**Работа в графическом и консольном режиме**

**Встроенная документация. Команды man, apropos**

На прошлом занятии мы уже касались поиска информации о нужной программе. Разберем более подробно возможности работы с документацией.

В Unix существует встроенная система документации, обратившись к которой, всегда можно найти название нужной утилиты, описание  параметра команды или пример ее использования. Например, если требуется  подробное описание команды cp, используем команду man (от MANual).

|  |
| --- |
| user@user-virtual-machine:~$ man cp |

Справка по командам обычно занимает несколько страниц, поэтому man использует для просмотра команду less. Для перехода к следующей странице используйте клавишу <PgDn>, обратно — <PgUp>. Если эти клавиши не срабатывают, используйте для перехода пробел и клавишу u соответственно. Для выхода из  man служит клавиша q. Есть еще пара полезных клавиш: g — переход в начало, G — в конец документа. Для поиска вперед используйте **/(пишем чтоискать)**, для обратного поиска — **?(пишем что искать)**, повторить поиск в том же направлении — n.

Теперь пара слов о стандартных разделах справочной страницы.

NAME — краткое описание команды.

Раздел SYNOPSYS описывает различные варианты синтаксиса. В этом разделе могут использоваться специальные обозначения. Например, если опции или параметры заключены в квадратные скобки, это означает, что они не обязательны для использования. Троеточие после параметра говорит, что он может повторяться многократно, как в случае, когда несколько файлов копируются в один каталог назначения.

В разделе DESCRIPTION описываются опции (иногда их называют ключами) программы. Бывает, что в описании через запятую перечислены 2 опции, например -v, --verbose. В таких случаях мы имеем дело с разными стилями одной опции. -v — классический вариант,  описанный в стандарте POSIX. --verbose — опция в стиле GNU, новомодный вариант. Можно использовать то, что больше нравится.

Еще один полезный раздел  обычно находится в конце справочной странице, он называется SEE ALSO.  Как правило, здесь приводят ссылки на другие утилиты с похожим функционалом.

Кроме man есть еще несколько полезных справочных утилит. Если мы не знаем точного имени утилиты, можно попробовать найти ее с помощью apropos, которая выполнит поиск по ключевым словам в описании утилит. Например найдем как называется утилита, которая меняет пароль пользователя:

|  |
| --- |
| user@user-virtual-machine:~$ apropos password  apg (1)          - generates several random passwords  chage (1)        - change user password expiry information  chgpasswd (8)    - update group passwords in batch mode  chpasswd (8)     - update passwords in batch mode  cpgr (8)         - copy with locking the given file to the password or g.. cppw (8)         - copy with locking the given file to the password or g.. cracklib-check (8)   - Check passwords using libcrack2  create-cracklib-dict (8) - Check passwords using libcrack2  crypt (3)        - password and data encryption [...] |

Мы получили нужную информацию, но слишком много вариантов — на 3 страницы текста.

Попробуем уточнить поиск, выбрав все строчки, где встречается слово change. Используем утилиту grep, которая ищет по заданной строке в выводе от предыдущей программы. Делается это так:

|  |
| --- |
| user@user-virtual-machine:~$ apropos password | grep change chage (1)        - change user password expiry information lppasswd (1)     - add, change, or delete digest passwords.  passwd (1)       - change user password smbpasswd (8)    - change a user's SMB password |

Вот она, passwd  — то, что нужно.  Обратите внимание на число в скобках после имени команды. Это глава или раздел справочного руководства man, в которых может встречаться описание утилиты. Разделов много, приведем самые полезные:

* 1 — команды пользователя,
* 2 — системные вызовы ядра (используется программистами),
* 5 — форматы файлов,
* 8 — команды администрирования.

Эти номера можно использовать в командах man и apropos. Например, passwd — не только команда, но и имя системного файла. Если нам интересен формат этого файла, надо набрать:

|  |
| --- |
| man 5 passwd |

Для apropos полезна опция -s. Например, если мы ищем программы для работы с паролями, набираем:

|  |
| --- |
| user@user-virtual-machine:~$ apropos -s 1 passwd  gpasswd (1)      - управление /etc/group и /etc/gshadow  passwd (1)       - изменяет пароль пользователя  grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB  lppasswd (1)     - add, change, or delete digest passwords. passwd (1ssl)    - compute password hashes  user@user-virtual-machine:~$ |

Описание passwd — по-русски. Иногда это может создать проблемы для поиска, поэтому есть еще полезная опция, -L, она задает язык, на котором будет справка. Чтобы все искать и читать по-английски, задаем  apropos -s 1 -L C passwd.

|  |
| --- |
| user@user-virtual-machine:~$ apropos -s 1 -L C passwd  gpasswd (1)          - administer /etc/group and /etc/gshadow  grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB  lppasswd (1)     - add, change, or delete digest passwords. passwd (1)       - change user password  passwd (1ssl)    - compute password hashes  user@user-virtual-machine:~$ |

Еще одна полезная справочная программа — whatis. От apropos отличается тем, что ищет по именам команд:

|  |
| --- |
| user@user-virtual-machine:~$ whatis passwd  passwd (5)       - файл паролей  passwd (1)       - изменяет пароль пользователя  passwd (1ssl)    - compute password hashes |

Иногда в разделе справочной страницы SEE ALSO встречается ссылка на более полное описание утилиты в системе info и проводится команда для получения полной справки. Пример для cp:

|  |
| --- |
| info coreutils 'cp invocation' |

Info — гипертекстовая система документации вроде текстового браузера, а команда в man содержит полную ссылку на информацию о вызове команды в info. Если запустить info без параметров, мы выйдем на главную страницу документации. Перемещение по страницам и поиск  как в man. Строчки, которые начинаются со звездочек, — гиперссылки, по которым можно пойти дальше. Чтобы вернуться из ссылки на уровень выше, используйте клавишу u как Back в браузере. Выход — по клавише q.

**Процессы, потоки, потоки ядра**

Современные операционные системы состоят из как минимум двух сильно отличающихся по своим возможностям частей — ядро, работающее в пространстве ядра, и на уровне режима работы процессора взаимодействующее с оборудованием и ресурсами, и пользовательское пространство, работающее в уровне ядра, и на уровне режима работы процессора  ограниченное в доступе к ресурсам. Попытки обратиться к не разрешенным ресурсам приводят к вызову процессором исключения и передачи управления ядру операционной системы.

Так как Linux — система многозадачная, существует три возможности реализации многозадачности, одна — для ядра, и два — для пользовательского пространства.

Для ядра это потоки (иногда говорят процессы) ядра.

Для пользовательского пространства это процесс и потоки. В Linux потоки также являются особого рода процессами (так они тоже имеют собственные PID), но при этом они делят одну и ту же память процесса, которому они принадлежат (первый созданный процесс, фактически первый поток процесса). При создании копии  процесса (fork) происходит копирование его ресурсов, так что это более безопасно, но менее эффективно.

Каждый процесс, кроме упомянутых UID и GID, также имеет идентификаторы PID (process ID) и PPID (parent process ID).

Перечень процессов можно посмотреть с помощью команды:

|  |
| --- |
| $ ps aux |

ps отображает процессы пользовательского пространства и потоки ядра (отображаются в квадратных скобках). Потоки не отображаются Чтобы посмотреть потоки, используйте ключ -T, если хотите посмотреть потоки конкретного процесса, используйте ключ -p

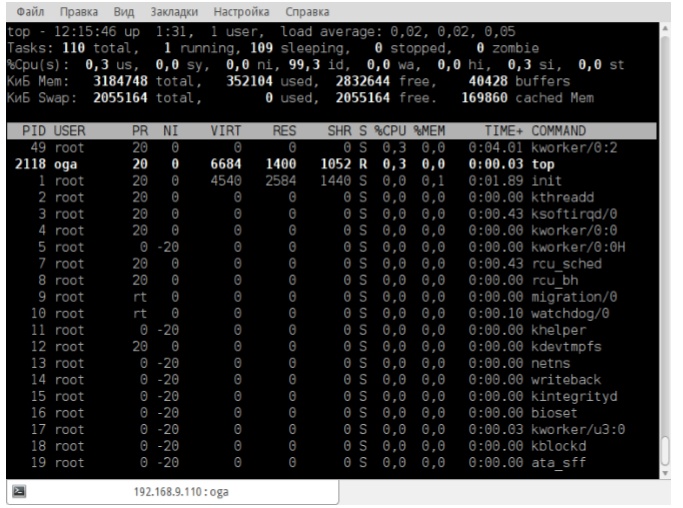
|  |
| --- |
| $ ps -T -p 4242 |

Чтобы посмотреть историю запущенных процессов, можно воспользоваться командой:

|  |
| --- |
| $ pstree |

Существуют и другие способы мониторинга процессов: top  и htop — обязательно ознакомьтесь с ними!

Консольная утилита top служит для динамического просмотра списка процессов. Утилита каждые несколько секунд выдает топ-20 процессов, по умолчанию отсортированный по потреблению процессорного времени. Можно сортировать список процессов по объёму занимаемой памяти, утилизации процессора, пользователям или номерам процессов.



top имеет горячие клавиши:

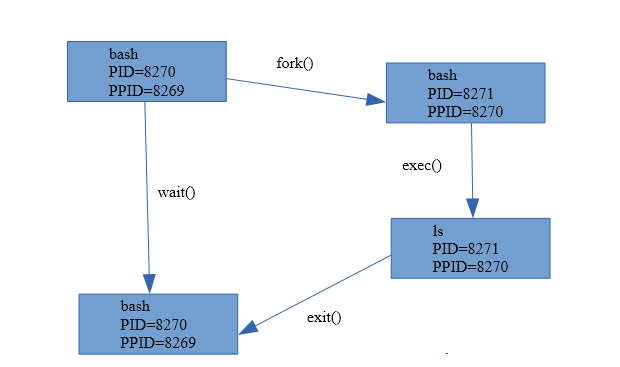
* h — вызов справки.  
  k — ввести PID для команды kill.
* r — указать PID для изменения параметра nice, влияющий на приоритет процесса ( -20 макс. приоритет, 19  мин. приоритет, 0 — по умолчанию). Обычный пользователь может только уменьшать приоритет своих процессов, назначая им nice от 0 до 19.
* q — завершить top.

Существует более продвинутая консольная утилита для мониторинга и управления процессами — htop. Она не установлена по умолчанию, но доступна из репозитария Ubuntu. Перечислим её возможности в сравнении с top:

* позволяет выполнять горизонтальный и вертикальный скроллинг, что позволяет видеть все процессы и параметры командной строки;
* htop запускается быстрее top, который предварительно собирает данные перед тем как показать список процессов;
* в htop не надо вводить PID для kill как в top, все через скроллинг и функциональные клавиши;
* также нет необходимости вводить PID и значение nice;
* в htop поддерживается мышь;
* top идет в системе по умолчанию и более распространен в мире UNIX.

## **Порядок запуска процесса**

Рассмотрим порядок запуска процесса. Например, работая в оболочке, мы запустили команду ls.



На первом этапе оболочка выполняет системный вызов fork(), в результате которого происходит ее клонирование: создается полная копия процесса оболочки, включая копию адресного пространства и контекста. Между клонами есть отличие: в процессе-родителе fork() возвращает PID потомка, а в потомке код возврата — 0. Каждый из клонов начинает выполнение с момента вызова fork, но уже самостоятельно и с учетом вышесказанного. Итак, потомок, выяснив, что он потомок (так как код возврата 0) с помощью системного вызова exec загружает в свое адресное пространство исполняемый код запускаемой программы, в нашем случае ls, и далее ее выполняет.

В это же время родительский процесс ожидает завершение дочернего процесса с помощью системного вызова wait(). Если мы запустили команду с & в конце, ожидания не будет.

Дочерний процесс завершается функцией exit(). В случае успешного завершения, exit возвращает 0, если произошла ошибка — код ошибки, отличный от нуля. После этого ядро освобождает ресурсы, которые занимал потомок, и передает код его завершения родителю в качестве кода возврата для wait().

Иногда бывает так, что процесс-родитель прекращает существование, не успев дождаться завершения wait(), то есть раньше завершения потомка. Процесс, у которого процесс-родитель завершился раньше) называют процессом-сиротой. Процессы-сироты усыновляет процесс init, который теперь считается для них родителем.

## **Сигналы процессам**

Допустим, программа зависла. Нажимаем Alt-F2 (или Ctrl-Alt-F2, если работали в X-сервер), смотрим id процесса.

|  |
| --- |
| $ ps ax | grep hello |

Процессу можно направить сигнал.

Например, 9 сигнал – принудительно снять.

|  |
| --- |
| $ kill -9 4242 |

В Linux существуют следующие сигналы:

* 1 (SIGHUP) — информирует программу о потери связи с терминалом. Ситуация часто возникала в прошлом, в режиме терминального доступа. В настоящее время может применяться для двух целей:
  + Информирование дочернего процесса о завершении родительского. При завершении родительского процесса ядро направит сигнал 1 дочерним процессам.
    - Если приложение запущено в фоновом режиме, с помощью указания & в конце команды или fg, при завершении консоли получит сигнал SIGHUP и программа, исполняющаяся в фоновом режиме, если была запущена таким образом. Это напоминает исходное значение SIGHUP.

|  |
| --- |
| $ someprogram& |

* + - либо

|  |
| --- |
| $ someprogram  ^Z  $ fg |

* + - Этому можно воспрепятствовать с указанием nohup.

|  |
| --- |
| $ nohup someprogram& |

* + Очевидно, что по умолчанию сигнал приводит к завершению программы. Но демоны перехватывают данный сигнал и используют для перечитывания файлов конфигурации.
* 2 (SIGINT) формируется при нажатии Ctrl-C. Завершает программу, но может перехватываться или блокироваться (например, в программах, в которых Ctrl-C используются для операции «копирование» или в принципе нет необходимости такого варианта остановки).
* 8 (SIGFPE) — ошибка операции с целочисленной арифметикой (переполнение, деление на ноль). Сохраняется дамп памяти.
* 9 (SIGKILL) — безусловное завершение программы. Сигнал не может быть перехвачен программой, потому позволяет её остановить в любом случае (но не позволит снять процесс-зомби).
* 11 (SIGSEGV) — формируется при попытке программы обратиться к не принадлежащей ей области памяти. Обычно выводится сообщение «Segmentation fault» и сохраняется дамп памяти. Как правило, такое случается в результате ошибок программиста при работе с указателями.
* 15 (SIGTERM) — вежливая просьба программе завершить работу. Программа может сохранить данные и т.д.
* Есть и другие варианты.

Для отправки сигнала из операционной системы можно использовать команду kill.

|  |
| --- |
| $ kill -15 4242 |

Перенаправление потоков ввода-вывода

Далее мы познакомимся с подходом UNIX-Way, одним из принципов которого является то, что каждая программа хорошо отлажена и решает только одну задачу, но сами программы-утилиты можно комбинировать, что очень часто применяется. Сейчас мы разберем, каким образом это работает.

При запуске каждая программа автоматически открывает 3 специальных файла, которые ещё называют потоками ввода-вывода. Их различают по номерам файловых дескрипторов (указателей на файл):

0 — стандартный ввод (stdin). По умолчанию принимается с клавиатуры терминала.

1 — стандартный вывод (stdout). По умолчанию — на экран терминала.

2 — стандартная ошибка (stderr). По умолчанию — на экран терминала.

Поток стандартного вывода можно перенаправить в файл (в том числе специальный, такой как принтер или файл «в никуда» /dev/null). Стандартный ввод программе можно назначить не с клавиатуры, а из файла. Опишем все варианты перенаправления для каждого потока:

Для stdout от программы cmd можно использовать символы перенаправления:

cmd > file перенаправить в файл с перезаписью (при каждой записи предыдущее содержимое файла будет утрачено),

cmd >>file в файл с добавлением (новые записи будут добавлены в конец файла),

cmd1 | cmd2 — конвейер. Запускаются cmd1 и cmd2, при этом stdout от cmd1 перенаправляется на stdin программы cmd2,

cmd | tee file — перенаправить копию потока в файл. После tee c потоком stdout можно выполнять дополнительные операции,

cmd | sudo tee file — вариант, если в file имеет возможность записать только root.

Для stdin программы cmd можно использовать символы перенаправления:

cmd < file перенаправить ввод из файла программе.

Для stderr от программы cmd можно использовать символы перенаправления:

cmd 2> file перенаправить в файл с перезаписью;

cmd 2>>file в файл с добавлением;

cmd1 2>&1 — присоединить к stdout;

После операций с потоком stderr можно выполнять дальнейшее перенаправление для stdout.

Пример:

cmd 2> /dev/null | tee f1 f2 f3

Перенаправить поток ошибок в /dev/null (спрятать поток), поток стандартного вывода на вход программе tee, которая дублирует его в файлы f1, f2, f3.

cmd < f1 | cmd2 2>&1 > f2

Направить программе cmd ввод из файла f1. затем стандартный вывод cmd направить на вход cmd2. Объединить поток ошибок от cmd2 со стандартным выводом от cmd2 и все перенаправить в f2.

Создать файл.

>file1 — самый простой способ создать пустой файл;

cat >file1 ввести файл с консоли (завершение Ctrl-Z).

**Архиваторы и компрессоры в Linux**

Наверняка вы использовали в Windows программы-архиваторы (rar и zip), которые позволяют упаковать иерархию каталогов c файлами в один сжатый архивный файл. В Linux тоже есть версии этих архиваторов, которыми можно распаковать архивы, созданные пользователями Windows. Также есть графические программы-оболочки, облегчающие пользователям работу с архивами. Но сейчас мы поговорим о других программах из мира UNIX, которые появились задолго до своих Windows-собратьев. Чем же они уникальны?

Во-первых, есть различие в терминах. Архиваторами в UNIX и Windows называют программы, немного отличающиеся по назначению. В UNIX архиваторы выполняют следующие задачи:

* создавать архивы как объединения заданных файлов в виде одного обычного или последовательности данных на внешнем носителе (диск, лента и т. д.),
* добавлять в архив или извлекать из него отдельные файлы или группы файлов.

В  Windows архиваторы делают то же что в UNIX, плюс дополнительно занимаются сжатием архивов. В UNIX сжатием занимаются отдельные утилиты, называемые компрессорами. Такое разделение функций является характерным для философии unixway: каждой задаче — свой продвинутый инструмент, плюс комбинация разных инструментов для решения новых задач. Подход оказался оправданным, он позволяет использовать новые, более мощные компрессоры в сочетании со старыми архиваторами. Кроме того, подход позволяет с помощью конвейеров добавлять в рабочую цепочку архиватор-компрессор другие программы, например, для шифрования данных, что очень актуально для хранения архивов на популярных облачных сервисах. Далее рассмотрим конкретные примеры программ для архивации и сжатия файлов.

В  UNIX с давних времен существуют две стандартные программы-архиватора: tar и cpio. В Linux также есть GNU-версии этих утилит. Они имеют свои уникальные опции с длинными именами, но для совместимости поддерживают стандартный набор ключей. При рассмотрении мы будем стараться придерживаться стандарта, чтобы вы могли использовать утилиты в любой UNIX-системе.

## **Утилита tar (Tape ARchiver)**

Судя по названию, предназначалась для создания архивах на магнитных лентах, но также прекрасно работает с архивами в виде файлов. Имеет три основных варианта использования:

### Создание архивов с tar

tar опции arc\_file file\_or\_dirname ...

где опции имеют следующее значение:

* с — создать архив.
* z — сжимать файл архива с помощью gzip (см. ниже). Эта опция есть только в GNU-версии.
* j — сжимать файл архива с помощью bzip2 (см. ниже). Эта опция есть только в GNU-версии.
* v — (verbose) выдает в терминал имя добавляемого файла. Когда файлов много, обычно эту опцию не указывают и tar работает молча.
* f — после этой опции указывают имя архивного файла. Часто используют специальное имя «-», которое означает что поток данных архива надо выдать в стандартный выходной поток. В этом режиме обычно используют конвейерную обработку, когда данные архиватора принимаются компрессором для последующего сжатия. Если забыть указать файл после -f, tar будет пытаться писать на ленточное устройство по умолчанию.
* file\_or\_dirname — список архивируемых файлов или каталогов. Каталоги архивируются рекурсивно вместе со всем содержимым.

Пример:

|  |
| --- |
| tar cf foo.tar ~/foo |

### Просмотр оглавления архива

|  |
| --- |
| tar tvf arc\_file |

t — прочитать оглавление архива. Остальные опции как в случае создания архива. Пример:

|  |
| --- |
| tar tvf foo.tar |

### Извлечь файлы из архива

|  |
| --- |
| tar xvf arc\_file [filename] |

x — извлечь файл(ы). Если последний параметр не указывать, будут извлечены все файлы из архива в текущий каталог. Имена извлекаемых файлов надо указывать, так как они показаны в листинге оглавления архива (с тем же относительным путём).

## **Компрессоры**

В UNIX и Linux семейство компрессоров представлено сразу несколькими программами: compress, gzip, bzip2. Они используют разные алгоритмы сжатия и в разной степени эффективны. Каждый компрессор включает в себя три утилиты: утилита сжатия (compress, gzip, bzip2), утилита декомпрессии (uncompress, gunzip, bunzip2, ) и ещё одна утилита декомпрессии, которая по умолчанию выдаёт результат в стандартный вывод. Утилиты разных компрессоров имеют унифицированный интерфейс, так что нет необходимости запоминать ключи для каждого компрессора отдельно.

### gzip

gzip — это GNU-версия программы compress, созданная, чтобы избежать патентных ограничений. Ключи у gzip такие же, как у compress. Дополнительно есть полезная опция, задающая с помощью числового значения от 1 до 9 степень сжатия. Это позволяет управлять соотношением сжатие/время в зависимости от потребностей, например можно задать -5 для больших файлов и выиграть на времени исполнения при достаточной степени сжатия. Суффиксы сжатых файлов, которые добавляет gzip — .gz. Для декомпрессии есть утилиты gunzip и gzcat, аналогичные uncompress и zcat. gunzip ищет файлы с суффиксами .gz и .Z, .z, tgz, taz.

|  |
| --- |
| gzip [ключи] [файл]... |

-с — направляет результат в стандартный вывод, исходные файлы не меняются. Обычно для использования совместно с архиватором.

-d — выполнить декомпрессию указанных файлов.

Без ключей gzip выполняет сжатие каждого файла из заданного списка и записывает результат в новый файл, добавляя к имени суффикс .gz. Исходные несжатые файлы удаляются.

Пример использования совместно с архиватором:

|  |
| --- |
| tar cf - ./data | gzip -c > data.tar.gz |

### gunzip

Декомпрессия с gunzip:

|  |
| --- |
| gunzip [файл]... |

Выполняет декомпрессию файлов из списка, если надо, добавляя к именам суффикс gz. Разархивированный файл записывается с именем без .gz. Исходный сжатый файл удаляется.

### zcat

Декомпрессия с выдачей в стандартный вывод zcat:

|  |
| --- |
| zcat [файл]... |

Этот вариант обычно используют совместно с архиватором для декомпрессии сжатых архивов:

|  |
| --- |
| zcat data.tar.gz | tar xf - |

Ещё один популярный компрессор — bzip2. Превосходит на большинстве файлов gzip по эффективности сжатия, но проигрывает в скорости. Опции такие же, как у gzip. Расширения архивов — .bz2.

**Установка программ в Ubuntu**

Самые базовые возможности установки мы разобрали на прошлом уроке. Разберем подробнее.

Большинство пользователей знает, как устанавливаются программы в Windows. Многие новички в Linux страдают от того, что опираются на эти навыки. По большей части, в Linux все делается по-другому.  Первое, что мы должны осознать — в Linux всегда существует несколько способов решить одну и ту же проблему. Но это не значит, что какой­-то из этих способов не будет лучше других. Поэтому рассмотрим разные способы установки программ в порядке снижения желательности их использования.

## **Ключевые понятия**

Репозиторий — хранилище пакетов. Обычно он располагается на сервере онлайн, хотя ничто не мешает создать локальный репозиторий или репозиторий на DVD. Под пакетом понимают файл .deb (в Ubuntu), который содержит в запакованном виде необходимые программные файлы, инсталляционные скрипты и разного рода дополнительную информацию. Его можно сравнить с файлами .msi в Windows, хотя и достаточно грубо. Часто для работы ряда программ требуются определенные дополнительные библиотеки. Тогда пакет, требующий для работы новую библиотеку, не установится, пока не появятся необходимые ему для работы пакеты. Это называется зависимостями. В Windows также можно иногда заметить что­-то подобное. Например некоторые игры требуют присутствия в системе определенной версии DirectX, то есть они зависят от него. В Linux такая практика является намного более распространенной. Такой подход позволяет существенно уменьшить трафик, а также место на диске, но иногда усложняет сам процесс установки. Также отметим, что .deb-пакеты присутствуют в ОС GNU/Linux, основанных на Debian и Ubuntu. В других системах — другие форматы пакетов и утилиты, например в RedHat и Centos — .rpm. Но в целом идеи во многом похожие, и, например, в Centos вместо выполнения команды sudo apt install mc вы будете выполнять команду sudo yum install mc.

## **Основной способ установки программ. Использование официальных репозиториев**

Самый простой способ установить новую программу в Ubuntu — выбрать пункт меню Программы ­> Установить/Удалить. Дальше выбрать необходимую программу в каталоге, поставить галочку и нажать Применить. Все остальное будет сделано автоматически. Система сама загрузит необходимые пакеты из Интернета, установит их и добавит новый пункт в меню. Более простой способ трудно найти.  
Если вы хотите иметь больше возможностей влиять на процесс, используйте более продвинутый инструмент —­ Synaptic. Выберите Система ­> Администрирование ­> Менеджер пакетов Synaptic и получите более функциональный инструмент для работы с пакетами. В частности, вы можете устанавливать программы частично, если вам не нужна документация или еще что­-то.  
  
Кроме того, можно использовать и консольные инструменты. Например если вы хотите установить программу для записи дисков Brasero, наберите команду sudo apt-­get install brasero и она установится.  
  
У такого метода установки программ есть масса преимуществ: это удобно, вы устанавливаете уже протестированные программы, которые гарантированно будут работать на вашей системе, зависимости между пакетами будут решаться автоматически, при появлении в репозитории новых версий установленных программ вас об этом проинформируют.

## **Использование сторонних репозиториев**

Хотя в официальных репозиториях программ очень много, однако при их использовании могут возникать проблемы. Например, некоторые программы могут присутствовать в уже устаревших версиях, а некоторых просто может не быть. При этом ряд проектов имеет собственные репозитории, такие как Wine, Elisa и др. В таком случае на официальном сайте нужной вам программы должно быть описано, как такие репозитории подключить. После подключения репозитория процесс установки не отличается от уже описанного ранее.

## **Непосредственная установка из deb-пакетов**

Конечно, всегда можно загрузить готовые deb­-пакеты с помощью браузера. Существуют порталы с выложенными deb-пакетами программ, такие как GetDeb.net. Также можно загружать файлы напрямую с официального сайта нужной вам программы. Скачанные пакеты легко устанавливаются двойным кликом мышки. Минус такого подхода —­ менеджер обновлений не будет отслеживать появление новых версий установленной программы.  
  
Если вам попался пакет rpm (для систем типа Fedora, Mandriva, SUSE и т.д.), его можно сконвертировать в deb с помощью утилиты alien (устанавливается отдельно). Однако делать это рекомендуется лишь в исключительных случаях, когда пакетов deb в доступе нет (например драйверы принтеров Canon на официальном сайте выложены исключительно в rpm).

## **Установка программ с собственным инсталлятором**

Иногда программы могут распространяться с собственным инсталлятором. Это ничем не отличается от ситуации в Windows. Только здесь, распаковав tar.gz архив с дистрибутивом программы, вы вместо setup.exe увидите что­-то типа INSTALL.sh. Если этот файл не захочет запускаться, вероятно, ему не даны права на запуск. Иногда дистрибутив программы распространяется в виде самораспаковывающегося архива. В таком случае это будет просто один-единственный файл .sh, который нужно запустить. Дальше вы получите мастер, где нужно будет ответить на ряд вопросов, так же, как это делается в Windows. Так устанавливаются официальные драйверы nVidia, ATI, среда разработчика NetBeans и т.п.

## **Установка программ, которые не нуждаются инсталляции**

Есть программы, которые распространяются в виде обычного архива tar.gz (часто называются тарболлы), который достаточно куда-­то распаковать. В Windows также есть такие программы, их еще часто называют словом Portable. Устанавливать такие программы можно куда­ угодно, но стандартное место — каталог /opt. Пункты на запуск в меню вам придется добавлять вручную, для этого нужно щелкнуть правой кнопкой по заголовку меню Программы и выбрать Правка меню.

## **Установка программ из исходных текстов**

Обычно следует избегать этого способа установки программ. Но иногда такой способ установки может оказаться единственно доступным. Основное, что понадобится — средства для компиляции, для этого сначала нужно установить пакет build­-essential. Дальше нужно распаковать архив с кодами программы во временную папку. Потом нужно найти файл README или INSTALL, прочитать его и выполнить то, что там написано. Чаще установка программ таким способом ограничивается последовательным выполнениям следующих команд:

|  |
| --- |
| ./configure make  sudo make install |

В некоторых случаях последовательность команд для установки может отличаться. Кроме того, после выполнения скрипта ./configure вы можете получить сообщение о том, что в системе не установлено нужных библиотек для компиляции программы. В таком случае нужно будет установить их самому и повторить процесс. Обычно процесс компиляции занимает определенное время и напрямую зависит от мощности вашего компьютера.

## **Репозитории в UBUNTU**

Репозитории — это специальные сервера­-хранилища таких файлов. Их также можно назвать «Источниками приложений». Пользовательские компьютеры подключаются к репозиториям по сети или через интернет и при помощи специальных утилит (например Synaptic) позволяют вам увидеть, какие пакеты установлены, какие доступны для установки. Большинство утилит поддерживают простой поиск по ключевым словам и способны разбивать группы пакетов по категориям.  
  
Связка репозиторий-­утилита позволяет использовать централизованный метод установки/удаления программ, а также предоставляет удобный способ выкладывания обновлений.

В Ubuntu уже подключены необходимые репозитории, однако никто не запрещает использовать сторонние репозитории.

В репозиториях Ubuntu содержится достаточно разнообразного ПО, и зачастую вы можете найти там необходимую программу. Только если вы не нашли нужную программу в репозиториях, стоит пробовать другие пути установки.  
Версии ПО, доступные через репозитории, не всегда свежие, однако они, как правило, лучше интегрированы в Ubuntu и более стабильны.

## **Структура репозитория Ubuntu**

В Ubuntu всё программное обеспечение делится на четыре секции, называемые компонентами, чтобы отразить разницу в лицензии и уровне доступной поддержки. Пакеты распределяются по компонентам таким образом:

* Main — свободное ПО, официально поддерживаемое компанией Canonical.
* Restricted — проприетарное ПО (в основном — драйверы устройств), официально поддерживаемое компанией Canonical.
* Universe — свободное ПО, официально не поддерживаемое компанией Canonical, но поддерживаемое сообществом пользователей.
* Multiverse — проприетарное ПО, не поддерживаемое компанией Canonical.

Существует четыре основных репозитория Ubuntu.

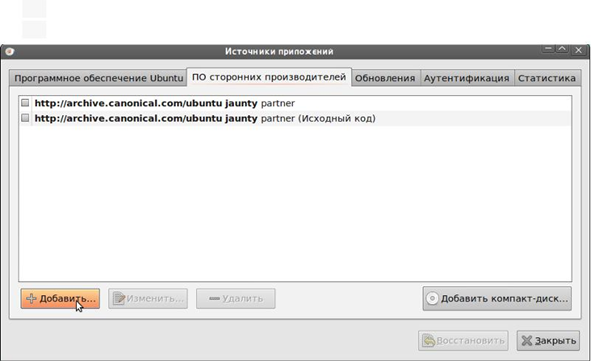
* $release — пакеты на момент выхода релиза.
* $release­security — пакеты критических обновлений безопасности.
* $release­updates — пакеты обновления системы (более поздние версии ПО, вышедшие уже после релиза).
* $release­backports — бэкпорты более новых версий ПО, которое доступно только в нестабильных версиях Ubuntu.
* partner — репозиторий, содержащий ПО компаний­ партнеров Canonical.

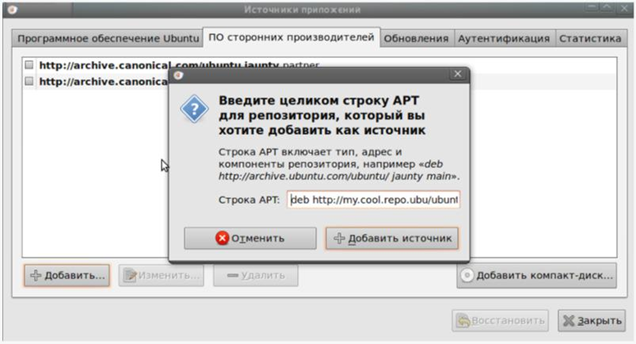
Кроме официальных, существует множество репозиториев от авторов программ и от тех, кто не поленился собрать из исходников пакет и поделиться им с другими.

## **Подключение репозитория**

Существует много сторонних репозиториев, подключив которые, вы получите доступ к дополнительному ПО. Сделать это можно как при помощи графического интерфейса, так и в консоли.  
  
Некоторые репозитории, помимо нужных пакетов, могут содержать экспериментальные сборки различного системного ПО, в том числе и ядер Linux. Поскольку версия этих экспериментальных пакетов, как правило, выше, чем установленная, Менеджер обновлений может попытаться «обновить» систему с этих репозиториев, что может повредить её. Поэтому внимательно читайте описание подключаемого репозитория и информацию в Менеджере обновлений.

## **Подключение при помощи графического интерфейса**

Для подключения репозитория выполните следующие шаги.  
  
1. Откройте Центр приложений Ubuntu.  
2. Откройте пункт меню Правка→Источники приложений…  


3. В появившемся окне выберите вкладку «Другое ПО», нажмите кнопку «Добавить».  
4. В появившемся окне заполните поле «Строка APT:» и нажмите кнопку «Добавить источник».  
  


Т.к. был подключен новый источник программного обеспечения, необходимо обновить информацию о пакетах. Появится окно с предложением это сделать. Нажмите «Обновить».

После обновления информации о пакетах окно «Источники приложений» закроется, и, скорее всего, вы получите ошибку о неподписанном источнике приложений, тем не менее, вы сможете устанавливать пакеты, содержащиеся в свежеподключенном репозитории, стандартными средствами. Для устранения ошибки неподписанного репозитория см. пункт про защиту репозиториев ниже.

## **При помощи консоли (рекомендуемый способ)**

В современных версиях Ubuntu добавлять репозиторий можно одной командой, вот пример для ppa-­репозитория:

|  |
| --- |
| sudo apt­-add­-repository ppa:ripps818/coreavc |

Системный список репозиториев содержится в файле /etc/apt/sources.list. Чтобы добавить репозиторий,­ отредактируйте файл так:

|  |
| --- |
| sudo nano /etc/apt/sources.list |

Добавьте туда APT-строку. Чем «выше» (ближе к началу файла) стоит строка, тем больший приоритет получит добавленный репозиторий.

Далее следует обновить список пакетов. Для этого выполните:

|  |
| --- |
| sudo apt­-get update |

Теперь вы можете устанавливать пакеты из нового репозитория, правда, для комфортной работы нужно импортировать в систему ключ репозитория, т.к. у вас постоянно будет появляться такое предупреждение:

|  |
| --- |
| W: Ошибка: http://site.com jaunty Release: Следующие подписи не могут быть проверены, так как недоступен открытый ключ: NO\_PUBKEY 1234567890ADBD |

## **Устройство репозитория**

Пакет (например \*.deb файл) размещается на общедоступном интернет-ресурсе (например, archive.ubuntu.com). Затем информация о пакете заносится в файл Packages, который, в свою очередь, для удобства пакуется в Packages.gz

Файлов Packages.gz может быть несколько (например, по одному для каждой архитектуры). Файл Release содержит описание репозитория в целом и ссылки на различные Packages.gz.

Общая схема работы выглядит примерно так:

1. Пользовательский компьютер подключается к репозиторию, и, при наличии защиты, проверяет его истинность (см. главу Защита репозиториев).
2. Читает файл Release, находит и скачивает необходимые Packages.gz.
3. На основе скачанных Packages.gz обновляет локальную базу данных пакетов.
4. Теперь пользовательский компьютер «знает», где находится тот или иной пакет и при необходимости легко может его скачать и установить.

## **Защита репозиториев**

Поскольку репозитории большей частью расположены в интернете, существует вероятность подмены их злоумышленником на свои, содержащие модифицированные пакеты. Таким образом, пользователь может установить модифицированный пакет и тем самым поставить безопасность системы под угрозу. Многие репозитории имеют защиту от подмены. Такая защита реализована при помощи сверки цифровых подписей репозитория и клиента. В случае, когда репозиторий имеет цифровую подпись, а пользовательский компьютер содержит открытый ключ для него, такой репозиторий считается доверенным.

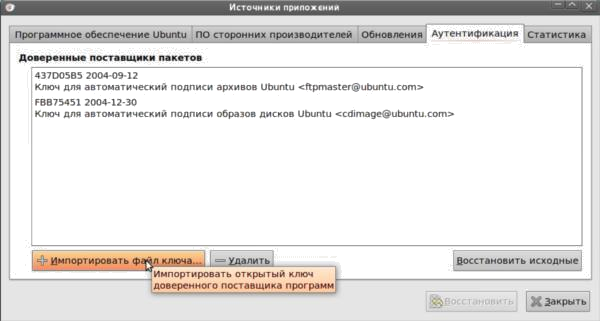
В Ubuntu по умолчанию доверенными являются репозитории на установочных дисках и основные интернет-репозитории — archive.ubuntu.com. При наличии на пользовательском компьютере нескольких подключенных репозиториев предпочтение отдается доверенным.

При подключении репозитория, защищенного цифровой подписью, вам нужно скачать (обычно с ресурса, рассказывающего про этот репозиторий, или с сервера ключей, что является более предпочтительным в любом случае) открытый ключ и добавить его в систему. Иногда для скачивания предоставляется доступный для установки пакет, который при установке сам прописывает ключ репозитория. Если вы скачиваете ключ с сайта репозитория, вы получите обычный файл с расширением .key, .gpg или другим. Добавить его в систему можно так:

|  |
| --- |
| sudo apt­-key add repo.key |

repo.key — полученный вами ключ репозитория.

Или при помощи графического интерфейса запустите «Источники приложений» (Система→Администрирование→Источники приложений), перейдите на вкладку «Аутентификация» и нажмите на кнопку «Импортировать файл ключа…»: ­ откроется диалог выбора файла. Выберите файл ключа и нажмите ОК.



Однако гораздо более предпочтительным является добавление ключа со специального защищённого сервера. Обычно, когда заходит речь о ключе, даётся его непонятный с первого взгляда буквенно­цифровой идентификатор вида 123ABCDEFG456 (строчка из произвольных цифр и букв латинского алфавита в верхнем регистре). Это ­ уникальное имя ключа. Иногда ключ описывается строчкой вида 1024R/123ABCD, тогда идентификатором является часть после слэша. Ключи преимущественно хранятся на специальных серверах, откуда любой может их получить. Ключи для репозиториев Ubuntu принято хранить на keyserver.ubuntu.com. Для получения и импортирования в систему ключа с сервера необходимо выполнить команду:

|  |
| --- |
| sudo apt­-key adv ­­--keyserver keyserver.ubuntu.com ­­recv­keys 12345678 |

Вместо keyserver.ubuntu.com можно подставить адрес другого сервера ключей, а вместо 12345678 написать идентификатор нужного вам ключа.

Чтобы разом попытаться импортировать все недостающие ключи репозиториев, выполните в консоли:

|  |
| --- |
| sudo apt-­key adv ­­recv­keys ­­--keyserver keyserver.ubuntu.com `sudo aptitude update 2>&1 | grep ­o '[0­9A­Z]\{16\}$' | xargs` |

Здесь присутствует регулярное выражение, мы научимся писать их на следующем уроке. Ряд часто встречаемых регулярных выражений и команд с их использованием  можно найти в интернете, но еще лучше уметь их писать.

## **Логи, просмотр логов**

После установки программы или в процессе ее работы могут возникнуть проблемы и ошибки, которые требуется просмотреть и проанализировать.

Для хранения логов предусмотрена директория /var/log.

При этом имеются как системные логи, так и логи для отдельных программ.

Общесистемные логи, как правило, хранятся в файле /var/log/syslog (или в /var/log/messages, например в Centos).

Сообщения ядра хранятся в /var/log/dmesg (может понадобится при добавлении драйверов-модулей ядра).

Записи в журналах обычно содержат метку времени, имя хоста, на котором выполняется описываемый процесс, и имя процесса.

Просматривать журналы можно при помощи программы постраничного вывода, например, less, искать определенные записи (например, сообщения ядра от определенного демона) можно при помощи команды grep:

Например:

|  |
| --- |
| # less /var/log/messages |

или

|  |
| --- |
| # grep "ppp" /var/log/messages | tail |

Посмотреть последние 25 строк файла можно командой tail:

|  |
| --- |
| $  tail -25 /var/log/syslog |

Без указания ключа tail выдаст последние 10 строк.

Если нужно выдавать последние строки файла по мере их появления, используйте ключ -f.

|  |
| --- |
| $ tail -f /var/log/syslog |

Чтобы прервать работу tail -f, используйте комбинацию Ctrl-C. Есть команда, аналогичная tail, только для начала файла. Это команда head. Без параметров выдает первые 10 строк указанного файла:

|  |
| --- |
| $ head filename |

**Основные элементы Linux: ядро, графическая оболочка, командный интерпретатор, UNIX way**

Разберем основные компоненты Linux-системы более подробно. Но для начала остановимся на философии UNIX, которой следуют UNIX-подобные операционные системы, в том числе и Linux.

## **UNIX way**

Философия Unix, известная как UNIX way, является подходом, которому следует архитектура UNIX-подобных систем. Есть несколько разных определений, что такое UNIX way, при этом среди них можно выделить несколько важных принципов. Среди них мы отметим:

* Утилиты — маленькие отлаженные программы, решающие только свою задачу.
* Более сложные задачи можно решать, комбинируя утилиты, запуская их с помощью конвейеров.
* Конфигурационные файлы хранятся в виде простых текстовых файлов.
* Все есть файл.

Это означает, что лучше одна программа, решающая только одну задачу, но делающая это хорошо. Благодаря использованию потоков и возможности работы с устройствами как файлами, вывод и ввод таких программ можно совмещать, добиваясь гибкости. Мы уже использовали это на примере комбинации команды apropos и grep.

Принципы UNIX вы будете использовать при работе Linux и наблюдать воочию постоянно.

## **Ядро**

Центральный элемент любой операционной системы — ядро. Ядро загружается в память при старте системы и его код выполняется процессором на привилегированном (по сравнению с пользовательскими процессами) уровне. Для пользователя эта часть системы прозрачна — не очень заметна, зато программисты активно используют функции ядра, называемые системными вызовами при написании программ, которые работают с файлами и периферийным оборудованием. Ядро  взаимодействует с оборудованием посредством встроенных или динамически подгружаемых драйверов и позволяет программистам работать с периферией как с обычными файлами. Доступ к файлам и файловым системам — также функция ядра. Вообще унифицированное понятие файла (файлом является почти любое устройство: жесткий диск, мышь, звуковая карта) — характерная для систем семейства UNIX черта, составляющая вместе с остальными ключевыми особенностями так называемый стиль UNIX way. Поскольку Linux является многозадачной и многопользовательской системой,  ядро также занимается распределением ресурсов между программами и обеспечивает разграничение доступа к ресурсам. Например, попытка процесса (программы на этапе исполнения ) обратиться за пределы своего адресного пространства немедленно приведет к его аварийному завершению системой без ущерба для остальных процессов. Переключение между процессами и диспетчеризация ввода-вывода также важные функции ядра.

## **Системные утилиты, оболочка**

Кроме ядра, важной частью системы являются системные утилиты. В Linux они представлены стандартным набором утилит, разработанных в  рамках проекта GNU. Среди них есть команды администрирования пользователей, утилиты для управления файлами и процессами, а также утилиты для обработки текста. Большинство утилит не имеет графического интерфейса; они  выдают и принимают на вход информацию в виде обычного текста, поэтому в UNIX существует множество стандартных утилит для обработки и форматирования текстов. Для запуска системных утилит и других программ служит командный интерпретатор или, как его еще называют, оболочка (shell), основной инструмент системного администратора Linux. Интерпретатор — своего рода язык программирования, позволяющий создавать сложные сценарии (скрипты), также он используется интерактивно для последовательного ввода команд. Одна из составляющих стиля UNIX way — каждая утилита предназначена для выполнения отдельной узкой задачи и способна выполнять ее максимально гибко при помощью большого набора опций. Кроме того, утилиты способны взаимодействовать друг с другом, последовательно передавая  друг другу результаты своей работы через конвейер. Это одна из наиболее мощных составляющих UNIX way, которая будет подробно описана далее.

## **Демоны**

Кроме системных утилит, есть класс специальных программ, которые называют демонами. Типичный  демон — программа, которая обслуживает сетевой сервис; обычно она уже запущена и находится в ожидании запросов. Примеры демонов, с которыми мы познакомимся дальше — веб-сервер, почтовый сервер, сервер доменных имен. Обратите внимание, что здесь слово «сервер» используется в значении «программа, предоставляющая сервис».

## **Приложения пользователя, графический интерфейс X Window System**

Последний класс программ, самый многочисленный — пользовательские приложения. Это различные прикладные программы: офисные приложения (LibreOffice), браузер (Mozilla Firefox), почтовый клиент (Thunderbird). Прикладные программы не являются частью системы, поэтому их набор и версии могут отличаться для разных дистрибутивов. Среди прикладных программ отдельно стоит выделить графические, которые используют систему графического интерфейса X Window System. Графическая система появилась в UNIX где-то с середины 80-x. В отличие от графической среды Microsoft Windows, в UNIX графическая среда не является составной частью ядра системы, это отдельный набор программ и библиотек, от которого при желании можно отказаться. Обычно на серверах  вообще не используют графическую оболочку, выполняя все задачи с использованием интерпретатора команд. X Window System изначально задумывалась как сетевая система клиент-серверного типа. Это подразумевает физическое разделение клиентов — программ, которые отправляют запросы на вывод графики, и X-сервера, программы, которая запущена на стороне пользователя и непосредственно занимается выводом графики на графическом устройстве клиента, обслуживая запросы X-клиентов на отрисовку графических примитивов. Это немного необычное расположение клиента и сервера, когда сервер работает на компьютере пользователя, а клиенты находятся на удаленной системе. Клиент-серверная архитектура X Window и наличие открытого стандартного протокола позволяет запускать графические приложения Linux на широком круге устройств и операционных систем. X-сервер как стандартное приложение  существует почти в любой Unix/Linux системе, также доступны X-серверы для ОС семейства Windows. Linux предоставляет великое множество графических окружений, хотя почти все они основаны на использовании X Windows. Их разнообразие проявляется в компонентах графической подсистемы: оконных менеджерах и графических средах. Кратко остановимся на самых распространенных.

## **Оконные менеджеры, окружения рабочего стола**

Оконные менеджеры — программы, которые управляют размещением на рабочем столе окон вновь запущенных приложений и позволяют управлять их размером и текущим расположением: менять размер, перетаскивать, схлопывать и разворачивать. Оконные менеджеры определяют внешний вид окон, оконное меню, рамки, кнопки управления, тени и так далее.  Наиболее популярные на сегодняшний день менеджеры окон — Enlightenment, FVWM, Openbox и еще много других. Как правило, оконные менеджеры используются в составе так называемых окружений рабочего стола. Окружение рабочего стола включает в себя набор специальных приложений для организации рабочего стола: панели запуска, меню доступа к приложениям, индикаторы событий, специальные поисковые элементы. Кроме того, в состав окружения обычно входит набор приложений пользователя, плотно интегрированных с остальными элементами. Современные окружения рабочего стола максимально задействуют ресурсы графического адаптера и сопровождают работу пользователя 3D-анимацией и различными  визуальными эффектами. Среди популярных на сегодняшний день окружений стоит отметить KDE4, Gnome 3 и Unity. Последняя как раз используется в основной десктоп-версии дистрибутива Ubuntu. Помимо основной версии, сообщество разработчиков Ubuntu поддерживает специальные дистрибутивы, основанные на альтернативных графических средах — KDE, XFCE и LXDE. Среди популярных альтернативных сборок Ubuntu: Kubuntu (с KDE), Xubuntu (XFCE) и Lubuntu (LXDE). Если на вашем компьютере мало ресурсов, лучше вместо основной версии Ubuntu с Unity использовать дистрибутив Lubuntu, использующий графический интерфейс LXDE, которому достаточно для работы 512 Мб RAM. LXDE использует привычную концепцию меню программ в стиле Windows XP и мы не будем на нем останавливаться. В курсе мы кратко расскажем о работе с Unity. Эта достаточно новая графическая среда может показаться непривычной после интерфейса Microsoft Windows или других графических сред Linux, поэтому далее будут описаны основные элементы рабочего стола Unity и приведен пример пользовательской  сессии.