



Факультет биоинженерии и биоинформатики

Московский государственный университет имени М.В.Ломоносова



Установка пакетов и программ



Системные менеджеры пакетов

dpkg

Установка, обновление и удаление пакетов

Удалять и обновлять пакеты нужно с помощью apt.

В случае пакета, отсутствующего в репозиториях:

получение списка файлов, которые будут установлены пакетом

```
$ dpkg-deb -c path-to-package.deb | less
```

просмотр информации о пакете

```
$ dpkg-deb -I path-to-package.deb
```

просмотр контрольного файла

```
$ dpkg-deb -I path-to-package.deb postinst | less
```

установка (или обновление) пакета

```
# dpkg -i path-to-package.deb
```



dpkg

Исследование установленных пакетов

поиск пакета-источника файла

```
$ dpkg -S /usr/bin/env
```

```
coreutils: /usr/bin/env
```



просмотр списка файлов пакета

```
$ dpkg -L coreutils | less
```

проверка целостности файлов, установленных пакетом

```
$ dpkg -V coreutils
```

поиск всех изменившихся установленных пакетами файлов

```
# dpkg -V # 'c' означает, что это файл конфигурации
```

```
??5?????? c /etc/ImageMagick-6/policy.xml
```

```
??5?????? c /etc/fuse.conf
```

```
??5?????? c /etc/grub.d/05_debian_theme
```

-- далее можно определить пакет и его версию, найти его в репозитории,

-- извлечь интересующий файл и сравнить с помощью diff

apt

Установка, обновление и удаление пакетов

обновить кэш пакетов

```
# apt update
```

...

8 packages can be upgraded. Run 'apt list --upgradable' to see them.

установить все обновления

```
# apt upgrade
```

удалить все автоматически установленные пакеты

которые больше не нужны

```
# apt autoremove
```

поиск по названию и описанию

```
$ apt search image
```

получение информации о пакете

```
$ apt show imagemagick
```

установить/удалить пакет

```
# apt install/remove imagemagick
```

удалить пакет полностью (вместе с конфигурацией)

```
# apt purge imagemagick
```





Альтернатива apt с дополнительными возможностями:

- ▶ вывод списка всех доступных версий пакета с указанием репозитория (`aptitude versions`);
- ▶ определение зависимых/конфликтующих пакетов (`aptitude why/why-not`);
- ▶ более гибкий поиск, можно искать почти по всем полям метаданных.

apt-file



Программа для поиска файлов в пакетах в репозитории (без предварительного скачивания и установки).

Позволяет:

- ▶ искать пакеты, содержащие файл с заданным именем;
- ▶ просматривать список файлов пакета в репозитории.

Не входит в состав apt, устанавливается из отдельного пакета apt-file.



Внесистемные менеджеры пакетов

Какие бывают



- ▶ Менеджеры пакетов из других дистрибутивов (`rpm` , `dnf`).
- ▶ Узкоспециализированные менеджеры пакетов (`pip` , `npm`).
- ▶ Менеджеры для (частично) изолированных окружений (`conda` , `flatpak`).

conda

Принципы функционирования



- ▶ Пакеты conda расчитаны на установку файлов в окружение (conda environment).
- ▶ Пакеты могут содержать бинарные файлы и динамические библиотеки, расчитанные на запуск в активированном окружении conda, т.е. с минимальной зависимостью от системных средств.
- ▶ Пакеты связаны системой зависимостей и хранятся в репозиториях, которые называются каналами (channel).
- ▶ Окружение – просто папка в файловой системе.
- ▶ Активация окружения – установка значений переменных окружения командной оболочки, в частности, PATH , CONDA_PREFIX , GSETTINGS_SCHEMA_DIR .
- ▶ Создание и активация окружений, установка и удаление пакетов осуществляется программой conda (сценарий python).

conda

Использование



создание окружения по указанному пути

\$ conda create -р /пу/ть [пакет] ...

или по имени (расположение определяется конфигурацией conda)

\$ conda create -n имя [пакет] ...

[де]активировать окружение (по умолчанию base)

\$ conda [de]activate [имя_или_путь]

установить, обновить или удалить пакет[ы] в окружение (по умолчанию в активное)

\$ conda install/update/remove [-n имя | -р /пу/ть] пакет ...

поиск пакетов

\$ conda search запрос

используется специальный язык запросов, примеры

\$ conda search numpy

\$ conda search 'num*'

\$ conda search conda-forge::numpy

\$ conda search 'numpy>=1.8,<2|3.1*'



Установка сторонних программ

Компиляция из исходных кодов



Типичные этапы:

- ▶ подготовка – загрузка архива, распаковка, установка зависимостей;
- ▶ конфигурация – генерация Makefile на основе заданных опций и обнаруженных зависимостей;
- ▶ компиляция – получение бинарных файлов из исходных кодов;
- ▶ установка – копирование файлов в требуемые папки, корректировка метаданных;
- ▶ тестирование – проверка программы на тестовых данных.

```
$ tar -xf program.tar; cd program
$ ./configure
$ make
$ make install # может потребовать права root
$ make test
```

make



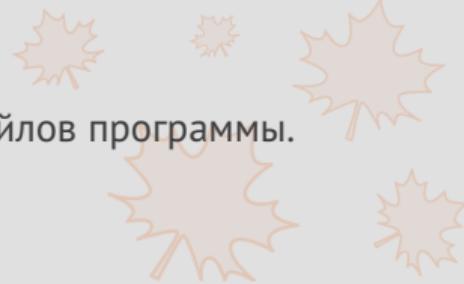
Система сборки программ на основе целей, зависимостей и команд.

- ▶ Текстовые файлы Makefile содержат правила.
- ▶ Правило определяет цель, реквизиты цели и команды для получения цели из реквизитов.
- ▶ Для получения цели (например, исполняемого файла программы) надо сначала получить реквизиты, для этого – реквизиты реквизитов, и т.д.
- ▶ Команды для получения существующей цели не выполняются, если реквизиты не изменились.

`CC = gcc`

```
program: main.o lib.o
        $(CC) -o program main.o lib.o
%.o: %.c
        $(CC) -o $@ -c $^
```

Установочные пути



Обычно на этапе конфигурации можно задать установочные пути для файлов программы.

Разные типы файлов обычно располагаются в разных папках:

`bin` – исполняемые файлы;

`lib` – библиотеки;

`etc` – файлы конфигурации;

`share` – документация и статические файлы;

`var` – файлы, которые могут изменяться.

Есть несколько стандартных установочных путей:

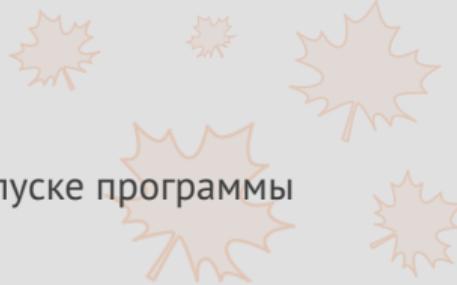
`/` – системные программы;

`/usr/` – пользовательские программы (из стандартных пакетов);

`/usr/local` – сторонние программы (из пакетов?);

`/opt` – сторонние программы.

Разделяемые библиотеки



Если при компиляции использовалась динамическая линковка, то при запуске программы осуществляется поиск и загрузка разделяемых библиотек.

С помощью динамического загрузчика `ld.so` / `ld-linux.so` в различных папках:

`DT_RPATH` – атрибут из бинарного файла;

`LD_LIBRARY_PATH` – переменная окружения;

`DT_RUNPATH` – атрибут из бинарного файла;

кэш `ld.so` – файл `/etc/ld.so.cache`;

`/lib` или `/lib64`;

`/usr/lib` или `/usr/lib64`.

Узнать требуемые библиотеки можно с помощью `ldd`.