się nazwy pakietu, który chcemy zainstalować.

5.2 Używanie programu dpkg do dowiadywania się nazwy pakietów

Jednym ze sposobów dowiedzenia się nazwy pakietu jest poznanie nazwy ważnego pliku umieszczonego wewnątrz pakietu. Dla przykładu, aby znaleźć pakiet, który jest dostarczany ze szczególnym plikiem nagłówkowym ".h" potrzebnym do jego kompilacji możesz uruchomić:

```
# dpkg -S stdio.h
libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h
```

lub:

```
# dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

Aby dowiedzieć się nazwy użytecznych pakietów zainstalowanych w Twoim systemie, na przykład, gdy zamierzasz doprowadzić do porządku Twój twardy dysk, możesz użyć:

```
# dpkg -l | grep mozilla
ii  mozilla-browser 0.9.6-7          Mozilla Web Browser
```

Problemem związanym z tą komendą może być "obcięcie" przez nią nazwy pakietu. W powyższym przykładzie pełną nazwą pakietu jest mozilla-browser. Aby poradzić sobie z tym, możesz użyć zmiennej środowiskowej COLUMNS w następujący sposób:

```
[kov]@[couve] $ COLUMNS=132 dpkg -l | grep mozilla
ii  mozilla-browser 0.9.6-7          Mozilla Web Brows
```

lub możesz użyć opisu pakietu lub jego części tak, jak pokazano poniżej:

```
# apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

5.3 Jak zainstalować pakiety "na żądanie"

Kompilujesz program, a tu nagle bęc! Wystąpił błąd, ponieważ potrzebny jest plik nagłówkowy .h, którego nie masz. Program auto-apt może Cię uchronić od takich sytuacji. Pyta Cię on czy zainstalować pakiety, które są potrzebne, zatrzymując dany proces i kontynuując go, gdy pakiet jest instalowany.

Wszystko, co musisz zrobić, to uruchomić:

```
# auto-apt run komenda
```

Gdzie 'komenda' jest komendą, której może potrzebować jakiś niedostępny plik. Na przykład:

```
# auto-apt run ./configure
```

Polecenie zapyta Cię czy zainstalować potrzebne pakiety i automatycznie wywoła program apt-get. Jeśli używasz X Window, graficzny interfejs zastąpi domyślny interfejs tekstowy.

Auto-apt używa baz danych, które muszą być uaktualniane, aby program działał skutecznie. Jest to osiągnięte poprzez wywoływanie poleceń `auto-apt update`, `auto-apt updatedb` i `auto-apt update-local`.

5.4 Jak sprawdzić, do którego pakietu należy plik

Jeśli chcesz zainstalować jakiś pakiet, ale nie wiesz jak się on nazywa i nawet wyszukiwanie za pomocą programu `apt-cache` nie przynosi rezultatu, ale znasz za to nazwę pliku tego programu lub innych plików należących do pakietu, to możesz się posłużyć programem `apt-file`, aby odszukać nazwę tego pakietu. W tym celu wydaj polecenie:

```
$ apt-file search nazwa_pliku
```

Działa ono jak komenda `dpkg -S`, z tą różnicą, że wyświetla również nazwy niezainstalowanych pakietów, które zawierają ten plik. Możesz go również użyć, aby dowiedzieć się, jakie pakiety zawierają brakujące w czasie kompilacji programu pliki nagłówkowe. Do tego celu bardziej jednak nadaje się program `auto-apt`, który został opisany w sekcji 'Jak zainstalować pakiety "na żądanie"' na poprzedniej stronie.

Możesz także zobaczyć jakie pliki wchodzi w skład pakietu, wydając polecenie

```
$ apt-file list nazwa_pakietu
```

Program `apt-file`, podobnie jak `auto-apt`, przechowuje informacje o tym jakie pliki zawierają wszystkie pakiety w bazie danych, którą należy co jakiś czas uaktualniać. Robi się do za pomocą komendy

```
# apt-file update
```

Domyślnie oba programy używają tej samej bazy danych. Więcej informacji na jej temat znajdziesz w sekcji 'Jak zainstalować pakiety "na żądanie"' na sąsiedniej stronie.

5.5 Jak otrzymywać informacje o zmianach w pakietach

Każdy zainstalowany pakiet zapisuje w swoim katalogu (`/usr/share/doc/nazwa_pakietu`) plik o nazwie `changelog.Debian.gz`, który zawiera listę zmian dokonanych w pakiecie od jego ostatniej wersji. Możesz czytać te pliki używając na przykład polecenia `zless`, ale czasem nie jest wcale takie proste, zwłaszcza po uaktualnieniu całego systemu, szukać zmian dla każdego zaktualizowanego pakietu.

Istnieje sposób automatyzacji tego zadania przy pomocy narzędzia zwanego `apt-listchanges`. Aby móc z niego korzystać musisz oczywiście zainstalować pakiet `apt-listchanges`. Zostanie on skonfigurowany przez `Debconf` w trakcie instalacji. Odpowiedz na pytania tak, jak chcesz.

Bardzo użyteczna jest opcja “Should `apt-listchanges` be automatically run by `apt`? (Czy `apt-listchanges` powinien być automatycznie uruchamiany przez `APT`-a?)”, ponieważ pokazuje ona listę zmian w każdym pakiecie, który będzie instalowany przez `APT`-a w trakcie aktualizacji systemu i pozwala zapoznać się z nimi przed kontynuacją procesu aktualizacji. Również opcja “Should `apt-listchanges` prompt for confirmation after displaying changes? (Czy `apt-listchanges` powinien zachęcać do potwierdzenia po wyświetleniu zmian?)” jest użyteczna, gdyż pyta Cię czy chcesz kontynuować instalację po przeczytaniu listy zmian. Jeśli odpowiesz, że nie chcesz kontynuować, to `apt-listchanges` zwróci błąd i `APT` przerwie proces instalacji.

Po zainstalowaniu pakietu `apt-listchanges`, gdy tylko pakiety będą ściągane (lub pobierane z płyty CD-ROM albo zamontowanego dysku) przez `APT`-a, to pokaże on listę zmian dokonanych w tych pakietach jeszcze przed ich instalacją.

Rozdział 6

Praca z pakietami źródłowymi

6.1 Pobieranie pakietów źródłowych

W świecie wolnego oprogramowania jest rzeczą powszechną studiować kod źródłowy programów, a nawet dokonywać w nim zmian, aby wyeliminować błędy. Aby to zrobić, musisz pobrać źródło programu. APT dostarcza Ci łatwego sposobu uzyskiwania kodów źródłowych wielu programów zawartych w dystrybucji, włączając w to wszystkie pliki potrzebne do stworzenia pakietu `.deb` dla danego programu.

Innym powszechnym zastosowaniem źródeł Debiana jest przystosowanie nowszej wersji programu, pochodzącej np. z niestabilnej dystrybucji, aby użyć go w wersji stabilnej. Zamiast pakietu stabilnego można skompilować źródła i wygenerować pakiety `.deb` z zależnościami przystosowanymi do pakietów dostępnych w tej dystrybucji.

Aby to osiągnąć, wpis ze słowem `deb-src` na początku linii w Twoim pliku `/etc/apt/sources.list` powinien wskazywać na archiwum niestabilne. Oczywiście powinien on być także włączony (tzn. odkomentowany). Zobacz sekcję 'Plik `/etc/apt/sources.list`' na 3 stronie, aby dowiedzieć się więcej szczegółów.

Aby pobrać pakiet źródłowy, użyjemy następującej komendy:

```
$ apt-get source pakiet
```

Komenda ta pobierze trzy pliki: `.orig.tar.gz`, `.dsc` i `.diff.gz`. W przypadku pakietów tworzonych specjalnie dla Debiana nie będzie pobierany ostatni plik, a pierwszy zwykle nie ma w nazwie słowa "orig".

Plik `.dsc` jest używany przez program `dpkg-source` do rozpakowania źródeł pakietu w katalogu `pakiet-wersja`. Wewnątrz każdego pobranego pakietu istnieje katalog `debian/`, zawierający pliki niezbędne do stworzenia pakietu `.deb`.

Aby automatycznie zbudować pakiet w czasie pobierania jego źródeł, po prostu dodaj do komendy opcję `-b` tak, jak pokazano to poniżej:

```
$ apt-get -b source pakiet
```

Jeśli nie zdecydowałeś się stworzyć pakietu `.deb` w czasie pobierania jego źródeł, możesz zrobić to później za pomocą polecenia:

```
$ dpkg-buildpackage -rfakeroot -uc -b
```

uruchomionego wewnątrz katalogu, który został utworzony dla pakietu po jego pobraniu. Aby zainstalować pakiet zbudowany przy pomocy powyższej komendy, musisz użyć bezpośrednio programu `dpkg`:

```
# dpkg -i file.deb
```

Istnieje różnica pomiędzy metodą `source` programu `apt-get`, a jego innymi metodami. Metoda `source` może być używana przez zwykłych użytkowników, nie posiadających uprawnień `roota`. Pliki są pobierane do katalogu z którego wywołano komendę `apt-get source pakiet`.

6.2 Pakiety potrzebne do kompilowania pakietów źródłowych

Zwykle specyficzne pliki nagłówkowe i biblioteki współdzielone muszą być obecne, aby można było skompilować pakiet źródłowy. Wszystkie pakiety źródłowe mają pole w plikach kontrolujących, które nazywa się 'Build-Depends:'. Pole te wskazuje, które dodatkowe pakiety są niezbędne, aby można było zbudować pakiet z jego źródeł.

APT jest wyposażony w łatwy sposób pobierania tych pakietów. Po prostu uruchom komendę `apt-get build-dep pakiet`, gdzie 'pakiet' jest nazwą pakietu, który zamierzasz zbudować. Na przykład:

```
# apt-get build-dep gmc
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-implib-dev implib-progs libgnome-dev libgnorba-de
  libgpmgl-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1 not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

Pakiety, które zostaną zainstalowane są pakietami niezbędnymi do prawidłowego zbudowania pakietu `gmc`. Bardzo ważną rzeczą do zauważenia jest fakt, że komenda nie szuka pakietu ze źródłami programu, który ma być skompilowany. Dlatego musisz wcześniej uruchomić polecenie `apt-get source`, aby osobno je pobrać.

Jeśli chcesz sprawdzić, które pakiety są potrzebne do zbudowania powyższej paczki, możesz użyć wariantu polecenia `apt-cache show` (zobacz sekcję 'Pobieranie informacji o pakietach' na [25](#) stronie), pokazującego oprócz innych informacji także linię `Build-Depends` z listą pakietów niezbędnych do zbudowania pakietu.

```
# apt-cache showsrc package
```


Rozdział 7

Jak radzić sobie z błędami

7.1 Najczęstsze błędy

Błędy zawsze będą się zdarzały, wiele z nich jest spowodowana przez użytkowników, którzy nie zachowują uwagi w czasie pracy z APT-em. Poniżej zamieściłem najczęściej zgłaszane błędy oraz informacje jak sobie z nimi radzić.

Jeśli próbujesz uruchomić `apt-get install package` i zobaczysz komunikat jak poniżej:

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/ Pack
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

to znaczy, że zapomniałeś uruchomić `apt-get update` po ostatnich zmianach, których dokonałeś w pliku `/etc/apt/sources.list`.

Jeśli błąd wygląda następująco:

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root
```

gdy próbujesz użyć którejś metody polecenia `apt-get` z wyjątkiem metody `source`, to znaczy, że próbujesz ją uruchomić jako zwykły użytkownik i że nie masz niezbędnych uprawnień administratora systemu.

Ten błąd jest podobny do tego powyżej i ma miejsce, gdy uruchamiasz w tym samym czasie dwie kopie programu `apt-get` lub gdy próbujesz uruchomić `apt-get`, gdy aktywny jest proces programu `dpkg`. Jediną metodą, która może być równocześnie użyta z inną jest metoda `source`.

Jeśli instalacja zostanie przerwana w trakcie trwania i nie jest już możliwa ani instalacja, ani usunięcie pakietów, spróbuj wtedy użyć tych dwóch komend:

```
# apt-get -f install
# dpkg --configure -a
```

I spróbuj jeszcze raz. Może być niezbędne uruchomienie drugiej z komend więcej niż jeden raz. To jest ważna uwaga dla tych, którzy używają wersji 'niestabilnej'.

Jeśli wynikiem działania polecenia `apt-get update` jest błąd "E: Dynamic MMap ran out of room", dodaj do swojego pliku `/etc/apt/apt.conf` poniższą linię:

```
APT::Cache-Limit 10000000;
```

7.2 Gdzie mogę znaleźć pomoc

Jeśli popadłeś już w zwątpienie, zajrzyj do obszernej dokumentacji na temat systemu pakietów Debiana. Opcja `--help` programów i ich podręczniki mogą być ogromnie pomocne dla Ciebie. Wiele informacji można też znaleźć w dokumentacji zawartej w katalogu `/usr/share/doc`, np. w katalogu `/usr/share/doc/apt`.

Gdy nawet ta dokumentacja nie zdoła Ci pomóc, spróbuj poszukać odpowiedzi na listach dyskusyjnych Debiana. Więcej informacji na temat poszczególnych list użytkowników można znaleźć na stronie Debian: <http://www.debian.org>.

Pamiętaj, że listy te i zasoby powinny być używane tylko przez użytkowników Debiana; użytkownicy innych systemów znajdą lepsze wsparcie w zasobach ich własnej dystrybucji.

Rozdział 8

Które dystrybucje wspierają APT-a?

Poniżej zamieszczono nazwy dystrybucji, które używają APT-a:

Debian GNU/Linux (<http://www.debian.org>) - to dla tej dystrybucji został stworzony APT

Conectiva (<http://www.conectiva.com.br>) - to była pierwsza dystrybucja, która zaadoptowała APT-a do pracy z pakietami .rpm

Libranet (<http://www.libranet.com>)

Mandrake (<http://www.mandrake.com>)

PLD (<http://www.pld.org.pl>)

Vine (<http://www.vinelinux.org>)

APT4RPM (<http://apt4rpm.sf.net>)

Alt Linux (<http://www.altlinux.ru/>)

Red Hat (<http://www.redhat.com/>)

Sun Solaris (<http://www.sun.com/>)

SuSE (<http://www.suse.de/>)

Yellow Dog Linux (<http://www.yellowdoglinux.com/>)

Rozdział 9

Podziękowania

Wielkie podziękowania składam moim wspaniałym przyjaciołom z projektu Debian-BR i samego projektu Debian, tym którzy są stałą pomocą dla mnie, zawsze dają mi siłę do kontynuowania pracy dla ludzkiego pożytku i pomagają mi w moim celu jakim jest ratowanie świata. :)

Chciałbym także podziękować CIPSGA za olbrzymią pomoc udzieloną naszemu projektowi i wszystkim wolnym projektom, które zrodziły się z wielkich idei.

Specjalne podziękowania należą się następującym osobom:

Yooseong Yang <yooseong@debian.org>

Michael Bramer <grisu@debian.org>

Bryan Stillwell <bryan@bokeoa.com>

Pawel Tecza <ptecza@debianusers.pl>

Hugo Mora <h.mora@melix.com.mx>

Luca Monducci <luca.mo@tiscali.it>

Tomohiro KUBOTA <kubota@debian.org>

Pablo Lorenzoni <spectra@debian.org>

Steve Langasek <vorlon@netexpress.net>

Arnaldo Carvalho de Melo <acme@conectiva.com.br>

Erik Rossen <rossen@freesurf.ch>

Ross Boylan <RossBoylan@stanfordalumni.org>

Matt Kraai <kraai@debian.org>

Aaron M. Ucko <ucko@debian.org>

Jon Aslund <d98-jas@nada.kth.se>

Rozdział 10

Nowe wersje tego podręcznika

Ten podręcznik został stworzony w ramach projektu Debian-BR (<http://www.debian-br.org>) z zadaniem niesienia pomocy w codziennym używaniu Debiana.

Nowe wersje tego dokumentu będą dostępne na stronie Projektu Dokumentacji Debiana (Debian Documentation Project) pod adresem <http://www.debian.org/doc/ddp>.

Komentarze i krytyczne uwagi mogą być wysyłane bezpośrednio do mnie na adres <kov@debian.org>.