

Debian/GNU z perspektywy administratora (2)

GRZEGORZ JACEK NALEPA

16.4.2000, Kraków, *Revision* : 1.8

Streszczenie

Artykuł jest drugim z cyklu opisującego specyfikę zarządzania pakietami w systemie Debian/GNU Linux. W artykule jest przedstawiony program `dselect` wraz z opisem organizacji bazy danych pakietów. Oprócz tego są opisane pakiety źródłowe i metody budowania pakietów binarnych DEB. Przedstawione w artykule omówienie ma na celu przybliżenie Debian/GNU administratorom i użytkownikom systemu GNU/Linux.

Spis treści

1	Dselect	2
1.1	Access	3
1.2	Update	4
1.3	Select	5
1.3.1	Okno główne Select	5
1.3.2	Opis pakietu w oknie głównym	5
1.3.3	Zależności	7
1.3.4	Pakiety wirtualne	8
1.4	Install	8
1.5	Pozostałe funkcje <code>dselect</code>	8
2	Baza danych o pakietach	9
3	Pakiety typu task	9
4	Pakiety źródłowe DEB	10
4.1	Rozpakowywanie pakietów źródłowych	10
4.2	Informacje kontrolne pakietu	11
4.3	Budowanie pakietów binarnych	11
5	Podsumowanie	13

¹Tekst ukazał się w: *Magazynie Linux & Unix*, nr 6/2000, wydawanym przez TAO Systems.

²Kontakt z autorem: [mail:gjn@agh.edu.pl](mailto:gjn@agh.edu.pl)

³Tytuł angielski: *Debian/GNU – Administrator's Perspective (2)*

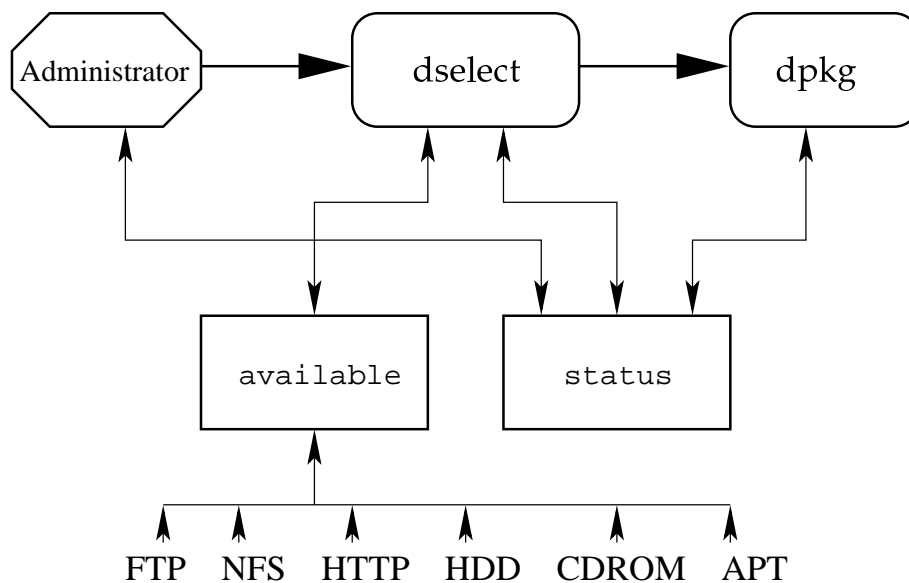
⁴Tekst jest rozpowszechniany na zasadach licencji *GNU Free Documentation License*, której pełny tekst można znaleźć pod adresem: <http://www.gnu.org/copyleft/fdl.html>

W poprzednim artykule został przedstawiony system pakietów Debian/GNU oraz podstawowe cechy programu `dpkg`. `Dpkg` jest typowym narzędziem używanym z linii poleceń, co ma swoje zalety jak i niedogodności. W związku z dużą złożonością systemu pakietów w dystrybucji Debian/GNU oraz znaczną liczbą pakietów, zostało stworzone narzędzie `dselect`, będące zaawansowanym interfejsem do systemu pakietów. W tym artykule zostanie przedstawiony `dselect` wraz z opisem organizacji bazy danych pakietów. Oprócz tego zostaną opisane pakiety źródłowe i metody budowania pakietów binarnych DEB.

1. Dselect

Program `dselect` jest interfejsem do systemu pakietów Debiana i programu `dpkg`. Umożliwia przede wszystkim przeglądanie bazy danych pakietów, wielokryterialne sortowanie informacji o dostępnych i zainstalowanych pakietach oraz aktualizowanie bazy danych pakietów i dostęp do pakietów z wielu źródeł takich jak dysk twardy, napęd CD, sieć lokalna, czy Internet. Pozwala na selekcjonowanie pakietów, ich instalowanie oraz odinstalowanie wraz z automatycznym sprawdzaniem zależności pomiędzy pakietami.

Program `dselect` komunikuje się z użytkownikiem (administratorem) i współpracuje z systemem pakietów i narzędziem `dpkg`. Przybliżony schemat tej współpracy jest pokazany na Rysunku 1. Informacje o aktualnie zainstalowanych i wyselekcjonowanych pakietach są przechowywane



Rysunek 1: Schemat pracy `dselect`

wywane w pliku `status`. Dane o dostępnych pakietach, uzyskiwane w zależności od wybranej metody dostępu, są przechowywane w pliku `available`.

Po uruchomieniu programu `dselect` wyświetlane jest menu główne programu, składające się z siedmiu pozycji:

- *Access* – wybranie źródła z którego będą pobierane informacje o pakietach oraz same pakiety DEB,
- *Update* – aktualizowanie informacji o dostępnych pakietach z wybranego w poprzednim punkcie źródła,

- *Select* – przeglądanie bazy danych o pakietach dostępnych i zainstalowanych oraz selekcjonowanie pakietów do zainstalowania i odinstalowania,
- *Install* – instalowanie pakietów wybranych przy pomocy *Select*,
- *Config* – konfiguracja zainstalowanych pakietów, które nie zostały wcześniej skonfigurowane,
- *Remove* – odinstalowanie pakietów wybranych przy pomocy *Select*,
- *Quit* – wyjście z programu *dselect*.

Dostęp do pozycji menu jest możliwy przy pomocy klawiszy kursora i klawisza Enter, poprzez cyfry od 0 do 6, lub przez pierwsze litery nazw (a,u,s,i,c,r,q). Kolejność pozycji menu odpowiada kolejności ich wywoływania. W związku z tym, najprościej jest je omówić właśnie w takim porządku.

1.1. Access

Przed instalowaniem pakietów należy wskazać programowi *dselect* gdzie się one znajdują. Funkcja *Access* pozwala na wybranie jednej z wielu metod dostępu do pakietów DEB. Standardowo w dystrybucji Debian/GNU 2.1 są to: *cdrom*, *multi_cd*, *nfs*, *multi_nfs*, *harddisk*, *mounted*, *multi_mount*, *floppy*, *ftp*, *apt*.

Metody *cdrom* i *multi_cd* pozwalają na instalowanie pakietów znajdujących się na jednej lub wielu płytach CD. W przypadku drugiej metody *dselect* prosi w miarę potrzeb o zmianę płyt w napędzie CDROM. Płyty CD mogą, ale nie muszą być podmontowane ręcznie. Metody *nfs* i *multi_nfs* pozwalają na instalację pakietów przez sieć lokalną z wykorzystaniem serwisu NFS. Metody *harddisk*, *mounted* i *multi_mount* umożliwiają instalowanie pakietów z podmontowanej lub nie podmontowanej partycji albo katalogu w systemie plików. Metoda *floppy* umożliwia instalowanie pakietów z dyskietek. Wszystkie z wymienionych metod do poprawnej pracy wymagają prawidłowych plików *Packages* znajdujących się w odpowiednich katalogach drzewa Debian. Najprościej jest instalować pakiety z drzewa plików stanowiącego kopię fragmentu oryginalnego drzewa plików Debiana (przedstawionego w pierwszym artykule). Jeżeli taka kopia nie jest dostępna można ręcznie wprowadzić katalogi w których znajdują się pakiety.

Metoda *ftp* umożliwia automatyczne kopiowanie pakietów, które się chce zainstalować ze wskazanego serwera FTP. Jest wygodna w przypadku instalowania bez dostępu do lokalnej kopii dystrybucji, na przykład z odległego serwera FTP. Metoda *ftp* nie jest częścią standardowego *dselect*, trzeba ją doinstalować jako dodatkowy pakiet.

Najciekawsza i przeważnie najwygodniejsza metoda to *apt*. Posługuje się narzędziem, które w przyszłości stanie się podstawowe w instalowaniu pakietów. Umożliwia kopiowanie pakietów spod dowolnego wskazanego zestawu adresów URL, takich jak *ftp* czy *http*. Podając URL typu *file* można instalować pakiety z dysku twardego lub podmontowanego CDROMu, czy partycji NFS. APT i jego konfiguracja będą dokładniej opisane w następnym artykule.

Należy podkreślić, że praca z *dselect* przebiega przeważnie bezproblemowo, jeżeli administrator czyta informacje wyświetlane przez program. Są one bardzo pomocne w trakcie pracy i umożliwiają pracę nawet niedoświadczonym użytkownikom.

Przykładowo, jeżeli chce się instalować pakiety z oficjalnej płyty CDROM Debian/GNU 2.1 Binary1 można wybrać metodę *cdrom* i skonfigurować ją tak, jak poniżej.

```
If you make a mistake, use the interrupt key (^C) to abort.
```

```
I see that /dev/cdrom exists and is a block device.
```

Insert the CD-ROM and enter the block device name [/dev/cdrom]:

All directory names should be entered relative to the root of the CD-ROM.

I would like to know where on the CD-ROM the top level of the Debian distribution is (eg. 'dists/stable') - this directory usually contains the Packages-Master file.

If the CD-ROM is badly organised and doesn't have a straightforward copy of the distribution you may answer 'none' and we'll go through the parts I need individually.

'/debian/dists/stable' exists and looks plausible, so that's the default.
Distribution top level ? [/debian/dists/stable]
Using '/debian/dists/stable/main/binary-i386' as main binary dir.
Using '/debian/dists/stable/main/binary-i386/Packages.gz' for main.

...

Hit RETURN to continue.

W przypadku każdej części dystrybucji (na przykład main, contrib) należy podać katalog z pakietami, oraz pełną ścieżkę dostępu do pliku **Packages** zawierającego spis wszystkich pakietów w tym katalogu.

Jak widać **dselect** stara się automatycznie odnaleźć pliki **packages** i w przypadku wiernej kopii dystrybucji nie stanowi to problemu. Jeżeli pakiety znajdują się w niestandardowych katalogach ścieżkę dostępu należy podać ręcznie.

Dodatkowo istnieje możliwość podania katalogu w którym znajdują się pakiety, a zamiast ścieżki do pliku **Packages** można wpisać słowo **scan**. Powoduje to przeskanowanie znajdujących się w katalogu pakietów i automatyczne wygenerowanie listy pakietów. Jest to metoda wygodna, ale czasochłonna.

1.2. Update

Po wybraniu metody w części *Access* należy zaktualizować bazę danych o dostępnych pakietach (ang. *available*). Umożliwia to funkcja *Update*, posługująca się wybraną poprzednio metodą. W zależności od tego jakie części dystrybucji (main, contrib) są dostępne pod wskazanym źródłem (na przykład na płycie CD) aktualizowane są odpowiednie części bazy danych.

Wybór metody dostępu i aktualizację bazy danych należy przeprowadzać po każdej zmianie źródła pakietów, lub zmianie dostępnych pakietów. Jeżeli przykładowo dotychczas korzystało się z płyty CDROM, a chce się instalować pakiety przez FTP, należy wybrać prawidłową metodę dostępu (*Access*), a następnie zaktualizować bazę (*Update*). Podobnie, jeżeli korzysta się ze zmieniającego się źródła pakietów, na przykład serwera FTP, czy HTTP należy systematycznie aktualizować bazę danych o pakietach, tak aby była zsynchronizowana ze spisem pakietów dostępnych na tym serwerze.

Kontynuując przykład z poprzedniego punktu, po wybraniu i skonfigurowaniu metody cdrom aktualizacja może przebiegać tak jak pokazano poniżej:

```
Uncompressing /var/lib/dpkg/methods/mnt/
```

```
debian/dists/stable/main/binary-i386/Packages.gz ... done.  
Replacing available packages info, using packages-main.  
Information about 1162 package(s) was updated.  
Update OK. Hit RETURN.
```

Jak widać, CDROM został automatycznie podmontowany do katalogu `/var/lib/dpkg/methods/mnt`. Zostały zaktualizowane informacje tylko o pakietach z sekcji `main`, ponieważ tylko takie znajdowały się na płycie.

1.3. Select

Najważniejszą częścią `dselect` jest interfejs umożliwiający selektywne przeglądanie bazy danych pakietów i dokonywanie wyboru pakietów, które mają być zainstalowane lub usunięte.

Po wybraniu funkcji *Select* pojawia się okno pomocy zawierające krótkie wprowadzenie, będące częścią systemu pomocy. Z tego okna można przejść dalej przy pomocy spacji.

W trakcie całej pracy z funkcją *Select* dostępny jest przejrzysty system pomocy uruchamiany przy pomocy klawisza znaku zapytania „?”. Składa się on z czterech części: wprowadzenia do systemu selekcji pakietów (klawisz „i”) oraz opisów używanych klawiszy („k”), informacji dotyczących stanu pakietu („l”) i sposobów zmiany wyglądu okna („d”).

1.3.1. Okno główne Select

Po wyjściu z systemu pomocy przechodzi się do okna głównego pokazującego listę pakietów. Okno, pokazane na Rysunku 2, składa się z dwóch głównych części górnej i dolnej rozdzielonych linią informacyjną oraz dwóch linii informacyjnych na górze i dole okna. Górna, czerwona linia zawiera informacje o aktualnym sposobie sortowania listy pakietów oraz przypomnienie klawiszy umożliwiających zaznaczanie pakietów i przejście do systemu pomocy. Sortowanie pakietów jest dwupoziomowe. Istnieją możliwości sortowania według: dostępności pakietów (ang. *availability*), alfabetycznie (ang. *alphabetically*), kategorii pakietów (ang. *by section*), stanu (zainstalowania) (ang. *status*) i priorytetu (ang. *priority*). Używając klawiszy „o” i „O” (Shift+o) można przełączać różne sposoby sortowania. W zależności od sposobu sortowania i kontekstu w górnym oknie pojawiają się dodatkowe hierarchiczne opisy sortowanych grup. Na rysunku widać na przykład opis „Up-to-date Important packages in section editors”, co można przełożyć jako „Pakiety, które nie wymagają uaktualnienia, o priorytecie ważne, z części edytory”. Kolejność członów zdania nie jest przypadkowa i jest związana ze sposobem sortowania pakietów.

1.3.2. Opis pakietu w oknie głównym

Każda linia w górnej części okna może być opisem grupy pakietów, lub opisem konkretnego pakietu. W tym drugim przypadku składa się z 10 pól opisanych na górze okna, pod czerwonym paskiem opisu. Są to:

1. *E* (*Error*) sygnalizuje błąd. Spacja oznacza brak błędu, „R” oznacza błąd i konieczność reinstalacji pakietu; najczęściej spacja.
2. *I* (*Installed state*) opisuje stan pakietu. Spacja – pakiet nie zainstalowany, „*” – zainstalowany, „-” – nie zainstalowany ale pozostały pliki konfiguracyjne, „U”, „C”, „I” – w trakcie instalacji nastąpił błąd po rozpakowaniu pakietu, w trakcie konfiguracji, lub w innym momencie.

```
dselect - main package listing (avail., priority) mark:+/=- verbose:v help:?
E10M Pri Section Package Inst.ver Avail.ver Description
*** Req base update 1.3-2 1.3-2 daemon to periodically fl
*** Req base util-linux 2.9g-6 2.9g-6 Miscellaneous system util
----- Up-to-date Important packages -----
----- Up-to-date Important packages in section admin -----
*** Imp admin at 3.1.8-4 3.1.8-4 Delayed job execution and
*** Imp admin cron 3.0pl1-50 3.0pl1-50 management of regular bac
----- Up-to-date Important packages in section base -----
*** Imp base lilo 21-4 21-4 LInux LOader - The Classi
----- Up-to-date Important packages in section doc -----
*** Imp doc info 3.12-0.1 3.12-0.1 Standalone GNU Info docum
*** Imp doc man-db 2.3.10-68 2.3.10-68 Display the on-line manual
*** Imp doc manpages 1.22-2 1.22-2 Man pages about using a L
----- Up-to-date Important packages in section editors -----
*** Imp editors ed 0.2-17 0.2-17 The classic unix line edi
----- Up-to-date Important packages in section interpreters -----
*** Imp interpre perl 5.004.04-7 5.004.04-7 Larry Wall's Practical Ex
----- Up-to-date Important packages in section libs -----
*** Imp libs libident 0.21-7 0.21-7 simple RFC1413 client lib
man-db installed; install (was: install). Important
man-db - Display the on-line manual.

This packages provides the man command, this utility is the primary way of
examining the on-line help files (manual pages). Other utilities provided
include the whatis and apropos commands for searching the manual page
database; the manpath utility for determining the manual page search path
and the maintenance utilities mandb, catman and zsoelim. This package uses
the groff suit of programs to format and display the manual pages

description of man-db
```

Rysunek 2: Okno dselect

3. *O (Old mark)*, wskazuje na to czy poprzednio pakiet był wybrany do zainstalowania lub usunięcia (ma takie wartości jak następne pole).

4. *M (Mark)*, wskazuje na wyselekcjonowanie pakietu. „*” – pakiet zostanie zainstalowany, „-” – pakiet zostanie usunięty ale bez plików konfiguracyjnych, „=” – stan pakietu nie zmieni się, „-” – pakiet zostanie usunięty wraz z plikami konfiguracyjnymi, „n” – pakiet jest nowy, to znaczy pojawia się na liście po raz pierwszy.
5. *Priority* – priorytet pakietu.
6. *Section* – kategoria pakietu, część dystrybucji do jakiej należy.
7. *Package* – nazwa pakietu.
8. *Installed version* – wersja aktualnie zainstalowanego pakietu.
9. *Available version* – dostępna wersja pakietu.
10. *Description* – krótki opis pakietu.

Sposób wyświetlania tych pól może być zmieniany przy pomocy klawiszy „v” i „V”. Listę pakietów można przeszukiwać przy pomocy klawisza „/” i powtarzać przeszukiwanie przy pomocy „\”.

Jedna linijka opisująca pakiet jest zawsze podświetlona, co oznacza, że pakiet jest wybrany i można zmieniać jego stan, lub wyświetlać o nim informacje. Środkowa niebieska linia zawiera skrócony opis aktualnie wyselekcjonowanego pakietu.

Druga, dolna część okna zawiera wyczerpujące informacje na temat wyselekcjonowanego pakietu. To, jakie informacje są pokazywane przełącza się klawiszem „i”. Stosunek wielkości dolnego i górnego okna zmienia się przy pomocy klawisza „I”. Mogą to być: obszerny opis pakietu, informacje na temat stanu pakietu, pełne informacje kontrolne pakietu (pochodzące z pliku `control` opisywanego przy okazji pakietów źródłowych) zawierające między innymi zależności pakietu. Warto pamiętać, że opis wszystkich omawianych klawiszy można znaleźć w systemie pomocy przy pomocy „? k”.

Stan pakietu, czyli to, czy ma być zainstalowany (lub uaktualniony) czy usunięty zmienia się przy pomocy klawiszy, które powodują zaznaczenie pakietu do:

- „+” – zainstalowania (uaktualnienia),
- „-” – usunięcia, ale bez plików konfiguracyjnych,
- „-” – usunięcia wraz z plikami konfiguracyjnymi,
- „=” – pozostawienia stanu pakietu w stanie wskazywanym przez pole „O”.

Po każdej zmianie stanu dowolnego pakietu automatycznie sprawdzane są zależności (ang. *dependencies*) pomiędzy tym a innymi pakietami. W celu uzgodnienia zależności `dselect` automatycznie wskazuje które pakietu powinny być zainstalowane, lub usunięte. Spis pakietów, których stan `dselect` proponuje zmienić pojawia się w oddzielnym oknie. Aby potwierdzić sugerowane zmiany należy nacisnąć Enter. Można jednak wymusić decyzję niezgodną z sugestiami programu przy pomocy klawisza „Q”.

1.3.3. Zależności

Zależności pomiędzy pakietami mogą być następujące:

- Pakiet A *zależy* (ang. *depends*) od pakietu B, jeżeli B, a często odpowiednia wersja B, musi być zainstalowany do pracy A.

- Pakiet A *zaleca* (ang. *recommends*) pakiet B, jeżeli twórca pakietu uważa, że B będzie potrzebny większości użytkowników do pracy z A.
- Pakiet A *sugeruje* (ang. *suggests*) pakiet B, jeżeli B ma funkcje rozszerzające możliwości A.
- Pakiet A *jest w konflikcie* (ang. *conflicts*) z pakietem B, jeżeli A nie może być równocześnie zainstalowany z B. Tego typu konflikt ma najczęściej miejsce gdy A *zastępuje* B.
- Pakiet A *zastępuje* (ang. *replaces*) pakiet B, jeżeli zainstalowanie A spowoduje nadpisanie części plików B.
- Pakiet A *zapewnia* (ang. *provides*) (funkcjonalność) pakietu B, jeżeli ma wszystkie funkcje B.

1.3.4. Pakiety wirtualne

Z ostatnią zależnością wiąże się pojęcie *pakietów wirtualnych*. Są to pewne pakiety uogólnione, na przykład takie jak **mail-transport-agent**, **mail-reader**, **xserver**. Oznaczają one tak naprawdę pewną ogólną funkcjonalność pakietów. Na przykład, funkcjonalność pakietu **mail-transport-agent** (serwera poczty) jest realizowana przez programy Exim, Smail i Sendmail, a funkcjonalność pakietu **xserver** jest realizowana przez dowolny XServer, niezależnie od karty graficznej. Mechanizm pakietów wirtualnych jest jedną z wielu zalet systemu pakietów Debian/GNU. Dzięki niemu nie jest konieczne wymuszanie zależności od konkretnego programu (na przykład Sendmail) ile od pewnych jego funkcji. Dlatego właśnie w Debian/GNU koegzystuje wiele różnych programów serwerów i klientów poczty elektronicznej i administrator sam decyduje, czy woli program Exim, Smail czy Sendmail.

Po zakończeniu wybierania pakietów z części *Select* wychodzi się przez klawisz Enter. Ponownie sprawdzane są zależności wszystkich pakietów w celu zachowania spójności instalowanych i zainstalowanych pakietów. W razie potrzeby program sugeruje odpowiednie korekty. Po ostatecznym uzgodnieniu zależności **dselect** powraca do menu głównego i można przystąpić do instalowania pakietów.

1.4. Install

Po wybraniu opcji *Install* z bazy danych wybierane są pakiety, które zostały wyselekcjonowane i określone jest miejsce w którym się znajdują. Następnie pakiety są kolejno instalowane przy pomocy polecenia **dpkg -iGROEB**. Użyte opcje powodują między innymi instalowanie pakietów w wersji nie niższej niż zainstalowane oraz umożliwiają zmianę konfiguracji zainstalowanych pakietów w celu zainstalowania nowych. Podobnie jak w przypadku ręcznego użycia **dpkg**, pakiety są najpierw rozpakowywane, a potem kolejno konfigurowane. Ponieważ często zdarza się, że instalowanych jest wiele pakietów o złożonych zależnościach, może dojść do sytuacji, w której nie wszystkie pakiety zostaną automatycznie skonfigurowane. Taka sytuacja zdarza się bardzo rzadko. Gdyby jednak do niej doszło należy wywołać funkcję *Config* z menu **dselect**.

1.5. Pozostałe funkcje dselect

Funkcja *Config* wywołuje polecenie **dpkg --configure --pending**. Umożliwia skonfigurowanie zainstalowanych pakietów, których konfiguracja nie powiodła się wcześniej.

Wybranie funkcji *Remove* powoduje usunięcie pakietów, które zostały zaznaczone w części *Select* do usunięcia. W przypadku wykonywania sekwencji *Select*, *Install*, *Config* powinna być uruchamiana na końcu, to znaczy po zainstalowaniu i skonfigurowaniu pozostałych pakietów.

Funkcja *Quit* powoduje wyjście z programu *dselect*.

2. Baza danych o pakietach

Na koniec warto powiedzieć kilka słów o tym jak w praktyce zrealizowana jest baza danych o pakietach i gdzie się ona znajduje w systemie plików.

Cała baza danych znajduje się w katalogu `/var/lib/dpkg` i jej schemat jest pokazany na Rysunku 3.

```
/var/lib/dpkg
|
+---alternatives/
+---info/
+---methods/
|   +---disk/
|   +---ftp/
|   +---mnt/
|   +---multitcd/
|
+---available
+---lock
+---methlock
+---status
```

Rysunek 3: Organizacja bazy danych pakietów

Katalog **alternatives** zawiera informacje na temat różnych alternatywnych wersji programów. Przykładowo system pakietów jest w stanie kontrolować kilka równocześnie zainstalowanych wersji powłoki Csh, czy edytora Emacs i odpowiednio modyfikować linki symboliczne dotyczące programów wykonywalnych i plików podręcznika man. Katalog *info* zawiera pliki **preinst**, **postinst**, **prerm** i **postrm**, spis plików i sumy kontrolne zainstalowanych pakietów. W katalogu **methods** znajdują się informacje na temat metod dostępu do pakietów wybieranych w punkcie *Access* i tymczasowe punkty montowania.

Bardzo ważne są pliki **available** i **status** przechowujące informacje na temat dostępnych pakietów oraz pakietów zainstalowanych i ich stanie. Pliki **lock** i **methlock** zabezpieczają przed pracą więcej niż jednej kopii programu *dselect*, czy danej metody (w przypadku użytkownika *root*).

W tym miejscu warto zwrócić uwagę, że podczas gdy większa część katalogu `/var` nie jest krytyczna dla działania systemu i zawiera najczęściej pewnego rodzaju dane tymczasowe, to katalog `/var/lib/dpkg` jest bardzo ważny i powinno się często robić jego kopie zapasowe.

3. Pakiety typu task

Oprócz pakietów wirtualnych istnieje jeszcze inny mechanizm związany z zależnościami między pakietami. W dystrybucji Debian/GNU zostały stworzone pakiety typu **task-nazwa**, gdzie

nazwa odnosi się do pewnej funkcji oprogramowania, lub pewnego zbioru programów. Pakiety **task** same nie zawierają żadnego oprogramowania, lecz mają zależności z innymi pakietami. Dzięki temu, instalując pakiet **task** automatycznie instaluje się wszystkie pakiety z którymi jest on powiązany.

Przykładem takiego pakietu może być **task-samba**, który zależy od pakietów: **samba**, **samba-doc**, **smbclient**, **swat**, **smbfs**. W ten sposób instalując **task-samba** instaluje się wszystkie pakiety związane z Sambą, co jest niezwykle wygodne.

Przykładem innego pakietu może być **task-polish**, zawierający polskie czcionki, dokumentację i słowniki. Innym przykładem mogą być **task-x-window-system** zawierającym najważniejsze aplikacje X Window, czy pakiety **task-gnome-xxx**, **task-kde-xxx** zawierające powiązania ze środowiskami GNOME czy KDE.

Pakiety typu **task** można instalować nie tylko poprzez APT, czy Dselect. Jest również narzędzie **tasksel** pozwalające na przeglądanie wszystkich tego typu pakietów i ich instalację. Jest ono przydatne w trakcie administrowania, a także wstępnej instalacji systemu.

4. Pakiety źródłowe DEB

Wszystkie pakiety DEB, instalowane przy pomocy **dpkg** i **dselect** są pakietami binarnymi, lub niezależnymi od platformy. Pakiety binarne (na przykład z katalogu **binary-i386**) zawierają kod wykonywalny na konkretnej platformie, zaś pakiety niezależne od platformy (z katalogu **binary-all**) mogą zawierać na przykład skrypty w języku Perl.

Pakiety DEB powstają z tak zwanych pakietów źródłowych. Pakiet źródłowy Debiana zawiera kod źródłowy programu, oraz informacje o sposobie jego kompilacji i integracji z dystrybucją Debian/GNU.

Pakiet źródłowy w Debian/GNU składa się z następujących plików:

- **nazwa_wersjaprogramu-wersjapakietu.orig.tar.gz** – zawiera pełny, oryginalny kod źródłowy programu,
- **nazwa_wersjaprogramu-wersjapakietu.diff.gz** – zawiera informacje kontrolne pakietu oraz wszystkie różnice wprowadzone przez twórcę pakietu do kodu programu,
- **nazwa_wersjaprogramu-wersjapakietu.dsc** – zawiera informacje o pakiecie, wraz z informacjami o zależnościach pakietu, oraz sumy MD5 dwóch powyższych plików wraz z podpisem GPG/PGP autora pakietu.

Jak widać pakiet źródłowy składa się z trzech oddzielnych plików, w przeciwieństwie do pakietu źródłowego RPM, czyli SRPM. W praktyce, w pakiecie SRPM znajdują się informacje tego samego typu, co w pakiecie źródłowym Debiana. To czy wygodniej jest mieć te dane w postaci jednego, czy trzech plików jest sprawą dyskusyjną. Z jednej strony prościej jest kopiować pojedyncze pliki SRPM. Z drugiej strony pakiet źródłowy można rozpakować i modyfikować bez specjalnych narzędzi, na przykład przy pomocy standardowych narzędzi typu **patch**.

Czasami w pakiecie źródłowym obecne są tylko pliki **.dsc** i **.tar.gz** (zamiast **.orig.tar.gz**), nie ma natomiast pliku **.diff.gz**. Taka sytuacja ma miejsce w przypadku programów napisanych przez członków projektu Debian, lub programów, które nie były w żaden sposób modyfikowane w trakcie tworzenia pakietu.

4.1. Rozpakowywanie pakietów źródłowych

Do operacji na pakietach źródłowych służą narzędzia z rodziny **dpkg-source**, znajdujące się w pakiecie **dpkg-dev**. W celu rozpakowania pakietu źródłowego należy użyć programu **dpkg-source**

z opcją `-x` (ang. *extract*). Jako argument należy podać nazwę pliku `.dsc`. Na przykład:

```
$ dpkg-source -x bash_2.01.1-4.1.dsc
dpkg-source: extracting bash in bash-2.01.1
```

Spowoduje to rozpakowanie pakietu do katalogu `nazwaprogramu-wersjaprogramu`, w tym przypadku `bash-2.01.1`. Użycie powyższego polecenia z dodatkową opcją `-su` spowoduje dodatkowo stworzenie katalogu z oryginalnym kodem źródłowym programu, w tym wypadku `bash-2.01.1.orig`.

4.2. Informacje kontrolne pakietu

Kod źródłowy pakietu różni się od oryginalnego kodu programu głównie katalogiem `debian`, zawierającym wszystkie informacje kontrolne pakietu. Oprócz tego, sam kod programu jest czasami modyfikowany przez autora pakietu, na przykład w celu nałożenia dodatkowych łat.

Zawartość katalogu zależy od konkretnego programu. Chociaż katalog może zawierać wiele różnych plików, zawsze są to przynajmniej pliki:

- `changelog` – zawiera informacje o zmianach jakie były wprowadzane do pakietu przez jego twórcę,
- `conffiles` – spis plików konfiguracyjnych pakietu, przydatny podczas instalacji i deinstalacji pakietu (na przykład polecenie `dpkg -r` nie odinstalowuje wymienionych tu plików),
- `control` – jeden z najważniejszych plików, zawiera opis pakietu oraz informacje o części dystrybucji do jakiej należy pakiet, jego priorytecie, twórcy i zależnościach,
- `copyright` – informacje o licencji programu i prawach autorskich jego twórcy,
- `rules` – plik wykonywalny służący do budowania różnych wersji pakietu binarnego DEB.

Oprócz wymienionych powyżej, często obecne są pliki:

- `preinst`, `postinst`, `prerm`, `postrm` – opisywane poprzednio pliki używane podczas instalowania i odinstalowywania pakietu; nie wszystkie z nich muszą występować,
- `README.debian` – plik zawierający informacje na temat istotnych z punktu widzenia integracji z Debian/GNU cech programu lub pakietu,
- `shlibs` – spis bibliotek dzielonych używanych przez pakiet.

4.3. Budowanie pakietów binarnych

Po rozpakowaniu pakietu źródłowego można z niego stworzyć pakiet binarny DEB. Istnieje wiele możliwości budowania pakietów binarnych, ponieważ system pakietów jest bardzo uniwersalny. Można między innymi budować wersje na różne platformy sprzętowe, lub niezależne od platformy.

Budowanie pakietu źródłowego jest rozpoczynane przez uruchomienie pliku `rules` z odpowiednim argumentem. Najważniejsze możliwości jakie są dostępne to:

- `build` – kompilacja kodu źródłowego programu,
- `clean` – usunięcie plików wynikowych po powyższej,
- `binary-indep` – zbudowanie niezależnej od platformy części pakietu,

- **binary-arch** – zbudowanie binarnej części pakietu,
- **binary** – zbudowanie całości pakietu (pakietów), to znaczy dwóch powyższych części.

Tu należy zauważyć, że z jednego pakietu źródłowego może być stworzonych kilka pakietów binarnych. Dzieje się tak na przykład w przypadku bibliotek, lub większych aplikacji. Przykładowo z pakietu źródłowego `libgtk1.2_1.5.5-1.dsc` są tworzone trzy pakiety binarne: `libgtk1.2-dev_1.2.1-1.deb`, `libgtk1.2-doc_1.2.1-1.deb`, `libgtk1.2_1.2.1-1.deb`, zawierające odpowiednio: pliki konieczne do kompilacji programów używających Gtk+ (*include*), dokumentację biblioteki, oraz pliki potrzebne do uruchamiania aplikacji (tzw. *runtime*). Natomiast program Gimp jest dzielony na części: `gimp`, `libgimp1` oraz `gimp-nonfree`. Ten ostatni pakiet zawiera części Gimp'a, które nie mogą znaleźć się w głównej części dystrybucji Debian/GNU ze względu na ograniczenia licencyjne. W tym przypadku chodzi o ograniczenia wynikające z patentu na algorytm kompresji stosowany w formacie GIF.

Przykład rozpakowywania pakietu źródłowego `gmp1_1.3.2-8.dsc` i budowanie pakietu binarnego przebiega następująco:

```
# dpkg-source -x gmp1_1.3.2-8.dsc
dpkg-source: extracting gmp1 in gmp1-1.3.2
# cd gmp1-1.3.2
# debian/rules binary
... (kompilacja)
... (tworzenie pakietu)
dpkg-deb: building package 'gmp1' in '../gmp1_1.3.2-8_i386.deb'.
... (kompilacja)
... (tworzenie pakietu)
dpkg-deb: building package 'gmp1-dev' in '../gmp1-dev_1.3.2-8_i386.deb'.
```

Jak widać w przypadku tej biblioteki powstają dwa pakiety binarne, zawierające pliki typu *runtime* i *include*.

Na koniec nasuwa się podstawowe pytanie: kiedy administrator systemu może być zainteresowany rekompilacją pakietów binarnych z pakietów źródłowych? Możliwości jest rzecz jasna wiele. Pomijając sytuację najbardziej oczywistą, kiedy administrator-programista chce zmodyfikować kod źródłowy programu, należy wskazać na sytuację, gdy administrator chce mieć dostęp do nowszej wersji programu, a nie może z różnych przyczyn zainstalować pakietu binarnego. Taka sytuacja może mieć miejsce na przykład gdy w systemie jest zainstalowana wersja X programu A, podczas, gdy pojawiła się już nowsza wersja Y tego programu. Wersja Y jest dostępna jako pakiet źródłowy, oraz pakiet binarny. Jeżeli jednak pakiet binarny został zkompilowany z nowszymi wersjami bibliotek, niż są zainstalowane w systemie,

Inna tego typu sytuacja ma miejsce, jeżeli chce się mieć dostęp do pewnych plików, lub informacji, które są obecne w pakiecie źródłowym, a zostały pominięte w trakcie tworzenia pakietu binarnego. Przykładowo, dokumentacja do projektu GNU jest tworzona w formacie TeXinfo. W trakcie tworzenia pakietu binarnego z plików źródłowych `.texi` tworzone są pliki w formacie `.info`. Jeżeli jednak administrator chce wydrukować wersje PostScript, potrzebuje dokonać konwersji plików `.texi` do PostScript. W tym celu może sięgnąć do pakietu źródłowego Debiana.

Kolejną możliwością jest optymalizacja pakietu dla konkretnej architektury. Debian jest standardowo dostępny w wersji i386 na platformę Intel x86 i M68000 na platformę Motorola M68k. Jest tak między innymi dlatego, że rekompilacja całej dystrybucji (aktualnie ponad 4000 pakietów!) nie ma sensu i niewiele daje. Tym niemniej można sobie wyobrazić, że chce się zrekompilować konkretny pakiet, na przykład z optymalizacją dla procesorów Pentium.

W takim przypadku można odpowiednio skonfigurować kompilator, lub nawet zainstalować odpowiedni pakiet Debiana, *pentium-builder*, i w trakcie pobierania pakietu przez APT nakazać mu automatyczną rekompilację.

5. Podsumowanie

W artykule zaprezentowano najważniejszy interfejs do systemu pakietów Debian/GNU, jakim jest program *dselect*. Jest to narzędzie bardzo zaawansowane, posiadające ogromne możliwości. Zdarza się jednak, że początkujących użytkowników zraża jego interfejs. Tymczasem, to właśnie ten interfejs, który jest bardzo zoptymalizowany pod kątem szybkości działania, wydajności pracy i ilości prezentowanych informacji stanowi o sile tego narzędzia.

W ostatnim artykule zostanie przedstawione narzędzie Alien pozwalające na instalowanie pakietów innych dystrybucji oraz system APT będący następną generacją narzędzi do zarządzania pakietami w systemie operacyjnym Debian/GNU.