APT HOWTO (Obsolete Documentation)

Gustavo Noronha Silva <kov@debian.org>
український переклад: Borys Yanovych <borman@univ.kiev.ua>

1.8.11 - серпень 2005

Анотація

Цей документ створено, що допомогти користувачу краще зрозуміти роботу інструменту керування пакунками Debian, APT. Його мета — полегшити життя новим користувачам Debian та допомогти тим, хто бажає поглибити свої знання у адмініструванні цієї системи. Документ було створено для проекту Debian з метою поліпшення підтримки цього дистрибутиву.

Авторські права

Зміст

Розділ 1

Вступ

Спочатку був .tar.gz. Користувачі повинні були компілювати кожну програму, котру вони хотіли використовувати на власних системах GNU/Linux. Коли створювався Debian, виникла потреба у включенні до системи інструменту керування пакунками, встановленими на машині. Його назвали dpkg. Цей знаменитий „пакунок“ першим з'явився в GNU/Linux ще до того, як Red Hat вирішила створити свою власну систему rpm.

Досить швидко нова дилема постала в думках розробників GNU/Linux. Їм був потрібен швидкий, практичний та ефективний шлях для встановлення пакунків, з автоматичною обробкою залежностей та збереженням конфігураційних файлів під час оновлення. Знову ж таки, Debian виявився лідером та породив систему APT, Advanced Packaging Tool, котра в подальшому була портована фірмою Conectiva для використання з rpm та пристосована також до деяких інших дистрибутивів.

В цьому підручнику не описується apt-rpm (портування Conectiva відоме саме під такою назвою), але „латки“ до цього документу з цього приводу завжди радо вітаються.

Цей підручник базується на випуску Debian Sarge.

Розділ 2

Базові налаштування

2.1 Файл /etc/apt/sources.list

Щоб забезпечити виконання однієї зі своїх задач, APT використовує список джерел, з котрих пакунки можуть бути завантажені. Це — файл /etc/apt/sources.list.

Зазвичай, записи в цьому файлі мають такий вигляд:

```
deb http://host/debian збірка розділ1 розділ2 розділ3
deb-src http://host/debian збірка розділ1 розділ2 розділ3
```

Звісно, наведені вище рядки є лише прикладом і не повинні використовуватись насправді. Перше слово кожного рядка, deb чи deb-src, вказує тип архіву: це або двійкові пакунки (deb), наперед-скомпільовані, які ми, як правило, і використовуємо, або джерельні пакунки (deb-src), котрі містять оригінали джерельних кодів програм і, додатково, керуючий файл Debian (.dsc) та diff.gz-архів з необхідними для „дебіанізації“ програми змінами.

Зазвичай, в стандартному файлі sources.list можна знайти такі рядки:

```
# 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 non-free
```

Ці рядки потрібні для базового встановлення Debian. Перший рядок deb вказує на офіційний архів пакунків, другий — на не-США архів, третій — на архів Debian, що призначений для оновлень безпеки.

Два останні рядки закоментовано (починаються з символу '#'), тому apt-get їх проігнорує. Ці deb-src рядки вказують на джерельні пакунки Debian. Якщо ви часто завантажуєте джерельні коди програм, з метою тестування або для рекомпіляції, просто розкоментуйте їх.

В файлі /etc/apt/sources.list можуть міститись декілька типів рядків. Наскільки мені відомо, APT вміє працювати з такими типами архівів як http, ftp, file (локальні файли; наприклад, тека з примонтованою файловою системою ISO9660) та ssh.

Не забудьте виконати apt-get update після внесення змін до файлу /etc/apt/sources.list. Це робиться для того, щоб змусити APT оновити списки пакунків з вказаних вами джерел.

2.2 Як використовувати APT локально

Інколи може виникнути ситуація, коли ви маєте купу пакунків .deb і хотіли б використати APT для їх встановлення, щоб залежності оброблялись автоматично.

Для цього вам потрібно створити теку та помістити в неї .deb-файли, які ви хочете проіндексувати. Наприклад:

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

Для вашого сховища ви можете напряму змінити набір значень, вказаних в файлі control кожного з пакунків, використовуючи файл override. Всередині цього файлу ви, якщо хочете, можете визначити деякі опції, котрі перекриють відповідні значення з файлів пакунків. Це виглядає приблизно так:

```
package priority section
```

package — це назва пакунка, priority може набувати таких значень як low, medium чи high, a section — це назва розділу, в якому знаходиться відповідний пакунок. До назви файлу не ставиться жодних вимог, ви будете вказувати її пізніше в якості аргументу до dpkg-scanpackages. Якщо ви не хочете створювати файл override, тоді при запуску dpkg-scanpackages просто використовуйте /dev/null.

Далі, перебуваючи в теці /root, виконайте:

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

У наведеному рядку слово file — це файл override. Команда генерує файл Packages.gz, котрий містить різноманітну інформацію про пакунки, яка буде використана APT. Нарешті, щоб почати використання пакунків, додайте:

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

Після всього цього, просто використовуйте APT, як звичайно. Ви можете створювати також і джерельні сховища. Для цього повторіть ті ж самі кроки, але пам'ятайте, що вам потрібно у відповідній теці мати файли .orig.tar.gz, .dsc та .diff.gz і замість Packages.gz генерувати файл Sources.gz. Для цього також використовується інша програма. Це — dpkg-scansources. Командний рядок повинен виглядати приблизно так

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

Зверніть увагу, що для dpkg-scansources не потрібен файл override. Рядок для sources.list виглядає так:

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

2.3 Визначення найкращого джерела для використання в файлі sources.list: netselect, netselect-apt

Дуже часто, переважно від нових користувачів, можна почути питання: „яке дзеркало Debian включати до sources.list?“. Існує багато методів визначення такого дзеркала. Експерти, ймовірно, мають скрипт, що вимірює та порівнює час відгуку від декількох джерел. Однак є програма, що зробить це і для нас: netselect.

Як звичайно, щоб встановити netselect:

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

При запуску її без параметрів відображається довідка. З списком дзеркал, розділених пробілами, вона поверне оцінку і один з вузлів. При визначенні цієї оцінки береться до уваги приблизний час відгуку, кількість „стрибків“ (вузлів, через які мережевий запит пройде перед тим, як потрапити до заданої цілі) та обернено пропорційна швидкість завантаження; отже, чим менше оцінка, тим краще. Програма повертає назву дзеркала, що має найнижчу оцінку (повний список оцінок можна побачити, додавши опцію -vv). Ознайомтесь з таким прикладом:

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

Це означає, що з дзеркал, вказаних в якості параметрів до netselect, найкращим виявилось ftp.debian.org.br з оцінкою 365. (Увага!! Оскільки команда виконувалась на моєму комп'ютері, а топографія мережі дуже сильно відрізняється в залежності від точки входу, ця цифра не обов'язково буде правильним значенням швидкості для інших машин).

Тепер просто помістіть найшвидше дзеркало, знайдене за допомогою netselect, у файл /etc/apt/sources.list (див. 'Файл /etc/apt/sources.list' на стор. 3) та виконуйте наведені у відповідному розділі вказівки (див. 'Керування пакунками' на стор. 9).

Зверніть увагу: список дзеркал завжди можна знайти в файлі http://www.debian.org/mirror/mirrors_full.

Починаючи з версії 0.3.ds1, джерельний пакунок netselect створює двійковий пакунок netselect-apt, котрий виконує всі наведені вище дії автоматично. Просто введіть гілку збірки в якості параметру (за замовчанням stable) і програма згенерує файл sources.list з найкращими загальними та не-США дзеркалами, після чого збереже його в поточній теці. Наступний приклад генерує sources.list для стабільної збірки:

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

Нагадуємо: файл sources.list генерується в поточній теці і його потрібно перемістити до теки /etc/apt.

Далі виконуйте вказівки наведені у відповідному розділі (див. 'Керування пакунками' на стор. 9).

2.4 Додавання CD-ROM'у до файлу sources.list

У випадку, коли ви використовуєте переважно CD-ROM для встановлення пакунків чи оновлення вашої системи за допомогою APT, ви можете додати його до вашого sources.list. Для цього можна скористайтися програмою apt-cdrom приблизно так:

```
# apt-cdrom add
```

попередньо помістивши компакт-диск Debian до приводу. Ця команда підмонтує КД і, якщо це справжній компакт-диск Debian, оновить інформацію про пакунки, переглянувши його вміст. Якщо ваш КД дещо незвичної конфігурації, ви також можете використати такі опції:

```
-h          - Довідка програми
-d directory - Точка монтування CD-ROM'у
-r          - Змінити назву розпізнаного CD-ROM'у
-m          - Не монтувати
-f          - Швидкий режим, не перевіряти файли пакунків
-a          - Режим ретельного пошуку
```

Наприклад:

```
# apt-cdrom -d /home/kov/mycdrom add
```

Ви також можете ідентифікувати CD-ROM, не додаючи його до вашого списку:

```
# apt-cdrom ident
```

Зауважте, що ця програма працює лише в тому випадку, коли ваш КД зконфігуровано правильно і відповідний запис міститься в системному файлі /etc/fstab.

Розділ 3

Керування пакунками

3.1 Оновлення списку наявних пакунків

Система керування пакунками використовує свою власну базу даних для відслідковування вже встановлених, не встановлених та доступних для встановлювання пакунків. Програма apt-get використовує цю базу даних як джерело інформації про пакунки, вказані користувачем в запиті на встановлення, та з'ясування, які додаткові пакунки є необхідними, щоб вибраний пакунок працював належним чином.

Для оновлення цієї бази ви можете скористатись командою apt-get update. Ця команда переглядає списки пакунків в архівах, вказаних в /etc/apt/sources.list; див. 'Файл /etc/apt/sources.list' на стор. 3, щоб дізнатись більше про цей файл.

Було б непогано запускати цю команду регулярно, щоб ви та ваша система були проінформованими щодо можливих оновлень пакунків і, особливо, оновлень їх безпеки.

3.2 Встановлення пакунків

Нарешті, те, чого всі так довго чекали! Коли ваш sources.list готовий і список доступних пакунків оновлено, вам залишається лише запустити apt-get, щоб встановити бажаний пакунок. Наприклад, ви можете виконати:

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

APT спробує знайти в базі даних найсвіжішу версію цього пакунку та завантажити його з відповідного архіву, визначеного в sources.list. У випадку, коли цей пакунок залежить від іншого — як наведено нижче — APT перевірить залежності та встановить всі потрібні пакунки. Наприклад:

```
# 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]
```

Пакунок nautilus залежить від вказаних бібліотек спільного користування, тому APT також завантажить їх з архіву. Якщо ви вкажете їх назви в командному рядку при запуску apt-get, APT не буде запитувати, чи хочете ви продовжувати, а автоматично прийме рішення, що ви бажаєте встановити всі ті пакунки.

Це означає, що APT потребує підтвердження лише в тому випадку, коли потрібно встановити пакунки, які не були згадані в командному рядку.

Наступні параметри apt-get можуть стати в нагоді:

```
-h  Текст довідки.
-d  Тільки завантажити~— НЕ встановлювати і НЕ розпаковувати пакунки
-f  Продовжувати, навіть якщо перевірка цілісності завершилась невдачею
-s  Не виконувати реальних дій, імітувати їх виконання
-y  Вважати Yes відповіддю на всі запитання, не виводити їх
-u  Також показувати список оновлених пакунків
```

В одному рядку можна вказувати декілька пакунків для встановлення. Завантажені з мережі файли поміщаються в теку /var/cache/apt/archives для подальшого встановлення.

Також, в тому ж командному рядку ви можете вказати пакунки, які потрібно видалити. Просто помістіть знак '-' одразу ж після назви обраного для видалення пакунка, наприклад:

```
# 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]
```

Перегляньте ‘Видалення пакунків’ на стор. 11, щоб детальніше ознайомитись з видаленням пакунків.

Якщо ви якимсь чином пошкодили встановлений пакунок чи просто хочете перевстановити файли з пакунка більш нової доступної версії, можете скористатись параметром —reinstall, ось так:

```
# 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 upgraded.
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

3.3 Видалення пакунків

Якщо ви більше не бажаєте використовувати пакунок, ви можете видалити його з вашої системи за допомогою APT. Для цього просто наберіть apt-get remove пакунок. Наприклад:

```
# 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]
```

Як можна побачити в прикладі вище, APT також турбується про видалення тих пакунків, які залежать від пакунка, вибраного для видалення. Видалити пакунок, не видаляючи при цьому тих пакунків, які залежать від нього, за допомогою APT неможливо.

Запуск apt-get так, як показано вище, видалить пакунки, однак їх конфігураційні файли, якщо такі існують, залишаться в системі неушкодженими. Для повного видалення пакунку виконайте:

```
# 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]
```

Зверніть увагу на символ '*' після назв пакунків. Це вказує на те, що конфігураційні файли кожного з цих пакунків також будуть видалені.

Як і у випадку методу install, ви можете скористатись спеціальним символом при remove, щоб інвертувати його дію для конкретного пакунка. У випадку видалення, якщо ви додаєте знак '+' одразу після назви пакунка, то такий пакунок буде встановлений, а не видалений.

```
# 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]
```

Зауважте, що apt-get виводить список додаткових пакунків, котрі будуть встановлені (таких, чиє встановлення є необхідною умовою для правильного функціонування пакунків, вказаних у початковому запиті на встановлення), пакунків, котрі будуть видалені, та загалом пакунків, котрі будуть встановлені (включаючи і додаткові пакунки також).

3.4 Оновлення пакунків

Оновлення пакунків — визначне досягнення системи APT. Це можна робити за допомогою однієї-єдиної команди: apt-get upgrade. Ви можете використовувати її і для оновлення пакунків з однієї і тієї ж збірки, і у випадку переходу до збірки нової, хоча, для останнього рекомендується використовувати apt-get dist-upgrade (див. 'Оновлення до нового випуску' на стор. 13, щоб дізнатись про це більше).

Доцільно запускати цю команду з опцією -u. За її використання APT виводить повний список пакунків, котрі будуть оновлені. Без неї ви будете оновлюватись всліпу. APT завантажить останні версії кожного з пакунків та встановить їх в правильному порядку. Перед оновленням пакунків завжди важливо виконувати apt-get update. Перегляньте 'Оновлення списку наявних пакунків' на стор. 9. Наприклад:

```
# 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 procps 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]
```

Процес є дуже простим. Відзначте, що в одному з перших рядків apt-get вказує на те, що деякі пакунки були kept back. Це означає, що є нові версії цих пакунків, але вони не можуть бути встановлені з деяких причин. Можливими причинами можуть бути неправильні залежності (вказану версію пакунка, від якого залежить даний, неможливо завантажити) або нові залежності (остання версія даного пакунка залежить від нового пакунка).

Для першого випадку не існує прозорого вирішення проблеми. В другому випадку доцільно запустити apt-get install, вказавши відповідний пакунок, що призведе до завантаження його залежностей. Більш прозорим рішенням буде використання dist-upgrade. Перегляньте ‘Оновлення до нового випуску’ на стор. 13.

3.5 Оновлення до нового випуску

Ця можливість APT дозволяє вам оновлювати одразу всю систему Debian, або через Інтернет, або з нового компакт-диску (придбаного чи завантаженого як ISO-образ).

Це також використовується, коли зміни торкаються взаємозв’язків між встановленими пакунками. При використанні apt-get upgrade такі пакунки залишились би незайманими (kept back).

Наприклад, припустимо, що ви використовуєте редакцію 0 стабільної версії Debian та купуєте компакт з редакцією 3. Ви можете скористатись APT для оновлення вашої системи з нового КД. Для цього скористайтесь apt-cdrom (див. ‘Додавання CD-ROM’у до файлу sources.list’ на стор. 6), щоб додати ваш КД до файлу /etc/apt/sources.list і запустіть apt-get dist-upgrade.

Важливо відзначити, що APT завжди шукає найсвіжіші версії пакунків. Тому, якщо до вашого списку в /etc/apt/sources.list було включено архів з пакунками більш нових версій, ніж на компакт-диску, APT буде завантажувати пакунки звідти.

В прикладі в одному з попередніх розділів (див. ‘Оновлення пакунків’ на стор. 12) ми бачили, що деякі пакунки були kept back. Зараз ми вирішимо цю проблему за допомогою методу 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
  procps 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]
```

Зверніть увагу, що пакунки буде оновлено і, крім цього, також буде встановлено нові пакунки (нові залежності). Зауважте також, що lilo все ще залишається kept back. Він, ймовірно, має серйознішу проблему, ніж нова залежність. Ми можемо її з'ясувати, виконавши:

```
# 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]
```

Як з'ясувалось, lilo конфліктує з пакунком debconf-tiny, а отже не може бути встановленим (чи оновленим) без видалення debconf-tiny.

Щоб дізнатись, яким чином приймається рішення про збереження або видалення пакунка, спробуйте:

```
# 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.
```

Таким чином, легко помітити, що пакунок python1.5-dev не може бути встановленим через незадоволену залежність: python1.5.

3.6 Видалення непотрібних файлів пакунків: apt-get clean та autoclean

Коли ви встановлюєте пакунок, APT завантажує потрібні файли з джерел, вказаних в /etc/apt/sources.list, записує їх в локальне сховище (/var/cache/apt/archives/) і потім встановлює, як описано вище (див. ‘Встановлення пакунків’ на стор. 9).

З часом локальне сховище може вирости і зайняти багато дискового простору. На щастя, APT забезпечує інструменти для керування локальним сховищем: методи apt-get — clean та autoclean.

apt-get clean видаляє все, за виключенням lock-файлів з /var/cache/apt/archives/ і /var /cache/apt/archives/partial/. Тому, у випадку перевстановлення пакунка, APT повинен буде завантажувати його знову.

apt-get autoclean — тільки ті файли пакунків, як більше не можна завантажити.

Наступний приклад демонструє роботу apt-get:

```
# 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
```

В теці /var/cache/apt/archives є два файли пакунка logrotate та один файл пакунка 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
```

apt-show-versions показує, що logrotate_3.5.9-8_i386.deb забезпечує поточну версію logrotate, отже logrotate_3.5.9-7_i386.deb більше не потрібен. Також gpm_1.19.6-11_i386.deb є непотрібним, оскільки є можливість завантажити більш нову версію цього пакунка.

```
# 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]
```

Отже, apt-get autoclean видаляє тільки застарілі файли. Щоб отримати більше інформації про apt-show-versions перегляньте ‘Яким чином оновлювати пакунки зі специфічних версій Debian’ на стор. 18.

3.7 Використання APT з dselect

dselect — це програма, котра допомагає користувачам обирати пакунки Debian для встановлення. Вона виглядає дещо громіздкою і трохи нудною, але з часом ви можете звикнути до її термінального, основаного на ncurses, інтерфейсу.

Однією з особливостей dselect є її вміння враховувати те, що пакунки Debian можуть „рекомендувати“ і „пропонувати“ інші пакунки під час встановлення. Для використання програми запустіть dselect в якості користувача root. Оберіть apt в якості методу доступу. Насправді, це не обов’язково, але якщо ви не використовуєте компакт-диски і хочете завантажувати пакунки з Інтернету, це найкращий спосіб використання dselect.

Щоб краще зрозуміти, як працює dselect, прочитайте відповідну документацію. Її можна знайти на сторінці Debian http://www.debian.org/doc/ddp.

Після того, як ви за допомогою dselect вибрали пакунки, використовуйте

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

як показано в прикладі нижче:

```
# 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]
```

Порівняйте це з виводом apt-get dist-upgrade на цій же системі:

```
# 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]
```

Зауважте, що багато пакунків з наведеного вище прикладу були встановлені тому, що інші пакунки „пропонують“ або „рекомендують“ їх. Інші були встановлені або видалені (наприклад, lbxproxy) в результаті вибору, який ми зробили за допомогою інтерфейсу dselect. Dselect може бути потужним інструментом, якщо використовується разом з APT.

3.8 Як підтримувати змішану систему

Іноді люди бажають використовувати одну з версій Debian в якості головного дистрибутиву системи та один чи декілька пакунків з іншої гілки.

Щоб вказати, яка версія Debian є головною, необхідно відредагувати файл /etc/apt/apt.conf (зазвичай його немає, в такому випадку створіть його), помістивши в нього такий рядок:

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

Де версія — це версія Debian, котру ви хочете використовувати в якості головної збірки. Можна вказувати такі значення як stable, testing та unstable. Далі, щоб встановити пакунки з інших версій, ви повинні використовувати APT в такий спосіб

```
# apt-get -t збірка install пакунок
```

Проте, що це працювало, вам потрібно мати як мінімум один рядок в /etc/apt/sources.list з джерелом APT для відповідної збірки і на цьому джерелі повинні бути необхідні пакунки.

Ви також можете здійснити запит щодо конкретної версії пакунка, скориставшись наступним синтаксисом:

```
# apt-get install пакунок=версія
```

Наприклад, при виконанні вказаної нижче команди встановиться пакунок nautilus з версією 2.2.4-1.

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

ВАЖЛИВО: збірка unstable — це збірка, до якої найновіші версії пакунків Debian потрапляють в першу чергу. Через цю збірку проходять всі ті зміни, котрих зазнають пакунки. І невеликі пакунки, і дуже важливі, від роботи яких залежать інші пакунки або система в цілому. З цієї причина, ця версія збірки не повинна використовуватись недосвідченими користувачами або тими, хто ставить стабільність системи на перше місце.

Збірка testing не обов'язково краща за unstable, оскільки не отримує оновлень безпеки за короткий час. Для серверів та інших виробничих систем завжди необхідно використовувати stable.

3.9 Яким чином оновлювати пакунки зі специфічних версій Debian

apt-show-versions забезпечує користувачів змішаних збірок безпечним способом для оновлення їх системи без отримання з менш стабільної збірки більшого, ніж вони мали на увазі. Наприклад, можна оновити лише ваші пакунки з unstable, виконавши, попередньо встановивши пакунок apt-show-versions,

```
# apt-get install ‘apt-show-versions -u -b | grep unstable | cut -d ’ ’ -f 1‘
```

3.10 Як зберегти встановленими конкретні версії пакунків (складний метод)

Можливо, сталось так, що ви змінили дещо в пакунку, але не маєте часу або не хочете вносити ці зміни до нової версії програми. Чи, наприклад, вам просто потрібно оновити вашу збірку Debian до 3.0, але є бажання продовжувати використовувати деякі версії пакунків зі збірки Debian 2.2. Ви можете „приколоти“ (pin) версію, котру ви використовуєте так, щоб вона більш не оновлювалась.

Використовувати цю можливість просто. Необхідно лише відредагувати файл /etc/apt/preferences.

Формат простий:

```
Package: <package>
Pin: <pin-означення>
Pin-Priority: <pin-пріоритет>
```

Кожен запис повинен відділятись від інших порожнім рядком. Наприклад, щоб зберегти версію 0.4.99 пакунка sylpheed, в котрий я додав можливість reply-to-list, я вказую:

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

Зверніть увагу, що я використав символ * (зірочка). Це „маска“, вона означає, що я хочу, щоб цей pin стосувався всіх версій, що починаються з 0.4.99. Це зроблено з тієї причини, що до версії в пакунках Debian додається номер „редакції Debian“ і я не хочу відміняти встановлення цих редакцій. Отже, наприклад, версії 0.4.99-1 та 0.4.99-10 будуть встановлені, як тільки стануть доступними. Зауважте, що якщо ви модифікували пакунок, такий метод вам не підійде.

Pin-пріоритет допомагає визначити чи оновлювати пакунок, що задовольняє вимогам в рядках Packages: та Pin:, вищий пріоритет збільшує шанси того, що відповідний пакунок буде встановлено. Ви можете прочитати apt_preferences(7), щоб ознайомитись зі всебічним обговоренням пріоритетів, але декілька прикладів допоможуть зрозуміти основну ідею. Далі описуються ефекти від встановлення різних значень поля Pin-Priority: для пакунку sylpheed з наведеного вище прикладу.

1001 Apt ніколи не замінить sylpheed, версії 0.4.99. Якщо буде потрібно, apt встановить версію 0.4.99, замінивши нею пакунок з більш високою версією. Тільки пакунки з пріоритетом, вищим ніж 1000 зможуть „понизити“ існуючий пакунок.

1000 Ефект такий самий, як і у випадку пріоритету 1001, за винятком того, що apt не буде знижувати версію вже встановленого пакунка до 0.4.99.

990 Версію 0.4.99 зможе замінити лише пакунок вищої версії зі збірки, котра визначена як переважаюча, шляхом використання змінної APT::Default-Release (див. ‘Як підтримувати змішану систему’ на стор. 17 вище).

500 Будь-яка доступна версія, вища за 0.4.99, пакунка sylpheed з будь-якого випуску буде мати перевагу над версією 0.4.99, але остання буде мати перевагу над нижчими версіями.

100 Вищі доступні версії sylpheed з будь-якої збірки будуть мати перевагу над версією 0.4.99, також будь які встановлені вищі версії slypheed; тобто 0.4.99 встановиться лише в тому випадку, якщо не встановлено жодної іншої версії. Це — пріоритет встановлених пакунків.

-1 Від'ємні значення також є прийнятними і забороняють встановлення 0.4.99.

„Приколювання“ пакунка може здійснюватись за такими критеріями як його версія (version), випуск (release) чи походження (origin) пакунка.

Фіксування за версією (version), як ми бачили, підтримує літерні номери версій і, крім цього, „маски“ з можливістю вибору декількох версій одночасно.

Опція release залежить від файлу Release зі сховища APT або з компакт-диску. Ця опція може не використовуватись у тому випадку, коли всі ваші сховища пакунків такого файлу не забезпечують. Вміст ваших файлів Release можна переглянути, відвідавши теку /var/lib/apt/lists/. Параметрами до release є: a (archive, архів), c (components, компоненти), v (version, версія), o (origin, походження) та l (label, мітка).

Наприклад:

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

В цьому прикладі ми обираємо версію Debian 2.2* (що може бути 2.2r2, 2.2r3 — це „точковий випуск“, що, зазвичай, містить виправлення безпеки та інші дуже важливі оновлення), сховище stable, розділ main (на противагу contrib чи non-free) та походження і мітку Debian. Походження (o=) визначає, хто створив цей файл Release, мітка (l=) — назва дистрибутиву: Debian для власне Debian, Progeny для Progeny, і т.д. Приклад файлу Release:

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i386_Release
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

Розділ 4

Дуже корисні помічники

4.1 Як встановлювати локально скомпільовані пакунки: equivs

Іноді людина хоче використовувати специфічну версію програми, доступну лише у вигляді джерельних кодів, а не пакунку Debian. Але пакункова система може стати для цього певною перешкодою. Припустимо, ви хочете скомпілювати нову версію вашого серверу електронної пошти. Все б чудово, але багато пакунків в Debian залежать від MTA (Mail Transport Agent). Однак, оскільки ви встановили дещо самостійно вами скомпільоване, система керування пакунками не знатиме про його існування.

І тут на сцену виходить equivs. Щоб скористатись ним, встановіть однойменний пакунок. Equivs створює порожній пакунок з повним набором залежностей; робить так, щоб у системи керування пакунками склалось враження, що всі залежності задовольняються.

Перед тим, як ми почнемо, непогано було б нагадати вам, що є і більш безпечні шляхи компіляції з різноманітними параметрами тих програм, для котрих вже існують пакунки Debian, і що не потрібно використовувати equivs для заміни залежностей, якщо ви не не впевнені в тому, що робите. Щоб дізнатись більше перегляньте ‘Робота з джерельними пакунками’ на стор. 31.

Отже, повернемось до нашого прикладу з MTA. Ви щойно встановили ваш наново скомпільований postfix і переходите до встановлення mutt. Раптом виявляється, що mutt вимагає встановити інший MTA. Але у вас вже є ваш власний.

Перейдіть до якоїсь теки (наприклад, /tmp) і запустіть:

```
# equivs-control name
```

Замініть name на назву керуючого файлу, який ви хочете створити. Він буде мати такий вигляд:

```
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
```

Нам просто потрібно привести його до потрібного нам вигляду. Погляньмо на формат полів і їх описи. Очевидно, нема потреби пояснювати кожне з них окремо, давайте просто зробимо те, що потрібно:

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

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

Так, це все. mutt залежить від віртуального пакунка mail-transport-agent, котрий забезпечується будь-яким MTA, я міг би просто назвати пакунок mail-transport-agent, однак вважаю за краще використовувати схему віртуальних пакунків за допомогою Provides.

Тепер треба просто зібрати пакунок:

```
# equivs-build name
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_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package 'name' in '../name_1.0_all.deb'.

The package has been created.
Attention, the package has been created in the current directory,
```

І встановити отриманий .deb-файл.

Як видно, equivs можна застосовувати по-різному. Наприклад, можна створити пакунок my-favorites, що залежить від програм, які ви зазвичай встановлюєте. Просто звільніть вашу уяву, але будьте обережними.

Важливо відзначити, що в теці /usr/share/doc/equivs/examples знаходяться приклади файлів control. Перевірте їх.

4.2 Видалення зайвих файлів локалей: localepurge

Багато користувачів Debian використовують тільки одну локаль. Наприклад, бразильський користувач зазвичай завжди використовує локаль pt_BR і не цікавиться локаллю es.

localepurge для таких користувачів є вельми корисним інструментом. Ви можете звільнити багато дискового простору, використовуючи лише ті локалі, котрі вам насправді потрібні. Просто виконайте apt-get install localepurge.

Налаштування є дуже простим, питання debconf допоможуть користувачеві здійснити його покроково. Будьте дуже обережними, відповідаючи на перше запитання, бо неправильна відповідь може видалити всі файли локалей, навіть тих, котрі ви використовуєте. Єдиним шляхом до відновлення цих файлів буде перевстановлення всіх пакунків, що їх забезпечують.

4.3 Як дізнатись, які пакунки можуть бути оновлені

Програма apt-show-versions показує, які пакунки в системі можуть бути оновленими та деяку корисну інформацію. Параметр -u виводить список пакунків, котрі можуть бути оновлені:

```
$ 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
```

Розділ 5

Отримання інформації про пакунки

Існують деякі програми-оболонки для системи APT, що значно спрощують отримання списку пакунків, доступних для встановлення або вже встановлених, а також відображення розділу, в якому знаходиться пакунок, його пріоритету, опису, тощо.

Однак... нашою метою зараз є вивчення можливостей власне APT. Отже, яким чином можна знайти назву потрібного нам пакунка.

Маємо кілька варіантів вирішення цієї задачі. Почнемо з apt-cache. Ця програма використовується системою APT для супроводу своєї бази даних. Ми зробимо тільки короткий огляд її найбільш практичних застосувань.

5.1 Пошук назв пакунків

Наприклад, уявимо, що ви хочете згадати про старі добрі дні з Atari 2600. Ви маєте бажання встановити емулятор Atari і потім завантажити деякі ігри. Ви можете зробити:

```
# 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
```

Ми знайшли декілька пакунків, що мають певне відношення до предмету пошуку, і побачили їх короткі описи. Щоб отримати більше інформації про конкретний пакунок я після цього можу використати:

```
# 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: 11b3e86a41a60fa1c4b334dd96c1d4b5
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
```

В цьому виводі ми дізнались багато деталей щодо пакунку, котрий хочемо (або не хочемо) встановити і переглянули повний опис пакунка. Якщо пакунок вже встановлено до вашої системи, але є його новіша версія, ви побачите інформацію про обидві версії. Наприклад:

```
# 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.
```

Зверніть увагу, що першим в списку йде доступний пакунок, а другим — той, що вже встановлений в системі. Щоб отримати більш загальну інформацію про пакунок, можна скористатись:

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
 1.4.5-1(/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_main_binary-i386_Packages)(/var/lib/dpkg/status)

Reverse Depends:
Dependencies:
     1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0) libsdl1.1 (0 (null)) zlib1g (2 1:1.1.3)
Provides:
1.4.5-1 -
Reverse Provides:
```

А щоб просто знайти його залежності:

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

Загалом, ми маємо повний арсенал озброєнь, котрі можна використовувати для пошуку назв потрібних нам пакунків.

5.2 Використання dpkg для пошуку назв пакунків

Одним з методів пошуку назви пакунка — за відомою назвою якогось важливого файлу з нього. Наприклад, щоб знайти пакунок, котрий забезпечує певний необхідний вам для компіляції .h-файл, треба виконати:

```
# 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
```

або:

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

Дізнатись назви пакунків, встановлених в вашій системі (що може бути корисним, наприклад, якщо ви плануєте очистити ваш жорсткий диск), можна так:

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

Недоліком цієї команди є „обрізання“ назв пакунків. В наведеному вище прикладі повна назва пакунку — mozilla-browser. Це можна виправити, перевизначивши змінну середовища COLUMNS, ось так:

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

або вказувати опис пакунка, чи його частину:

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

5.3 Як встановлювати пакунки „при потребі“

Ви компілюєте програму і, раптом, бабах! Помилка — немає потрібного .h-файлу. Програма auto-apt може вберегти вас від такого розвитку подій. Вона запитує вас, чи хочете ви встановити необхідні пакунки, зупиняє відповідний процес та відновлює його після встановлення пакунків.

Все, що потрібно зробити, це запустити:

```
# auto-apt run command
```

Де command — це команда, якій під час виконання можуть знадобитись деякі недоступні файли. Наприклад:

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

Тоді запити щодо встановлення потрібних пакунків та виклики apt-get будуть відбуватись автоматично. Якщо у вас запущені X, графічний інтерфейс замінить звичний текстовий.

З метою більшої ефективності auto-apt повинна використовувати базу даних з найсвіжішої інформацією. Це досягається шляхом виконання команд auto-apt update, auto-apt updatedb та auto-apt update-local.

5.4 Як дізнатись до якого пакунку належить файл

Якщо ви хочете встановити пакунок, котрий не можете знайти за допомогою apt-cache, але знаєте назву самої програми або деякі назви файлів з нього, ви можете скористатись apt-file для пошуку назви пакунка. Це виглядає приблизно так:

```
$ apt-file search назва_пакунка
```

Вона працює подібно до dpkg -S, однак також покаже вам невстановлені пакунки, що містять певний файл. Її також можна використовувати під час компіляції програм для відслідковування пакунків з необхідними файлами заголовків, хоча auto-apt є набагато кращим методом для вирішення цієї проблеми, див. ‘Як встановлювати пакунки „при потребі“’ на стор. 28.

Також можна вивести список файлів в пакунку, виконавши:

```
$ apt-file list назва_пакунка
```

apt-file, як і auto-apt, зберігає всю необхідно інформацію (про те, які файли в якому пакунку знаходяться), в базі даних, котру потрібно оновлювати. Це робиться так:

```
# apt-file update
```

Зазвичай, apt-file використовує ту ж саму базу даних, що й auto-apt (див. ‘Як встановлювати пакунки „при потребі“’ на стор. 28).

5.5 Як залишатись проінформованим про зміни в пакунках

Кожен пакунок встановлює в свою теку для документації (/usr/share/doc/packagename) файл під назвою changelog.Debian.gz, що містить список змін, зроблених в останній версії. Ви можете прочитати ці файли за допомогою, наприклад, zless, але це трохи нецікаво після завершення оновлення системи займатись пошуками журналів змін для кожного оновленого пакунка.

За допомогою інструменту apt-listchanges цю задачу можна автоматизувати. Для початку потрібно встановити пакунок apt-listchanges. Під час встановлення Debconf його налаштує. Деякі питання, можливо, не будуть відображатись, в залежності від того, який пріоритет ви вказали при налаштуванні самого Debconf. Давайте такі відповіді на питання, які вам більше довподоби.

Спочатку вас запитають, яким чином apt-listchanges інформуватиме вас про зміни. Їх можуть відсилати вам електронною поштою, що, вочевидь, виглядає доцільним при автоматичних оновленнях, або ви можете переглядати ці зміни за допомогою чогось схожого на less перед тим як почнеться власне процес оновлення пакунків. Якщо ви не хочете, щоб apt-listchanges запускалась автоматично під час оновлень системи, можете вибрати варіант відповіді none.

Після встановлення apt-listchanges, як тільки apt завантажить пакунки (або візьме їх з компакт-диску, або з підмонтованому диску), apt-listchanges виведе список змін, внесених до пакунків, перед їх встановленням.

Розділ 6

Робота з джерельними пакунками

6.1 Завантаження джерельних пакунків

Звичайною річчю в світі вільного програмного забезпечення є вивчення джерельного коду чи, навіть, внесення виправлень в помилковий код. Щоб здійснити це, вам потрібно завантажити джерельні коди програми. Система APT пропонує простий метод отримання джерельного коду багатьох програм, що містяться в дистрибутиві, включаючи всі файли, потрібні для створення .deb-пакунку програми.

Іншим загальноприйнятим використанням джерельних пакунків Debian є адаптація більш свіжих версій програм, наприклад, зі збірки unstable, для використання в stable. Компіляція пакунка в середовищі stable згенерує .deb-файл з залежностями, пристосованими до доступних в цій збірці пакунків.

Для досягнення цього, запис deb-src в вашому /etc/apt/sources.list повинен вказувати на unstable і бути дозволеним (розкоментованим). Див. ‘Файл /etc/apt/sources.list’ на стор. 3.

Для завантаження джерельного пакунка скористайтесь такою командою:

```
$ apt-get source packagename
```

При цьому завантажується три файла: .orig.tar.gz, .dsc та .diff.gz. Для пакунків, створених спеціально для Debian, останнього файлу немає, а в назві першого зазвичай відсутня вставка orig.

Файл .dsc використовується програмою dpkg-source для розпаковування джерельного пакунка в теку назва_пакунка-версія. Кожен завантажений джерельний пакунок має теку debian/, в якій знаходяться необхідні для створення .deb-пакунка файли.

Щоб після завантаження пакунок автоматично збирався, просто додайте -b до командного рядка, ось так:

```
$ apt-get -b source packagename
```

Якщо ви вирішили не створювати .deb-файл одразу після завантаження, ви можете зробити це пізніше, виконавши команду

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

в теці, створеній для пакунка після його завантаження. Щоб встановити пакунок, збудований за допомогою вказаної вище команди, потрібно скористатись менеджером пакунків напряму, наприклад:

```
# dpkg -i файл.deb
```

Існує відмінність між методом source та іншими методами apt-get. Метод source можуть використовувати звичайні користувачі, він не потребує спеціальних прав root. Файли завантажуються до теки, з якої викликається команда apt-get source package.

6.2 Пакунки, потрібні для компіляції джерельних пакунків

Зазвичай, для компіляції джерельного пакунка потрібні деякі заголовки та бібліотеки спільного користування. В усіх джерельних пакунках в файлі control є поле під назвою Build-Depends:, котре показує, які додаткові пакунки потрібні для збірки джерельного.

З APT ці пакунки можна завантажувати дуже просто. Просто запустіть apt-get build-dep package, де package — це назва пакунка, який ви хочете зібрати. Наприклад:

```
# 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-imlib-dev imlib-progs libgnome-dev libgnorba-dev
  libgpmg1-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]
```

Пакунки, що будуть встановлені, потрібні для коректної збірки gmc. Важливо відмітити, що ця команда не завантажує джерельний пакунок програми, котру ви маєте компілювати. Для цього потрібно окремо запустити apt-get source.

Якщо вашою метою є лише перевірка того, які пакунки потрібні для збірки даного пакунка, ви можете, як варіант, використати команду apt-cache show (див. ‘Отримання інформації про пакунки’ на стор. 25), яка, серед іншої інформації, показує також список таких пакунків в рядку Build-Depends.

```
# apt-cache showsrc пакунок
```

Розділ 7

Як виправляти помилки

7.1 Загальні помилки

Помилки завжди будуть траплятись, більшість з них пов'язані з неуважністю користувачів. Далі наводиться список деяких помилок, про які найчастіше повідомляють, і методи їх вирішення.

Якщо під час виконання apt-get install package ви отримуєте подібне повідомлення...

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/ Packages' (/var/state/apt/lists/peopl
stat (2 No such file or directory)
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

ви забули запустити apt-get update після внесення останньої зміни до файлу /etc/apt/sources.list.

Якщо помилка виглядає так:

```
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?
```

коли ви намагаєтесь запустити apt-get з будь-яким методом, окрім source, значить ви не маєте прав root, тобто запускаєте програму як звичайний користувач.

Помилка, подібна до наведеної вище трапляється ще й тоді, коли ви намагаєтесь запустити дві копії apt-get одночасно або при спробі виконати apt-get в той час, коли dpkg є активним. Єдиний метод, який можна застосовувати одночасно з іншими — це метод source.

Якщо процес встановлення переривається посередині і виявляється, що потім встановлювати або видаляти пакунки неможливо, то виконайте такі команди:

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

І спробувати знову. Запуск останніх двох команд може знадобитись більше одного разу. Це важливий урок для тих авантюристів, котрі використовують unstable.

Якщо ви отримуєте помилку „E: Dynamic MMap ran out of room“ при запуску apt-get update, додайте такий рядок до /etc/apt/apt.conf:

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

7.2 Де мені шукати підтримки?

Якщо ви у безвихідному становищі, проконсультуйтесь з обширною документацією системи управління пакунків Debian. Параметри —help і сторінки довідки можуть дуже вам допомогти, як і документація, що знаходиться в підтеках /usr/share/doc, наприклад /usr/share/doc/apt.

Якщо ця документація не в змозі розвіяти ваші сумніви, спробуйте знайти відповідь в списках розсилки Debian. Ви можете знайти більше інформації про спеціальні списки для користувачів на веб-сайті Debian: http://www.debian.org.

Пам’ятайте, що ці списки і ресурси повинні використовувати тільки користувачі Debian; користувачі інших систем знайдуть кращу підтримку на ресурсах спільнот їх власних збірок.

Розділ 8

Які дистрибутиви підтримують APT?

Ось назви деяких дистрибутивів, котрі використовують APT:

Debian GNU/Linux (http://www.debian.org) - саме для цього дистрибутиву APT і був розроблений

Conectiva (http://www.conectiva.com.br) - це перший дистрибутив, який портував APT для використання з 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/)

Розділ 9

Подяки

Велика подяка моїм чудовим друзям з проекту Debian-BR і самого Debian, які постійно допомагають мені і завжди додають сил для праці на благо людства, а також підтримують мене в моїх потугах зберегти світ. :)

Також я хочу подякувати CIPSGA за посильну допомогу нашому проекту та всім вільним проектам — носіям величних ідей.

І особливі подяки:

Yooseong Yang <yooseong@debian.org>

Michael Bramer <grisu@debian.org>

Bryan Stillwell <bryan@bokeoa.com>

Pawel Tecza <pawel.tecza@poczta.fm>

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

Luca Monducci <luca.mo@tiscali.it>

Tomohiro KUBOTA <kubota@debian.org>

Pablo Lorenzzoni <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 Åslund <d98-jas@nada.kth.se>

Розділ 10

Нові версії цього підручника

Цей підручник створено в рамках проекту Debian-BR (http://www.debian-br.org) з метою допомоги у щоденному використанні Debian.

Нові версії цього документу будуть доступними на сторінці проекту документації Debian http://www.debian.org/doc/ddp.

Коментарі та зауваження можна відсилати мені напряму за адресою <kov@debian.org>.