Яблочкин К.В.. «Введение в Linux для начинающих с нуля» или «Несколько записей по GNU/Linux для начинающих с нуля. И шпаргалка для опытных пользователей» Черновик!

Оглавление

«Предупреждение»

Данный файл создавался без определённых целей. Я (Яблочкин Константин Владимирович, адрес эл. почты: qzfq1989@gmail.com) пытался создать хороший документ, но мне это, возможно, не удалось. Здесь могут быть ошибки. Если вы используете этот файл, вы сами отвечаете за последствия и я ни за что не отвечаю. По состоянию на сентябрь 2023 года у меня 20 лет пользовательского стажа в системах Linux (GNU/Linux). Последняя правка этого файла: 2024-02-01 (01 февряля 2024 года «нашей» эры). Я считаю, что и некоторые опытные пользователи Linux могут найти здесь что-то новое для себя. Не исключено, что я согласен не со всем, что написано в этом файле (со временем точка зрения могла измениться или исчезнуть, а перечитывать и корректировать весь файл я не хочу).

Лицензия распространения файла: делайте с ним что хотите бесплатно и за последствия я не отвечаю.

Конец раздела «предупреждение»

Обновление от 2024-02-01

(добавились ссылки, удалился фрагмент статьи от acetone)

И мне кажется, что файл получился местами не оень плохой. Может быть кто-то его доработает (просьба за основу брать файл с расширением «.odt» и дорабатывать по возможности «качественно» в каком-то смысле) и выложит. (У меня, возможно, не будет времени заниматься доработкой.)

И кто не понял, LMDE6 = Linux Mint Debian Edition 6.

Обновление от 2024-01-31

1. Первое

Используйте следующую команду (в bash) для получения списка всех доступных команд:

```
$ compgen -c
```

2. Второе

После установки системы LMDE6 я рекомендую её запустить и выполнить следующую команду (доллар в начале не вводить; всё, что идёт после слова, начинающегося на символ «#», интерпретатор bash проигнорирует):

```
$ sudo dpkg-divert --add --local --rename /etc/ssh/sshd_config ; sync # защита от случайного запуска sshd с небезопасными настройками в случае установки пакета openssh-server (оно его автоматически пытается запустить, а без файла конфигурации он не запустится).
```

3. Третье

По состоянию на 2024-01-31 я рекомендую начинающим пользователям использовать «Linux Mint Debian Edition 6» (это не Linux Mint; в отличие от Linux Mint, здесь внутренности от Debian 12 (а в Linux Mint внутренности от Ubuntu)).

4. Четвёртое

Файл создан в программе LibreOffice Writer. Ещё из бесплатного для офиса в LMDE6 есть: Calligra Sheets, Calligra Words, Calligra Stage, Abiword, Gnumeric, LibreOffice Calc, LibreOffice Impress, Libreoffice Draw. Может быть, есть что-то ещё (в Debian 12, на котором основан LMDE6, более 60 000 пакетов, а в подготовке его выпуска, согласно материалам из интернета, участвовали примерно 6500 человек).

Конец раздела «Обновление от 2024-01-31»

"Введение"

Операционные системы GNU/Linux иногда называется Linux. GNU (расшифровывается «GNU is Not Unix») - Unix-подобная операционная система, имеющая ядро GNU Hurd. Ядро — компонент операционной системы, который работает в привилегированном режиме и управляет системой. Ядро управляет запущенными процессами и устройствами. Unix - операционная система, некоторые её варианты продолжают развиваться, например, FreeBSD, NetBSD, OpenBSD. В сборках, называемых дистрибутивами Linux, взяты программы из проекта GNU, а ядро Hurd заменено ядром Linux. Кроме этого, в сборку входят программы/библиотеки, не относящиеся к проекту GNU.

Список популярных дистрибутивов GNU/Linux можно посмотреть здесь: https://distrowatch.com/ . Там же есть поиск дистрибутива по критериям (например, можно найти дистрибутивы, поддерживающие 32-битные процессоры): https://distrowatch.com/search.php

По рейтингу distrowatch, одни из самых популярных дистрибутивов - MX Linux и Linux Mint. Я обычно использую дистрибутив Linux Mint. (устаревшая информация: «Его могу рекомендовать как подходящий для начинающих пользователей».) **Обновление**: по состоянию на 2024-01-31 я рекомендую начинающим пользователям использовать «Linux Mint Debian Edition 6» (это не Linux Mint; в отличие от Linux Mint, здесь внутренности от

Debian 12 (а в Linux Mint внутренности от Ubuntu)). После установки системы LMDE6 я рекомендую её запустить и выполнить следующую команду:

```
$ sudo dpkg-divert --add --local --rename /etc/ssh/sshd_config ; sync # защита от случайного запуска sshd с небезопасными настройками в случае установки пакета openssh-server (оно его автоматически пытается запустить, а без файла конфигурации он не запустится).
```

В GNU/Linux есть несколько окружений рабочего стола: Xfce, Gnome, Kde, LXDE, LXQt, Mate, Cinnamon, другие. Окружение рабочего стола - это файловый менеджер, рабочий стол, панели, кнопка "Пуск" (точнее, аналогичная ей), некоторые программы (например, эмулятор терминала xfce4-terminal входит в комплект Xfce).

Xfce - используемый мной легковесный рабочий стол. В Linux Mint вариант с окружением рабочего стола Xfce отмечен как самый стабильный.

Для некоторых действий в Linux можно пользоваться командной строкой (т.е. работать в терминале, в текстовом режиме). Для начала предлагается запомнить некоторые команды командной строки, возможности работы с ними и некоторые факты об операционных системах GNU/Linux.

"Среда для изучения GNU/Linux"

Можно изучать GNU/Linux через виртуальную машину. Для создания тестовой среды достаточно включить аппаратную виртуализацию (Intel VT-x или AMD-V) в BIOS или UEFI, установить VirtualBox, создать виртуальную машину с 2 (минимум - около 1) GiB оперативной памяти и виртуальным жёстким диском размером 14 GiB. 1 GiB равен 2^{30} байт, что равно 1073741824 байт (примерно равно 10^9 байт).

При работе в виртуальной машине можно случайно сломать установленную на виртуальную машину операционную систему (ОС). От этого не перестанет работать основная ОС на физическом компьютере. Виртуальную машину в случае поломки можно удалить и пересоздать. Кроме того, можно создать снимок (snapshot) виртуальной машины (VM, Virtual Machine) в исправном (включенном или выключенном) состоянии и в случае поломки восстановить виртуальную машину из этого снимка.

При наличии функции аппаратной виртуализации рекомендуется установить последнюю версию VirtualBox и установить последнюю версию Linux Mint (вариант Xfce) на виртуальнюю машину в VirtualBox. Если аппаратная виртуализация недоступна, можно установить VirtualBox версии 6.0.24 (https://www.virtualbox.org/wiki/Download_Old_Builds_6_0) и установить MX Linux i686 (оно же x32, x86) на виртуальную машину. Или установить на виртуальную машину debian i386

КОНЕЦ "Среда для изучения GNU/Linux"

КОНЕЦ "Введение"

"Базовые понятия и обозначения"

Натуральное число: число из последовательности 1, 2, 3, 4, … (их бесконечно)

Целое число: число из последовательности 0, 1, -1, 2, -2, 3, -3, 4, -4, … (здесь есть 0, ряд натуральных чисел и ряд чисел, противоположных натуральным; других

элементов здесь нет)

Конечная последовательность элементов (кортеж): набор из N элементов, где N - натуральное
число или 0. Элементы последовательности имеют номера от 1 до N и индексы от 0 до N-1. Индекс = номер - 1. Записывается списком элементов через запятую в скобках. Например, (1,5,6) - последовательность из 3 элементов 1, 5 и 6. () - пустая последовательность (N = 0).

Пара - последовательность из двух элементов. Например, (1,2) или ((1,2,3),(4,5,6)). Второй элемент последней пары - кортеж (последователььость) (4,5,6).

Множество — набор элементов, для которого определено, какие элементы в него входят, и не определён порядок элементов. Конечные множества можно записывать списком всех элементов в фигурных скобках. Например, {1,2} = {2,1} = множество их двух натуральных чисел (числа 1 и 2).

Число A, умноженно на B (произведение A и B), записывается так: A * B

Число A в степени B записывается так: A^B или A**B. Например, 2^10 = 2 ** 10 = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 (десять двоек, 9 знаков *) = 1024

Бит: элемент множества {0,1} (это множество из двух элементов: 0 и 1)

Байт как число: элемент множества {0, 1, 2, 3, …, 255} (целые числа от 0 до 255)

Байт как последовательность бит (октет): последовательность из 8 бит

Позиционная система счисления по основанию q (q — целое число, не меньше 2), она же - q-ичная система счисления: способ записи целых чисел. В записи используются: знак «-» (минус) и цифры с числовыми значениями от 0 до q-1. Для обозначения целых значений от 0 до 9 можно использовать цифры от 0 до 9. Для обозначения целых значений от 10 до 35 можно использовать латинские буквы от A до Z (A = 10, B = 11, C = 12, ..., Z = 35). Для обозначения чисел больше 35 можно ввести другие символы (цифры). Каждому целому числу
соответствует одна запись без лишних лидирующих нулей (нелишний лидирующий нуль есть
только в записи числа 0 (это запись «0»)). Обычно для записи чисел используется позиционная система счисления по основанию 10 (десятиричная система счисления, q = 10). Примеры записи целых чисел при q=10: «0», «12», «-104».

Алгоритм перевода целого неотрицательного числа из десятичного представления в представление в q-ичной системе счисления (q - целое число, не меньше 2). Выход (результат работы) алгоритма - последовательность цифр от старшей цифры к младшей (это порядок цифр big endian, он используется в популярной десятиричной системе счисления).
Если нужно перевести отрицательное число x из десятиричной системы счисления в q-ичную, можно
взять его модуль (неотрицательное целое число -x), перевести его в q-ичную систему счисления
и приписать к результату слева «-» (минус).
Алгоритм:
1. Присвоить x = исходное число.

2. Присвоить СТРОКА = «» (пустая последовательность символов).
3. Если x = 0, то результат = «0» и закончить выполнение алгоритма.
4. Разделить x на q с остатком. Результат деления - представление x = a * q + b, где b - целое число от 0 до q-1, a - целое число, знак * - умножение.
5. Приписать цифру с числовым значением b слева к СТРОКА (изменить значение переменной СТРОКА). Смотрите комментарий после алгоритма.
6. Если a = 0, то результат = СТРОКА и закончить выполнение алгоритма.
7. Присвоить x = a, перейти к шагу 4 этого алгоритма.

Комментарий: цифры с числовыми значениями 0-9 - это цифры 0-9 (0, 1, 2, ..., 9). Цифры со значениями 10, 11, 12, ... 35 - это A, B, C, D, E, F, G, H, I, ..., Z (A = 10, B = 11, ..., Z = 35). При q <= 36 (q меньше или равно 36) используются цифры со значениями не более 35.

При q > 36 использовать другие цифры (латинского алфавита не хватит).

Пример: перевести 243 в 16-ричную систему счисления, получить «F3».

Алгоритм перевода целого неотрицательного числа из q-ичной системы счисления в десятиричную:

1. Взять СТРОКА = последовательность цифр в исходном представлении числа.
2. В СТРОКА каждую цифру замения её числовым значением (например, послеловательность "AB1" перейдёт в последовательность (10, 11, 1))
3. Каждому числу в СТРОКА приписать индекс, справа налево, начиная с нуля (например, (10, 11, 1) перейдёт в ((10, 2), (11, 1), (1, 0)). Здесь это последовательность из трёх пар, где первая пара - пара из элементов 10 и 2).
4. Каждое число (первые элементы пар в СТРОКА) домножить на $q^{индекс}$ (q в степени индекс). Например, при q=16 из ((10, 2), (11, 1), (1, 0)) получим ((2560, 2), (176, 1), (1, 0)).
5. РЕЗУЛЬТАТ = сумма всех полученных чисел, кроме индексов (например, (2560+176+1) = 2737). Закончить алгоритм.

Пример: Перевести «1A» из 36-ричной, получить 46. «K1» (q = 22) перейдёт в 441. «F4» при q=16 перейдёт в 244.

Битовое представление байта как числа: последовательность цифр представления этого числа (байта) в двоичной системе счисления, дополненная нулями слева до последовательности (строки) длины 8.

"Единицы измерения количества данных"

Таблица 1: Единицы количества данных по степеням десятки

Префикс (по-русски)	Префикс (по-английски)	Единица (рус.)	Единица (англ.)	Сокр. (рус.)	Сокр. (англ.)	Через предыдущее	То же, степ. 10	Через Б, степ. 10	Через Б, степ. 1000
-	-	1 Байт	1 Byte	1 Б	1 B	-	-	10^0 Б	1000^0 Б
K	K	1 КилоБайт	1 KiloByte	1 КБ	1 KB	1000 Б	10^3 Б	103 Б	1000^1 Б

Префикс (по-русски)	Префикс (по-английски)	Единица (рус.)	Единица (англ.)	Сокр. (рус.)	Сокр. (англ.)	Через предыдущее	То же, степ. 10	Через Б, степ. 10	Через Б, степ. 1000
М	M	1 МегаБайт	1 MegaByte	1 МБ	1 MB	1000 КБ	10^3 КБ	10^6 Б	1000^2 Б
Г	G	1 ГигаБайт	1 GigaByte	1 ГБ	1 GB	1000 МБ	10^3 МБ	10^9 Б	1000^3 Б
Т	T	1 ТераБайт	1 TeraByte	1 ТБ	1 TB	1000 ГБ	10^3 ГБ	10^{12} Б	1000^4 Б
П	P	1 ПетаБайт	1 PetaByte	1 ПБ	1 PB	1000 ТБ	10^3 ТБ	10^{15} Б	1000^5 Б
Э	E	1 ЭксаБайт	1 ExaByte	1 ЭБ	1 EB	1000 ПБ	10^3 ПБ	10^{18} Б	1000^6 Б
З	Z	1 ЗеттаБайт	1 ZettaByte	1 ЗБ	1 ZB	1000 ЭБ	10^3 ЭБ	10^{21} Б	1000^7 Б
Й	Y	1 ЙоттаБайт	1 YottaByte	1 ЙБ	1 YB	1000 ЗБ	10^3 ЗБ	10^{24} Б	1000^8 Б

Таблица 2: Единицы количества данных по степеням двойки

Префикс (по-русски)	Префикс (по-английски)	Единица (рус.)	Единица (англ.)	Сокр. (рус.)	Сокр. (англ.)	Через предыдущее	То же, степ. 2	Через Б, степ. 2	Через Б, степ. 1024
-	-	1 Байт	1 Byte	1 Б	1 B	-	-	2^0 Б	1024^0 Б
Ки	Ki	1 КибиБайт	1 KibiByte	1 КиБ	1 KiB	1024 Б	2^{10} Б	2^{10} Б	1024^1 Б
Ми	Mi	1 МебиБайт	1 MebiByte	1 МиБ	1 MiB	1024 КиБ	2^{10} КиБ	2^{20} Б	1024^2 Б
Ги	Gi	1 ГибиБайт	1 GibiByte	1 ГиБ	1 GiB	1024 МиБ	2^{10} МиБ	2^{30} Б	1024^3 Б
Ти	Ti	1 ТебиБайт	1 TebiByte	1 ТиБ	1 TiB	1024 ГиБ	2^{10} ГиБ	2^{40} Б	1024^4 Б
Пи	Pi	1 ПебиБайт	1 PebiByte	1 ПиБ	1 PiB	1024 ТиБ	2^{10} ТиБ	2^{50} Б	1024^5 Б
Эи	Ei	1 ЭксбиБайт	1 ExbiByte	1 ЭиБ	1 EiB	1024 ПиБ	2^{10} ПиБ	2^{60} Б	1024^6 Б
Зи	Zi	1 ЗебиБайт	1 ZebiByte	1 ЗиБ	1 ZiB	1024 ЭиБ	2^{10} ЭиБ	2^{70} Б	1024^7 Б
Йи	Yi	1 ЙобиБайт	1 YobiByte	1 ЙиБ	1 YiB	1024 ЗиБ	2^{10} ЗиБ	2^{80} Б	1024^8 Б

КОНЕЦ "Единицы измерения количества данных"
КОНЕЦ "Базовые понятия"

"Как вводить команды"

Для ввода команд нужно открыть эмулятор терминала или переключиться в виртуальную консоль. Эмулятор терминала можно найти в меню программ (нажать кнопку, аналогичную кнопке "Пуск", найти эмулятор терминала). На виртуальную консоль можно переключиться нажатием одной из комбинаций: от Ctrl+Alt+F1 до Ctrl+Alt+F12 (обычно до F6). Для возврата в графический режим жмём Alt+F7 или Alt+F1 (или комбинацию с F с другим номером).

В командной строке отображается некоторая информация, затем символ «$» (знак доллара, для обычного пользователя) или символ «#» (решётка, для суперпользователя (root)), затем пробел.

Далее можно вводить команду с заданием переменных окружения и аргументов. Сначала идут задания переменных окружения в виде пар ИМЯ=ЗНАЧЕНИЕ, потом команда, потом аргументы. Всё это разделяется пробелами.

Есть 6 типов команд:

1. Встроенные команды "bash".
2. Псевдонимы (aliases).
3. Исполняемые программы, находимые в путях, заданных в переменной окружения "PATH".
4. Определённые пользователем (в том числе, заданные в системных файлах настройки "bash") функции "bash".
5. Исполняемые порграммы, указанные через абсолютный путь (смотрите далее) к исполняемому файлу.
6. Исполняемые программы, указанные через относительный путь (смотрите далее) к исполняемому файлу. Нельзя использовать непосредственно имя файла из текущего каталога (папки, директории).

Пример:

```
$ LC_ALL=C ls -lapt /dev
```

(Знак доллара вводить не надо, он будет в терминале обычного пользователя (не суперпользователь root) после информационной строки. Если команду нужно запускать от имени root, вместо знака $ ставится знак #. Его тоже вводить не надо и он тоже появляется на экране.)

Здесь указана одна переменная окружения (LC_ALL со значением «C»), введена команда "ls" и задано два аргумента: "-lapt" и "/dev/".

(Переменная окружения LC_ALL, выставленная в значение "C", говорит программе игнорировать все остальные

настройки региона и использовать английский язык для вывода информации и прочие настройки,

соответствующие "C" (формат времени и другое).)

Аргументы, начинающиеся с знака "минус" ("-"), являются модификаторами поведения команды (их называют флаги).

Один из стандартных подходов - это использование одного знака "-" перед однобуквенными модификаторами

и двойного знака "минус" ("--") перед словесными модификаторами. В предыдущем примере команде "ls"

передан аргумент "-lapt", который она воспринимает как 4 однобуквенных модификатора ("l", "a", "p", "t").

Указанная команда эквивалентна следующей:

```
$ LC_ALL=C ls -l -a -p -t /dev/
```

Опция "-a" команды "ls" является эквивалентом опции "--all". Поэтому можно то же самое сделать так:

```
$ ls -lpt --all /dev
```

Выполняемая программа/команда может получать данные, вводимые пользователем в терминале, через чтение потока stdin.

Данные, записанные программой в stdout и stderr, попадают в терминал (но можно перенаправить).

Код завершения

Каждая команда при завершении завершается с кодом завершения (return code, exit code). Через этот код команда может передавать информацию об успешности своего выполнения. Код 0 означает успешное выполнение команды, коды больше 0 означают неуспешное выполнение команды. Если запускать конвейер (несколько команд с pipe, т.е. перенаправление stdout в stdin, то результатом выполнения всей команды будет код завершения последней подкоманды (той, что идёт после последнего символа '|')).

Пример:

```
$ false | true
```

вернёт success (код завершения 0), т.к. false всегда возвращает неудачу/failure (не 0), true всегда возвращает успех/success (код завершения 0) и берётся код из последней части ("true").

Пример:

```
$ true | true | false
```

вернёт неуспех/failure (не 0), т.к. false ео вернёт. Аналогично с выполнение команд списком через точку с запятой: кодом завершения будет код завершения последней команды. Пример:

```
$ false ; true; false
```

— код завершения не-0 (неуспех/failure).

КОНЕЦ "Код завершения"

В некоторых случаях если программе указать имя файла "-", она использует stdin или stdout вместо файла с именем

"-". Если нужен файл с этим именем, можно писать "./-" или указывать абсолютный путь (пример: "/home/user/dir1/-")

Перенаправление вывода и ввода команд

Каждый работающий процесс имеет стандартный поток ввода (файловый дескриптор 0, stdin), стандартный поток вывода (файловый дескриптор 1, stdout) и стандартный поток ошибок (файловый дескриптор 2, stderr). Основную информацию для пользователя (не об ошибках) принято выводить в stdout. При обычном запуске направленный программой в stdout текст (точнее, поток байт) появляется в терминале. Его можно перенаправить на ввод другой программе с использованием "pipe" (pipe = канал), если использовать символ вертикальной черты («|»).

Пример:

```
$ man -k ls | less -i
```

КОНЕЦ "Перенаправление вывода и ввода команд"

КОНЕЦ "Как вводить команды"

"Переменные среды (окружения) и переменные bash"

Переменные среды (окружения) - это строки вида ИМЯ=ЗНАЧЕНИЕ, доступные выполняемым программам.

При порождении дочерних процессов переменные среды наследуются от родительского процесса. В

интерпретаторе bash есть экспортируемые и неэкспортируемые переменные. Экспортируемые переменные

будут переменными окружения (среды) в запускаемых из bash программах. Чтобы сделать перемнную bash

экспортируемой, можно использовать команду:

```
$ export ИМЯ_ПЕРЕМЕННОЙ
```

Например,

```
$ LC_ALL=C
$ export LC_ALL
```

Того же результата можно добиться одной командой:

```
$ export LC_ALL=C
```

Можно переопределить несколько переменных среды для одной запускаемой команды, указав несколько пар `ИМЯ=ЗНАЧЕНИЕ` перед командой. Пример:

```
$ LC_ALL=C man bash
```

КОНЕЦ "Переменные среды (окружения) и переменные bash"

"Подстановки с тильдой (~)"

"bash" заменяет отдельное "~" на абсолютный путь к домашнему каталогу текущего пользователя, а

"~ИМЯ_ПОЛЬЗОВАТЕЛЯ" - на абсолютный путь к домашнему каталогу пользователя с именем "ИМЯ_ПОЛЬЗОВАТЕЛЯ".

КОНЕЦ "Подстановки с тильдой (~)"

"Папка ~/bin"

В папку ~/bin можно положить исполняемые файлы. Если при входе пользователя эта папка существует, она

будет добавлена в переменную среды PATH и можно будет использовать команды с именами такими же, как

имена исполняемых файлов в ~/bin

Вот этот код из файла ~/.profile добавляет папку ~/bin :

```
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

(см. `man test`, `man [`, `help test`, `help [`)

КОНЕЦ "Пака ~/bin"

"Базовые команды"

Команда cd

"cd" (Change Directory) заменяет текущий каталог интерпретатора команд каталогом, указанным в первом аргументе. Например,

```
$ cd dir1
```

заставляет "bash" перейти в каталог "dir1", находящийся в текущем каталоге. "cd" - встроенная команда "bash". "cd" без аргументов:

```
$ cd
```

переводит "bash" в домашнюю папку пользователя. Того же резальтата можно добиться так:

```
$ cd ~
```

или

```
$ cd ~ИМЯ_ПОЛЬЗОВАТЕЛЯ
```

КОНЕЦ "cd"

Команда ls

Команда ls выводит список содержимого указанных в аргументах каталогов, информацию об указанных в аргументах файлах и других
объектах файловой системы. Если аргументов, отличных от модификаторов поведения, нету, выводится содержимое текущего каталога
(текущего каталога для процесса интерпретатора вводимых команд; часто интерпретатором команд является программа "bash"
(Bourne-Again Shell)).
Примеры:

```
$ ls dir1
```

- вывести содержимое каталога dir1

```
$ ls -a dir1
```

- вывести содержимое каталога dir1, включая скрытые объекты файловой системы (т.е. те объекты, имена
которых начинаются на точку ('.'))

```
$ ls -d dir1
```

- вывести информацию о каталоге dir1

```
$ ls -ld dir1
```

вывести ту же информацию в длинном формате

```
$ ls -l symlink1
```

- вывести информацию о символический ссылке symlink1 в длнном формате (включая путь к объекту, на который указывает
эта ссылка)

```
$ ls -l symlinkToDir1/
```

- вывести содержимое каталога (в длинном формате), на который указывает ссылка symlinkToDir1

```
$ ls -ld symlinkToDir1/
```

- вывести в длинном формате информацию о цели символической ссылки symlinkToDir1 (если она ссылается на каталог)

```
$ ls -Hd symLink1
```

- вывести информацию о цели символической ссылки symLink1

КОНЕЦ "Команда ls"

"Команда cat"

```
   $ cat имя_файла1 ... имя_файлаN
```

выводит на stdout по очереди содержимое всех файлов с именами "имя_файла1", ..., "имя_файлаN".

Если таких имён нету ("cat" без аргументов), то данные из stdin выводятся на stdout. Имя "cat" - сокращение

от conCATenation (сцепление, объединение последовательностей).

КОНЕЦ "Команда cat"

"Команда cp"

```
   $ cp путь1 путь2
```

создаёт файл "путь2" с содержимым из файла "путь1". Например,

```
  $ cp 2.txt /home/user/dir/4.txt
```

скопирует файл "./2.txt" в папку "/home/user/dir" под именем "4.txt". Чтобы "cp" копировало каталоги рекурсивно, используем ключ "-r":

```
  $ cp -r /home/user/dir1 /home/user/backup/dir1
```

Чтобы скопировать несколько файлов и/или каталогов в один каталог под исходными именами, указываем

сначала копируемые объекты, потом каталог-назначение со знаком "слэш" ("/") в конце:

```
  $ cp -r /home/user/dir1 ./2.txt /home/user/backup/
```

КОНЕЦ "Команда cp"

"Команды less и man"

Примеры:

```
$ cat /var/log/syslog | less -i
$ less -i /var/log/syslog
```

(эти две команды открывают один и тот же файл)

```
$ man -k ls | less -i
```

запустит процесс "man" с аргументами "-k" и "ls" и запустит процесс "less" с аргументом "-i". Поток stdout процесса "man" будет связан с потоком stdin процесса "less". (Поток stdout процесса "less" будет связан с текущим терминалом, как и поток stdin процесса "man").

Команда less отображает текст, полученный ей через её stdin, так же, как его отображает "man" при просмотре одной страницы ("man -k ИМЯ" - это поиск, а не просмотр, "man ИМЯ" — это просмотр). Можно нажать "q" для выхода, двигаться по тексту стрелками "Вверх" и "Вниз", ВЛЕВО и ВПРАВО и кнопками PgUp, PgDn, пробел. Можно искать текст через "/", затем "САМ_ТЕКСТ" (или регулярное выражение), затем Enter.

Команда

```
$ man ИМЯ
```

находит в одном из разделов руководств man страницу с именем ИМЯ и отображает её так же, ка кэто делает ls (доступен поиск по "/", РЕГУЛЯРНОЕ_ВЫРАЖЕНИЕ_ИЛИ_ПРОСТО_СТРОКА, Enter; доступен выход по кнопке q; доступна прокрутка через ВВЕРХ, ВНИЗ, ВЛЕВО, ВПРАВО, PgUp, PgDn, пробел, Home, End).

Команда

```
$ man НОМЕР ИМЯ
```

отображает страницу документации man с именем ИМЯ из раздела с номером НОМЕР.

Номера справочных разделов и описание их содержимого (см. `$ man man`):

1. Исполняемые программы или команды оболочки (shell)

2. Системные вызовы (функции, предоставляемые ядром)
3. Библиотечные вызовы (функции, предоставляемые программными библиотеками)
4. Специальные файлы (обычно находящиеся в каталоге /dev)
5. Форматы файлов и соглашения, например о /etc/passwd
6. Игры
7. Разное (включает пакеты макросов и соглашения), например man(7), groff(7)
8. Команды администрирования системы (обычно, запускаемые только суперпользователем)
9. Процедуры ядра [нестандартный раздел]

"Отдельно о команде man"

"man" - стандартный для Unix способ получения справки (manual) о команде или чём-то ещё. "man" используется и в GNU/Linux. Для получения справки по команде "ls", пишем:

```
$ man ls
```

Пока запущен man, можно прокручивать текст стрелками ↑ и ↓, стрелками ← и → или кнопками: PgUp, PgDn, пробел, Home, End.

Можно искать текст, нажав "/", набрав искомый текст и нажав Enter (вместо такста можно использовать регулярное выражение (regular expression, не путать с шаблонами — wildcards)).

Выйти из "man" можно кнопкой "q". Команда:

```
$ man -k регулярное_выражение
```

«Ищет в кратких описаниях справочных страниц ключевые слова и показывает любые совпадения». Например:

```
$ man -k ls
```

выдаёт много строк, включая такую:

```
ls (1) - list directory contents
```

Это значит, что в разделе 1 руководств "man" есть страница с именем "ls". Её можно получить, набрав:

```
$ man 1 ls
```

Если просто набрать "man ls", будет выдана одна из страниц "ls", если такая есть (страницы руководств «man» с одинаковыми именами могут быть в разных разделах).

КОНЕЦ "Отдельно о команде man"

КОНЕЦ "Команды less и man"

Команда mkdir

Команда "mkdir ИМЯ" создаёт каталог с именем ИМЯ. Например,

```
$ mkdir dir1
```

(отноистельный путь) и

```
$ mkdir /home/user/dir2
```

(абсолютный путь).

Можно создать каталог со всеми несуществующими родительскими каталогами:

```
$ mkdir -p 1/2/3/4
```

создаст каталоги 1, 1/2, 1/2/3, 1/2/3/4, если ни одного из них ещё нет.

Фрагмент man-страницы mkdir:

```
   -p, --parents
              no error if existing, make parent directories as
needed
```

КОНЕЦ "Команда mkdir"

Команда info

В проекте GNU есть своя программа просмотра документации - "info". Для просмотра страницы документации "grub" можно ввести команду:

```
$ info grub
```

По тексту можно перемещаться стрелками "Вверх" и "Вниз", стрелками <-- и --> кнопками PgUp и PgDn (page up, page down), кнопками пробел, Home, End. При наведении курсора на ссылку, можно нажать Enter и перейти к соответствующему тексту info. Для выхода из info можно нажать кнопку "q".

КОНЕЦ "Команда info"

Команда help

Для некоторых встроенных команд и ключевых слов "bash" доступна справка через "help". Например, команда

```
$ help cd
```

выдаёт текст:

```
   cd: cd [-L|[-P [-e]] [-@]] [каталог]
     Change the shell working directory.
     …
```

КОНЕЦ "Команда help"

Команда mv

```
$ mv путь1 путь2
```

перемещает(move)/переименовывает файл "путь1", чтобы он был доступен по пути "путь2". Например,

```
$ mv 2.txt /home/user/backup/2.txt
```

перемещает файл "2.txt" из текущей папки в папку "/home/user/backup".

```
$ mv путь1 ... путьN путь_к_папке/
```

— перемещение N объектов в папку "путь_к_папке", аналогично команде "cp":

```
$ mv 2.txt 3.txt /home/user/backup/
```

Если файл по новому адресу (куда перемещается старый файл) существует, mv его заменит без предупреждений.

Чтобы mv спрашивал перед заменой, добавляем флаг -i:

```
$ mv -i 1.txt 2.txt
$ mv -i dir1/* /mnt/1TB/old_files/
```

КОНЕЦ "Команда mv"

Команда exit

«exit» завершает текущий процесс «bash» (при этом может закрыться окно эмулятора терминала). Команде можно передать в качестве аргумента код завершения интерпретатора команд:

```
$ exit 0
```

Если использовать "exit" без аргументов, кодом завершения интерпретатора команд будет код завершения
последней выполненной команды, который доступен через переменную "$?":

```
$ exit
```

эквивалентно

```
$ exit $?
```

КОНЕЦ "Команда exit"

Команда pwd

«pwd» расшифровывается так: «Print Working Directory». Эта команда печатает на stdout полный путь к текущему рабочему каталогу. Пример:

```
$ pwd
```

Выводит:

```
/home/username
```

, если запущена из этого каталога

КОНЕЦ "Команда pwd"

Команда rm

```
$ rm путь1 путь2 ... путьN
```

удаляет указанные файлы. Чтобы удалять каталоги рекурсивно (т.е. со всем, что лежит внутри) без запроса на подтверждение, пишем:

```
$ rm -r -f путь1 ... путьN
```

(-r = рекурсивно, -f = без вопросов)

КОНЕЦ "Команда rm"

Команда rmdir

```
$ rmdir имя_каталога
```

удаляет указанный каталог, если он пустой. Например,

```
$ rmdir /home/user/dir1
```

Можно удалить каталог со всеми родительскими, если они останутся пустыми:

```
$ rmdir -p a/b/c
```

— это как если набрать

```
$ rmdir a/b/c a/b a
```

КОНЕЦ "Команда rmdir"

Команда alias

«alias» без аргументов выдаёт список настроенных псевдонимов:

```
$ alias
```

выводит

```
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo
terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-
9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

Чтобы добавить псевдоним, пишем:

```
$ alias "имя=команда аргумент1 аргумент2 ... аргументN"
```

Например,

```
$ alias "lslapt=ls -lapt"
```

КОНЕЦ "Команда alias"

Команд unalias

Чтобы отменить задание псевдонима, используем команду "unalias ИМЯ_ПСЕВДОНИМА". Например:

```
$ unalias lslapt
```
КОНЕЦ "Команд unalias"

Команда touch

Команда "touch" заменяет метки времени доступа и модификации на указанных файлах на текущее время (если файл не существует, создаётся пустой файл с соответствующим именем):
```
$ touch 1.txt 2.txt
```
КОНЕЦ "Команда touch"

Команда type

Чтобы узнать тип команды "команда", вводим "type команда". Например, для "ls":
```
$ type ls
```
Это выдаёт вывод:
```
ls — это псевдонимом для «ls --color=auto»
```
Или для "for":
```
$ type for
```
выдаёт:
```
for — это ключевое слово командного процессора
```
Ещё пример:
```
$ type man
```
выдаёт:
```
man является /usr/bin/man
```
Ещё:
```
$ type help
```
выдаёт:
```
help — это встроенная команда bash
```
КОНЕЦ "Команда type"

Команда which

«which» ищет исполняемый файл (не встроенную команду bash, не функцию bash, не псевдоним bash) в каталогах, заданных в переменной PATH:
```
$ which info
```
выдаёт:
```
/usr/bin/info
```
КОНЕЦ "Команда which"

Команда more

«more» работает аналогично «less». При запущенном «more»: q = выход, пробел = прокрутить дальше.
Примеры:
```
$ cat /var/log/dmesg | more
$ more /var/log/dmesg
```
КОНЕЦ "Команда more"
КОНЕЦ "Базовые команды"

Перенаправление ввода из файлов и вывода в файлы

Вывод stdout можно перенаправить в файл с именем "ИМЯ", если приписать "> ИМЯ" к команде. Например,

```
$ ls -l -apt > 2.txt
```

выведет содержимое текущего каталога (кроме скрытых файлов, т.е. файлов, имена которых начинаются с символа ".") в файл 2.txt. Старое содержимое файла при этом заменится выводом указанной команды.

Можно сделать так же, но с дописыванием вывода указанной команды в конец файла (после имеющихся в нём данных). Для этого используется последовательность символов ">>":

```
$ ls -lapt >> 2.txt
```

Если файл «2.txt» не существует, он будет создан в текущей папке (создан пустым и примет направленный в него поток байт). Аналогично со значком ">" (если файла нет, он сначала создаётся пустым; если файл есть, он сначала превращается в пустой).

Можно привязать stdin процесса к содержимому файла. Например,

```
$ less -i < 2.txt
```

По этой команде будет запущен процесс "less", который покажет содержимое файла "2.txt" в терминале (он получит это содержимое через stdin). Аналогичный результат даёт команда:

```
$ cat 2.txt | less -i
```

То же самое можно получить и так:

```
$ cat < 2.txt | less -i
```

И так:

```
$ less -i 2.txt
```

Для получения подробностей введите:

```
$ man less
```

и

```
$ man cat
```

КОНЕЦ «Перенаправление ввода из файлов и вывода в файлы»

Абсолютные и относительные пути

Абсолютный путь - это указание полного пути к файлу/каталогу, начиная с корня (корень - каталог с именем "/"). Например файл

```
/dev/sda
```

имеет имя "sda" и находится в каталоге с именем "dev", который находится в корневом каталоге ("/"). Пути, не начинающиеся с "/", являются относительными и неявно начинаются с текущего каталога (у каждого работающего процесса в каждый момент времени есть текущий каталог, который может измениться через время). Например, в команде

```
$ cat 2.txt
```

используется относительный путь к файлу. Это файл с именем "2.txt" в текущем каталоге. Содержимое этого же файла можно вывести командой:

```
$ cat ./2.txt
```

КОНЕЦ "Абсолютные и относительные пути"

Текстовые файлы и комментарии

В GNU/Linux некоторые настройки хранятся в текстовых файлах, которые можно редактировать (изменять) текстовым редактором, например, nano, mousepad, pluma, xed, gedit, kate, kwrite, mcedit, vi, vim, pico, geany. В некоторых файлах разрешаются комментарии,

начинающиеся с символа '#'. Случаях комментариев символ '#' и всё, что идёт после него до конца строки, считается комментарием и игнорируется обрабатывающими файл программами.
КОНЕЦ "Текстовые файлы и комментарии"

"Создание скриптов на bash и работа в интерактивном режиме bash"

"Про комментарии в командной строке bash и скриптах bash"

Если в строке файла-скрипта или в строке, введённой пользователем в интерактивно запущенном bash,
есть слово, начинающееся с символа '#', то оно и всё, что идёт после него до конца строки,
является комментарием и игнорируется bash'эм.
КОНЕЦ "Про комментарии в командной строке bash и скриптах bash"

Скрипт bash — текстовый файл, содержащий инструкции интерпретатора команд bash. В начале этого файла идёт комментарий "#!/bin/bash", задающий путь к интерпретатору bash. По этому комментарию система понимает, что файл нужно запускать через интерпретатор bash (запускается процесс из файла /bin/bash и первым аргументом ему передаётся путь к файлу запускаемого скрипта). Отдельные команды bash можно разделять точкой с запятой (";") или символом перевода строки (нажимать Enter, что даёт вставку символа '\n', имеющего код 0x0a (равно 10) (см. `man ascii`)).

Все команды, вводимые в интерактивный bash, записываются в файл ~/.bash_history. Их можно посмотреть, введя команду "history". Чтобы очистить историю, вводим "history -c".
Можно напрямую посмотреть ~/.bash_history:
```
$ less -i ~/.bash_history
```
или
```
$ more ~/.bash_history
```
Можно вывести историю через less:
```
$ history | less -i
```
Если в начале выполняемой команды стоит пробел, команда не записывается в историю:
```
$  history | less -i
```

Управляющие конструкции bash

В bash можно использовать стандартные для программирования управляющие конструкции.

Для удобства чтения скриптов (кодов) на bash можно выравнивать строки символами табуляции или пробела (ставить эти символы в начале строк, bash их проигнорирует)

"Ветвление (условное выполнение, if)"

По команде
```
$ help if
```
получаем описание конструкции if, в котором говорится:
```
if: if КОМАНДЫ; then КОМАНДЫ; [ elif КОМАНДЫ; then КОМАНДЫ; ]... [
else КОМАНДЫ; ] fi
    Выполнение команд в зависимости от условий.
```

```
    Сначала выполняется список «if КОМАНДЫ».  Если состояние
выхода нулевое,
    выполняется список «then КОМАНДЫ».  В противном случае
выполняется по очереди
    все списки «elif КОМАНДЫ», и если их состояние выхода будет
нулевым,
    выполнится список «then КОМАНДЫ», и команда if завершится.  В
противном случае
    выполнится список «else КОМАНДЫ», если он указан.  Состояние
выхода всей
    конструкции соответствует состоянию выхода последней
выполненной команды или будет нулевым,
    если ни одна проверка условия не возвратила истину.

    Состояние выхода:
    Возвращает состояние последней выполненной команды.
```

Ветвление (условное выполнение, if) - выполнение одной из частей кода в зависимости от условий.

Если написать команду вида "if КОМАНДЫ_1; then КОМАНДЫ_2; [elif КОМАНДЫ_3; then КОМАНДЫ_4;]... [else КОМАНДЫ_else;] fi", bash сначала выполнил список команд КОМАНДЫ_1. Если эта операция завершилось успешно (код завершения 0), bash выполнит КОМАНДЫ_2 и выйдет из if. Скобки [...] говорят о том, что то, что в скобках, может присутствовать или отсутствовать. После КОМАНДЫ_2 могут идти несколько блоков "elif КОМАНДЫ_elif_условие ; then КОМАНДЫ_elif_выоплнить ; ", после которых может идти блок "else КОМАНДЫ_else ;". В конце идёт ключевое слово "fi".

Если не выполнено первое условие (КОМАНДЫ_1 завершились с кодом не 0), bash выполняет по очереди все блоки "КОМАНДЫ_elif_условие", пока не найдёт условие выполненное. Если найдёт, он выполняет соответствующий блок "КОМАНДЫ_elif_выоплнить" и выходит из "if". Если не найдёт, он выполняет (при наличии) "КОМАНДЫ_else" и выходит из "if".

Пример if

Вот пример (немного модифицированный) применённого когда-то скрипта с ветвлением (некоторые значения переменных изменены):

```
#!/bin/bash

myvar_mountpoint="/mnt/cryptsetup/hostname/cryptsetup-00"
myvar_partition="/mnt/storage/cs/cs-00.bin"
myvar_device_name="cs-00"

mkdir -p "${myvar_mountpoint}"

if mountpoint "${myvar_mountpoint}" ; then
        echo "Уже подключено. Нажмите Ctrl+D для выхода из этого скрипта"
        cat
        exit 0
else
        mkdir -p "${myvar_mountpoint}"
        echo "Создаём устройство. После появления сообщения о вводе парольной
фразы или пароля (с двоеточием в конце) введите пароль ДВА РАЗА (пароль, Enter,
пароль, Enter). При вводе пароля пароль отображаться не будет (и звёздочек тоже
не должно быть). Щёлкать мышкой перед вводом пароля никуда не надо"
        if cryptsetup -y open --type plain "${myvar_partition}" "$
{myvar_device_name}"
; then
```

```
                echo "Устройство создано. Проверяем файловую систему на
устройстве"
                fsck.ext4 -f -y "/dev/mapper/${myvar_device_name}"
                echo -e "\\n\\nЕщё раз проверяем файловую систему на устройстве"
                if fsck.ext4 -f -y "/dev/mapper/${myvar_device_name}" ; then
                        echo "Проверено. Подключаем"
                        mkdir -p "${myvar_mountpoint}"
                        if mount -t ext4 "/dev/mapper/${myvar_device_name}" "$
{myvar_mountpoint}" ; then
                                echo "Подключено. Точка монтирования: \"$
{myvar_mountpoint}\". Нажмите Ctrl+D для выхода из этого скрипта"
                                cat
                                exit 0
                        else
                                echo "Не удалось подключить. Удаляем устройство"
                                if cryptsetup close "${myvar_device_name}" ;
then
                                        echo "Удалено. Нажмите Ctrl+D для
выхода"
                                        cat
                                        exit 2
                                else
                                        echo "Не удалось удалить. Нажмите Ctrl+D
для выхода"
                                        cat
                                        exit 3
                                fi
                        fi
                else
                        echo "Не удалось проверить файловую систему на
устройстве. Удаляем устройство"
                        if cryptsetup close "${myvar_device_name}" ; then
                                echo "Удалено. Нажмите Ctrl+D для выхода"
                                cat
                                exit 4
                        else
                                echo "Не удалось удалить. Нажмите Ctrl+D для
выхода"
                                cat
                                exit 5
                        fi
                fi
        else
                echo "Не удалось создать устройство. Нажмите Ctrl+D для выхода"
                cat
                exit 1
        fi
fi
```

КОНЕЦ «Пример if»

Другой пример: if, case, for in, export, return, комментарии

Вот файл «/etc/profile.d/95custom.sh»:

```
#!/bin/bash

for i in /opt/custom/* ; do
 case "$i" in
 "/opt/custom/*" )
  echo -en
  ;;
 *)
```

```
  if test -d "${i}/include" ; then
   if test "x$CPPFLAGS" = "x" ; then
    CPPFLAGS="-I${i}/include"
   else
    CPPFLAGS="-I${i}/include ${CPPFLAGS}"
   fi
  fi
  if test -d "${i}/lib/x86_64-linux-gnu" ; then
   if test "x$LDFLAGS" = "x" ; then
    LDFLAGS="-L${i}/lib/x86_64-linux-gnu"
   else
    LDFLAGS="-L${i}/lib/x86_64-linux-gnu ${LDFLAGS}"
   fi
  fi
  if test -d "${i}/lib" ; then
   if test "x$LDFLAGS" = "x" ; then
    LDFLAGS="-L${i}/lib"
   else
    LDFLAGS="-L${i}/lib ${LDFLAGS}"
   fi
  fi
  if test -d "${i}/lib/x86_64-linux-gnu/pkgconfig" ; then
   if test "x$PKG_CONFIG_PATH" = "x" ; then
    PKG_CONFIG_PATH="${i}/lib/x86_64-linux-gnu/pkgconfig"
   else
    PKG_CONFIG_PATH="${i}/lib/x86_64-linux-gnu/pkgconfig:$
{PKG_CONFIG_PATH}"
   fi
  fi
  if test -d "${i}/lib/pkgconfig" ; then
   if test "x$PKG_CONFIG_PATH" = "x" ; then
    PKG_CONFIG_PATH="${i}/lib/pkgconfig"
   else
    PKG_CONFIG_PATH="${i}/lib/pkgconfig:${PKG_CONFIG_PATH}"
   fi
  fi
  if test -d "${i}/lib/x86_64-linux-gnu" ; then
   if test "x${LD_LIBRARY_PATH}" = "x" ; then
    LD_LIBRARY_PATH="${i}/lib/x86_64-linux-gnu"
   else
    LD_LIBRARY_PATH="${i}/lib/x86_64-linux-gnu:${LD_LIBRARY_PATH}"
   fi
  fi
  if test -d "${i}/lib" ; then
   if test "x${LD_LIBRARY_PATH}" = "x" ; then
    LD_LIBRARY_PATH="${i}/lib"
   else
    LD_LIBRARY_PATH="${i}/lib:${LD_LIBRARY_PATH}"
   fi
  fi
  if test -d "${i}/bin" ; then
   if test "x$PATH" = "x" ; then
        PATH="${i}/bin"
```

```
   else
        PATH="${i}/bin:${PATH}"
   fi
  fi
  if test -d "${i}/sbin" ; then
   if test "x$PATH" = "x" ; then
        PATH="${i}/sbin"
   else
        PATH="${i}/sbin:${PATH}"
   fi
  fi
  ;;
 esac
done

export CPPFLAGS LDFLAGS PKG_CONFIG_PATH

export LD_LIBRARY_PATH PATH

return 0

#LD_LIBRARY_PATH="/opt/custom/lib/x86_64-linux-gnu:/opt/custom/
lib"

#CPPFLAGS="-I/opt/custom/include"
#LDFLAGS="-L/opt/custom/lib/x86_64-linux-gnu -L/opt/custom/lib"
#PKG_CONFIG_PATH="/opt/custom/lib/x86_64-linux-gnu/pkgconfig:/
opt/custom/lib/pkgconfig"

#export CPPFLAGS LDFLAGS PKG_CONFIG_PATH


#LD_LIBRARY_PATH="/opt/custom/lib/x86_64-linux-gnu:/opt/custom/
lib"
#if test "x$PATH" = "x" ; then
#    PATH="/opt/custom/bin:/opt/custom/sbin"
#else
#    PATH="/opt/custom/bin:/opt/custom/sbin:$PATH"
#fi
#export LD_LIBRARY_PATH PATH



#CPPFLAGS="-I/opt/gobject-introspection/include -I/opt/atk/include
-I/opt/gdk-pixbuf/include -I/opt/glib-2.72.0/include
-I/opt/libepoxy-1.5.9/include -I/opt/libsigc++-3.0.7/include
-I/opt/pango-1.50.6/include -I/opt/gtk-4.6.2/include -I/opt/cairo-
git/include -I/opt/cairomm-git/include -I/opt/pixman-git/include -
I/opt/glibmm-git/include -I/opt/pangomm-git/include
-I/opt/graphene-git/include -I/opt/gtkmm-4.6.1/include"
#LDFLAGS="-L/opt/gobject-introspection/lib/x86_64-linux-gnu
-L/opt/atk/lib/x86_64-linux-gnu -L/opt/gdk-pixbuf/lib/x86_64-
linux-gnu -L/opt/glib-2.72.0/lib/x86_64-linux-gnu -L/opt/libepoxy-
```

```
1.5.9/lib/x86_64-linux-gnu -L/opt/libsigc++-3.0.7/lib/x86_64-
linux-gnu -L/opt/pango-1.50.6/lib/x86_64-linux-gnu -L/opt/gtk-
4.6.2/lib/x86_64-linux-gnu -L/opt/cairo-git/lib/x86_64-linux-gnu -
L/opt/cairomm-git/lib/x86_64-linux-gnu
-L/opt/pixman-git/lib/x86_64-linux-gnu -L/opt/glibmm-git/lib
-L/opt/pangomm-git/lib/x86_64-linux-gnu
-L/opt/graphene-git/lib/x86_64-linux-gnu
-L/opt/gtkmm-4.6.1/lib/x86_64-linux-gnu"
#PKG_CONFIG_PATH="/opt/gobject-introspection/lib/x86_64-linux-
gnu/pkgconfig:/opt/atk/lib/x86_64-linux-gnu/pkgconfig:/opt/gdk-
pixbuf/lib/x86_64-linux-gnu/pkgconfig:/opt/glib-2.72.0/lib/x86_64-
linux-gnu/pkgconfig:/opt/libepoxy-1.5.9/lib/x86_64-linux-gnu/
pkgconfig:/opt/libsigc++-3.0.7/lib/x86_64-linux-gnu/pkgconfig:/
opt/pango-1.50.6/lib/x86_64-linux-gnu/pkgconfig:/opt/gtk-4.6.2/
lib/x86_64-linux-gnu/pkgconfig:/opt/cairo-git/lib/x86_64-linux-
gnu/pkgconfig:/opt/cairomm-git/lib/x86_64-linux-gnu/pkgconfig:/
opt/pixman-git/lib/x86_64-linux-gnu/pkgconfig:/opt/glibmm-git/
lib/pkgconfig:/opt/pangomm-git/lib/x86_64-linux-gnu/pkgconfig:/
opt/graphene-git/lib/x86_64-linux-gnu/pkgconfig:/opt/gtkmm-4.6.1/
lib/x86_64-linux-gnu/pkgconfig"



#export CPPFLAGS LDFLAGS PKG_CONFIG_PATH

#LD_LIBRARY_PATH="/opt/gobject-introspection/lib/x86_64-linux-
gnu:/opt/atk/lib/x86_64-linux-gnu:/opt/gdk-pixbuf/lib/x86_64-
linux-gnu:/opt/glib-2.72.0/lib/x86_64-linux-gnu:/opt/libepoxy-
1.5.9/lib/x86_64-linux-gnu:/opt/libsigc++-3.0.7/lib/x86_64-linux-
gnu:/opt/pango-1.50.6/lib/x86_64-linux-gnu:/opt/gtk-4.6.2/lib/
x86_64-linux-gnu:/opt/cairo-git/lib/x86_64-linux-gnu:/opt/cairomm-
git/lib/x86_64-linux-gnu:/opt/pixman-git/lib/x86_64-linux-gnu:/
opt/glibmm-git/lib:/opt/pangomm-git/lib/x86_64-linux-gnu:/opt/
graphene-git/lib/x86_64-linux-gnu:/opt/gtkmm-4.6.1/lib/x86_64-
linux-gnu"
#if test "x$PATH" = "x" ; then
#    PATH="/opt/gobject-introspection/bin:/opt/atk/bin:/opt/gdk-
pixbuf/bin:/opt/glib-2.72.0/bin:/opt/libepoxy-1.5.9/bin:/opt/
libsigc++-3.0.7/bin:/opt/pango-1.50.6/bin:/opt/gtk-4.6.2/bin:/
opt/cairo-git/bin:/opt/cairomm-git/bin:/opt/pixman-git/bin"
#else
#    PATH="/opt/gobject-introspection/bin:/opt/atk/bin:/opt/gdk-
pixbuf/bin:/opt/glib-2.72.0/bin:/opt/libepoxy-1.5.9/bin:/opt/
libsigc++-3.0.7/bin:/opt/pango-1.50.6/bin:/opt/gtk-4.6.2/bin:/
opt/cairo-git/bin:/opt/cairomm-git/bin:/opt/pixman-git/bin:$PATH"
#fi
#export LD_LIBRARY_PATH PATH
```

КОНЕЦ «Другой пример if»
КОНЕЦ "Ветвление (условное выполнение, if)"

В bash два варианта цикла for

Цикл for как в C++

Этот вариант работает как аналогичный цикл for в языке программирования C++
Вот что говорит "man bash" на эту тему:

```
for (( expr1 ; expr2 ; expr3 )) ; do list ; done
       First, the arithmetic expression expr1 is evaluated
according to the rules described below under ARITHMETIC
EVALUATION.   The  arithmetic  expression
       expr2 is then evaluated repeatedly until it evaluates to
zero.  Each time expr2 evaluates to a non-zero value, list is
executed and the arithmetic ex-
       pression expr3 is evaluated.  If any expression is
omitted, it behaves as if it evaluates to 1.  The return value is
the exit status of the last  com-
       mand in list that is executed, or false if any of the
expressions is invalid.
```

Тело цикла идёт между "do" и "done". Сначала выполняется expr1, потом expr2. Если рещультат выполнения expr2 ложный (false), цикл завершается. Если истинный, выполняется тело цикла и потом expr3 и возвращаемся к проверке expr2. На это этапе снова проверяется expr2 и в зависимости от результата, поведение такое же. Цикл закончится, как только первый раз результат выполнения expr2 будет ложным.
Пример:

```
$ for ((i=0;i<4;++i)) ; do touch ${i}.txt; done
```

Этот цикл создаст файлы 0.txt, 1.txt, 2.txt, 3.txt.
КОНЕЦ "Цикл for как в C++"

Цикл for in

Вот что говорится в "man bash":

```
for name [ [ in [ word ... ] ] ; ] do list ; done
              The  list  of  words following in is expanded,
generating a list of items.  The variable name is set to each
element of this list in turn, and list is
              executed each time.  If the in word is omitted, the
for command executes list once for each positional parameter that
is set (see  PARAMETERS  below).
              The  return status is the exit status of the last
command that executes.  If the expansion of the items following in
results in an empty list, no com-
              mands are executed, and the return status is 0.
```

Пример:

```
$ for i in *.txt ; do rm -fv $i ; done
```

*.txt раскроется в список всех файлов, чьи имена заканчиваются на ".txt" (кроме скрытых файлов (это файлы, имена которых начинаются на точку ('.'))). Переменная $i Будет пробегать все полученные имена файлов и для каждого значения $i будет выполнено по одноу разу тело цикла (тело — между "do" и "done"). Этот цикл удалит без предупреждения все файлы с расширением ".txt", кроме скрытых.
КОНЕЦ "Цикл for in"

Инструкция return

```
$ return
```

говорит bash закончить выполнение текущей функции (вызванной по имени функции) или скрипта (вызванной командой «source» или «.»).
Текст из `$ help return`:

```
return: return [n]
    Возврат из функции командного процессора.

    Выполняет выход из функции или исходного скрипта со значением
возврата,
    указанным как N. Если N не указан, используется состояние
возврата
    последней команды, выполненной в функции или скрипте.

    Состояние выхода:
    Возвращает N или ошибку, если командный процессор не выполняет
функцию или скрипт.
```

КОНЕЦ "Инструкция return"

"Циклы while, until"

Цикл while работает аналогично чиклу while C++ (который является упрощённым вариантом циалв for C++). Отличие цикла while от упрощённого варианта bash-цикла for (варианта C++-style) - использование списка команд, а не выражения expt2. Вот что говорится в "man bash":

```
  while list-1; do list-2; done
       until list-1; do list-2; done
              The while command continuously executes the list
list-2 as  long
              as the last command in the list list-1 returns an
exit status of
              zero.  The until command is identical to the while
command,  ex-
              cept that the test is negated: list-2 is executed as
long as the
              last command in list-1 returns a non-zero exit
status.  The exit
              status of the while and until commands is the exit
status of the
              last command executed in list-2, or zero if none was
executed
```

Выполняется list-1 (здесь это список команд, а в for C++-style - используется выражение expr2, не список команд).

Если результат 0 (успех), выполняется list2 и обратно к выполнению list-1. Когда первый раз будет неуспех в list-1, цикл while завершится.

until работает аналогично, но цикл завершается при первом успехе.

КОНЕЦ "Циклы while, until"

"Ключевое слово break"

Из циклов можно выходить досрочно, если выплнить инструкцию "break". Вот что говорится в "man bash":

```
  break [n]
              Exit from within a for, while, until, or select
loop.  If  n  is
              specified, break n levels.  n must be ≥ 1.  If n is
greater than
```

```
              the number of enclosing loops, all enclosing loops
are  exited.
              The  return  value is 0 unless n is not greater than
or equal to
              1
```

КОНЕЦ "Ключевое слово break"

"Ключевое слово continue"

Вот что говорится в "man bash":

```
  continue [n]
              Resume the next iteration of the enclosing for,
while, until, or
              select  loop.   If  n  is specified, resume at the
nth enclosing
              loop.  n must be ≥ 1.  If n is greater than the
number  of  en-
              closing  loops, the last enclosing loop (the ``top-
level'' loop)
              is resumed.  The return value is 0 unless n is not
greater  than
              or equal to 1
```

continue делает переход к следующей итерации цикла (в while и until это переводит к проверке условия (list-1), в for C++-style - к выполнению expr3, после чего идёт проверка условия (expr2), в for in $i принимает следующее значение и начинается следующая итерация (а если нет следующего значения, цикл завершается))

КОНЕЦ "Ключевое слово continue"

Определение функции

Вот что говорит `$ help function`:

```
function: function ИМЯ { КОМАНДЫ ; } или ИМЯ () { КОМАНДЫ ; }

    Определение функции командного процессора.

    Создаёт функцию командного процессора с указанным именем.  При
запуске в качестве простой команды

    ИМЯ выполняет КОМАНДЫ в контексте вызывающего их командного
процессора.  При вызове ИМЕНИ

    аргументы передаются в функцию как $1...$n, а функция получает

    название $FUNCNAME.

    Состояние выхода:

    Возвращает успех, если переменная ИМЯ доступно для записи.
```

КОНЕЦ «Определение функции»

"Инструкция exec"

Команда exec позволяет заменить текущий процесс bash процессом выполнения заданной команды

Вот что говорится в "man bash":

```
  exec [-cl] [-a name] [command [arguments]]
              If command is specified, it replaces the shell.  No new  process
              is  created.  The arguments become the arguments to command.  If
              the -l option is supplied, the shell places a dash at the begin-
              ning of the zeroth argument passed to command.  This is what lo-
              gin(1) does.  The -c option causes command to be executed  with
              an  empty environment.  If -a is supplied, the shell passes name
              as the zeroth argument to the executed command.  If command can-
              not  be executed for some reason, a non-interactive shell exits,
              unless the execfail shell option is enabled.  In that  case,  it
              returns  failure.   An  interactive shell returns failure if the
              file cannot be executed.  A subshell  exits  unconditionally  if
              exec  fails.  If command is not specified, any redirections take
              effect in the current shell, and the return  status  is  0.   If
              there is a redirection error, the return status is 1
```

КОНЕЦ "Инструкция exec"

"Ключевое слово case"

case позволяет выполнять разные блоки кода в зависимости от соответствия строки разным шаблонам (не путать с регулярными выражениями). Вот что говорится в "man bash":

```
  case word in [ [(] pattern [ | pattern ] ... ) list ;; ] ... esac
              A case command first expands word, and tries to match it against each pattern in turn, using the matching rules described under Pattern  Matching  be-
              low.   The  word is expanded using tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process substitution
              and quote removal.  Each pattern examined is expanded using tilde expansion, parameter and variable expansion, arithmetic expansion, command substitu-
              tion,  and  process substitution.  If the nocasematch shell option is enabled, the match is performed without regard to the case of alphabetic charac-
```

```
              ters.  When a match is found, the corresponding list
is executed.  If the ;; operator is used, no subsequent matches
are  attempted  after  the  first
              pattern match.  Using ;& in place of ;; causes
execution to continue with the list associated with the next set
of patterns.  Using ;;& in place of ;;
              causes the shell to test the next pattern list in
the statement, if any, and execute any associated list on a
successful match.  The  exit  status  is
              zero if no pattern matches.  Otherwise, it is the
exit status of the last command executed in list
```

Пример:

```
for i in * ; do
 case $i in
  backup )
   echo "Ignoring $i" ;;
  *) rsync -aH --progress "$i" /mnt/backup/ ; sync ;;
 esac
done
```

Здесь переменная $i будет пробегать все файлы (в текущей папке), кроме скрытых, и в случае $i = "backup" копирование выполняться

не будет, а в других случаях будет выполняться копирование в папку /mnt/backup

КОНЕЦ "Ключевое слово case"

"Ключевое слово test"

в конструкциях if, while, until можно использовать команду test для проверки условий. Смотрите "help test | less -i" для

подробного описания. Пример:

```
$ for i in * ; do
    if test -d "$i" ; then
     echo "$i is a directory"
    else
     echo "$i is not a directory"
    fi
   done
```

КОНЕЦ "Ключевое слово test"

КОНЕЦ "Управляющие конструкции bash"

"Экранирование символов в bash"

Для включения в команду специальных символов нужно ставить перед ними обратный слэш ('\', backslash). К специальным символам относятся: '$', '\', '#', '*', '?', '[', ']', '{', '}', '(', ')', ' ' (пробел), "'" (одинарная кавычка), '"' (двойная кавычка), '=', '!', '<', '>', '|', ';', '~' (тильда), '&' (амперсанд). Например, для удаления файла "Фото 1 [море].jpg" можно написать:

```
$ rm -fv Фото\ 1\ \[море\].jpg
```

Чтобы вывести строку "*\", пишем:

```
$ echo \\\*\\
```

("\\" превращается в "\", "*" в "*", "\\" в "\")

Если строка взята в кавычки (двойные: "строка"), то кавычки не являются частью строки, а символы '*', '?', ' ' (пробел) в ней можно не экранировать. Кроме того, строку можно брать в кавычки одинарные ('строка'), тогда '$' тоже перестаёт быть специальным символом

Например,
$ echo "$BASHPID"
может вывести
29298
, а
$ echo '$BASHPID'
выведет
$BASHPID
(в одинарных кавычках символ '$' (знак доллара) не имеет специального значения)
КОНЕЦ "Экранирование символов в bash"

"here-документы"

можно задать содержимое потока stdin команды так:
$ команда <<СЛОВО
ТЕКСТ
ТЕКСТ
ТЕКСТ
СЛОВО
"СЛОВО" в конце должно совпадать с "СЛОВО" в начале. И перед ним не должно быть символов в начале строки.
Тогда на ввод команде попадёт весь текст от начала до "СЛОВО" (не включая "СЛОВО").
Пример:
$ cat > 1.txt <<EOF
строка 1
строка 2
EOF
Этот код запишет строки 1 и 2 в файл 1.txt (вывод команды cat перенаправлен в 1.txt)
КОНЕЦ "here-документы"

"Многострочые команды"

В скрипте или интерактивном bash символом перевода строки (клавиша Enter) завершается текущая команда
(другой способ разделения команд - точка с запятой (';')). Можно поставить в конце строки (перед символом
перевода строки: '\n') обратный слэш ('\'), тогда символ перевода строки проигнорируется, и можно будет
продолжать команду на следующей строке. Можно использовать более 2-х строк. Пример

```
#!/bin/bash
echo  слово1  слово 2 \
 слово 3 \
 слово4
```

КОНЕЦ "Многострочые команды"

"Переменные bash"

Переменная bash $? содержит код завершения последней выполненной команды
(если выполнялся конвейер (использовался pipe) вида "cmd_1 | cmd_2 | ... | cmd_n",
то $? будет содержать результат выполнения cmd_n)

$ exit
эквивалентно
$ exit $?

Переменая $BASHPID содержит идентификатор процесса (PID, Process ID) интерпретатора bash.
Переменная $BASH_SUBSHELL содержит 1, если команда выполняется в subshell и 0, если нет.
Примеры:
$ (echo $BASH_SUBSHELL)
выведет
1
, но
$ echo $BASH_SUBSHELL
выведет
0
(subshell выполняет код в скобках)

Переменные $1, $2, $3, ... содержат аргументы переданные скрипту при его запуске.
$0 - имя скрипта. "$@" - эта строка преобразуется во все аргументы по отдельности.
"$*" - эта строка преобразуется во все аргументы одной строкой (склеенные).
(при запуске bash интерактивно тоже можно задавать значения переменных $1, $2, ...)

Переменная $# содержит количество определённых переменных в ряду $1, $2, $3, ... (без $0)
Вот скрипт, который выводит $#:
```
#!/bin/bash
echo "\$#: $#"
```
(конец скрипта)

Например, если запустить скрипт 1.bash так:
$ ./1.bash str1
, то переменная $1 будет содержать строку "str1"

Командой shift в скрипте или в интерактивном bash можно сместить значения переменных $1, $2, ... влево. Например, если были заданы $1="q", $2="a", $3="z", $4 не определено, то после
команды "shift" $3 будет не определено, $2 будет ="z", $1 будет равно "a".
Значение переменой $0 не изменится.
Вот что говорится в "man bash":
```
shift [n]
   The positional parameters from n+1 ... are renamed  to  $1  ....
   Parameters  represented by the numbers $# down to $#-n+1 are un-
   set.  n must be a non-negative number less than or equal to  $#.
   If  n is 0, no parameters are changed.  If n is not given, it is
   assumed to be 1.  If n is greater than $#, the positional param-
   eters  are  not changed.  The return status is greater than zero
   if n is greater than $# or less than zero; otherwise 0
```
(можно после shift указать число n)
КОНЕЦ "Переменные bash"

Шаблоны bash

В аргументах команд можно использовать шаблоны (это globbing с использованием wildcards; не путать с регулярными выражениями: regular expressions (см. `$ man 7 regex`)):

1. * (звёздочка, asterisk)

2. ? (вопросительный знак, question mark)
3. [] (блоки с квадратными скобками (square brackets))

- `*` соответствует любой (в том числе, пустой) последовательности символов.
- `?` соответствует любому одному символу.
- блок с квадратными скобками соответствует одному из символов из перечисленных в квадратных скобках символов и диапазонов (пример: [a-c,f] соответствует любому из символов a, b, c, f). Если первым в квадратных скобках идёт символ '^', блок соответствует любому символу, кроме перечисленных.

Например,

```
$ ls -d photo-*.png
```

Выведет информацию обо всех объектах файловой системы в текущей папке, чьи имена начинаются на "photo-" и заканчиваются на ".png".

Примеры шаблонов bash

Пример: путь к файлу (или другому объекту файловой системы) соответствует шаблону (wildcard)

```
*photo*.png
```

, если существуют такие строки str1 и str2, что этот путь является результатом объединения строк str1, "photo", str2 и ".png" в указанном порядке.

Пример: путь соответствует шаблону

```
photo.p[pg]m
```

, если существует такая строка str1, состоящая из одного символа 'p' или 'g', что путь представим как объединение (сцепление, конкатенация) строк "photo.p", str1 и "m" в указанном порядке. Этому шаблону соответствуют два пути: "photo.ppm" и "photo.pgm".

Пример: путь соответствует шаблону

```
photo-??.p[pg]m
```

, если существуют такие строки str1, str2, str3, что str1 и str2 являются строками из одного символа, str3 - строка из одного символа \'p\' или \'g\' и указанный путь представим как результат склеивания (конкатенации) строк "photo-", str1, str2, ".p", str3, "m" в указанном порядке.

КОНЕЦ «Примеры шаблонов bash»

КОНЕЦ «Шаблоны bash»
КОНЕЦ "Создание скриптов на bash"

Разделы жёсткого диска

Содержимое жёсткого диска - конечная последовательность байт. Обычно в начале жёсткого диска создаётся таблица разделов (обычно это таблица разделов MBR (она же - msdos) или GPT (GUID Partition Table)). При использовании таблицы разделов GPT первый сектор диска (первые 512 байт) будут заняты защитной (protective) таблицей разделов MBR, в которой всё место, кроме таблицы разделов MBR (первого сектора), выделено под раздел типа "0xEE". При таком подходе программы, не распознающие таблицу разделов GPT, будут считать, что место выделено под что-то, и не будут изменять данные на диске. Для таблицы разделов GPT копии её главного заголовка (второй сектор) и данных о разделах (сектора с номерами 3-34) хранятся в конце диска. При использовании таблицы разделов на диске выделяются непрерывные непересекающиеся блоки, называемые разделами. Например, на жёстком диске размером 1024 GiB можно создать таблицу разделов GPT, один раздел размером около 32 GiB и один раздел размером около 992 GiB (вся таблица разделов GPT (включая копии в конце диска) должна занимать 33.5 KiB).

КОНЕЦ «Разделы жёсткого диска»

Безопасность

Базовые права на объекты файловой системы (раздел не доработан; см. английский текст по этой теме в этом файле)

Базовые права доступа на файлы и каталоги (папки) описываются последовательностью из 12 бит, разделённой на 4 группы по 3 бита:

Биты базовых прав доступа

	Дополнительно			**user (владелец)**		
	Set UID (утсановить User Id)	**Set GID (установить Group ID)**	**Sticky bit**	**Read (чтение)**	**Write (запись)**	**Execute (исполнение)**
Номер бита	1	2	3	4	5	6

	group (группа-владелец)			**others (остальные)**		
	Read (чтение)	**Write (запись)**	**Execute (исполнение)**	**Read (чтение)**	**Write (запись)**	**Execute (исполнение)**
Номер бита	7	8	9	10	11	12

У каждого объекта файловой системы есть владелец (он же - пользователь-владелец, привязанный по числовому идентификатору UID (User Identificator)) и группа-владелец, привязанная по числовому идентификатору группы (GID, Group ID).

Биты 4-6 определяют права на доступ к объекту файловой системы от имени пользователя-владельца файлы. Биты 7-9 - от имени всех пользователей-членов группы.

КОНЕЦ «Базовые права на объекты файловой системы»

КОНЕЦ «Безопасность»

Краткий сбор команд, комбинаций клавиш и другого

Большой список команд командной строки, без аргументов

gnome-disks pmount cd ls mkdir rmdir rm cp mv echo type declare man help info touch cat less htop vlock pwgen apt-get apt apt-cache calc brasero xorrisofs xfburn gimp vlc top ps free lsblk blkid id w who du df find mount umount mkfs.x e2label tune2fs swaplabel ntfslabel dd xz bzip2 gzip gunzip bunzip2 7z rsync ssh scp aqemu xscreensaver-demo xfce4-appearance-settings xfwm4-settings xfwm4-tweaks-settings dm-tool set env screen mc mcedit mcview vim vi mknod file lvm2 mkisofs gparted parted iptables telnet cryptsetup partprobe nano aptitude useradd groupadd userdel groupdel adduser addgroup usermod sync time date update-grub grub-install update-initramfs which locate poweroff halt reboot swapon swapoff read test ln dir sed awk python c++ gdb make modprobe depmod rmmod lsattr chattr gpasswd galculator chgrp chmod chown newgrp cut x11vnc remmina xsane nmap traceroute tracepath mtr ping host gucharmap dig pix geeqie simple-scan unzip zip tar lsof diff patch hexdump bash su sudo lshw lsusb lsmod lspci hwinfo wget curl timedatectl eject source . : uname ulimit uptime wc grep lins lynx glxinfo alsamixer sudoedit whoami basename dirname alias unalias more dpkg dpkg-reconfigure rpm onboard journalctl systemctl tr mintupdate-cli inxi pastebin synaptic mintinstall xflock4 tee shred joe mugshot xfce4-settings-manager menulibre xfce4-appfinder login kill killall xkill fdisk truncate head tail stty base64 keyctl shift pushd popd pkexec exec valgrind guvcview whatis whereis ldd lshal? ldconfig cal openssl openvpn tmux lsb_release hostname last dmesg lsdev watch dmidecode iotop mousepad shutdown chkconfig service init update-rc.d groupmod pmap sar blockdev fuser renice nice ionice jobs fg bg nohup pkill umask hdparm smartctl smbclient cdrecord losetup tcptraceroute ethtool tty netstat netcat nc ip ifconfig iwconfig iwlist routel route arp socklist nslookup tcpdump tc domainname gdisk findmnt xinit ypdomainname nisdomainname dnsdomainname pv insmod mkfifo groups users apropos l ll lp lpr lpq lpadmin simplescreenrecorder recordmydesktop kazam pidgin phototonic peek pavucontrol nmapsi4 minicom linuxvnc ledit laptop-detect kate kwrite kamera kpartx kdenlive cvt rename xfce4-mouse-settings mkfs.ext4 xfce4-keyboard-settings xfce4-power-manager-settings neofetch cmp gcc g++ java wall talk mesg write emacs mysqldump mysql mkpasswd whois paste rev factor xaos seq script yes aview asciiview nl shuf ss tree pstree stat column at look tac strace disown getconf tput reset convert dstat bind pdftk startx updatedb ac sa lastcomm tasksel dselect dpkg-deb dpkg-split nethogs pidof vmstat iostat monit nethogs iftop monitorix arpwatch vnstat nagios4 nmon collectl glances sarg pidstat icinga2 wireshark shar ar cpio clonezilla partimage fsarchiver declone lftp hw-probe daemonize dialog meson ninja select apachetop ftptop mytop ntopng clear let eval bc command finger iconv socket arch clock logout pwck grck uniq readlink badblocks testdisk geany indent xev lslogins cpuid rfkill apt-file VboxManage printenv local compgen comm grep egrep fgrep rgrep lscpu lsmem lslogins lslocks lsof lspci lspcmcia lsusb lsdiff lsattr lsb_release imvirt time-admin sfdisk sgdisk cfsidk cgdisk partx flock hwclock ssh-copy-id startxfce4 mcopy pico xterm uudecode uuencode unarj conntrack xhost udevadm setxkbmap split ecryptfs-wrap-passphrase ecryptfs-insert-wrapped-passphrase-into-keyring fallocate a2ensite a2dissite a2enmod watch what-provides autokey-gtk netwatch lxc-create lxc-ls lxc-start lxc-info lxc-execute lxc-monitor lxc-wait lxc-stop lxc-console ddrescue nm-applet nm-connection-editor nmcli ssh-keygen xrandr gtf cvt wipefs stat ssh-add nproc xargs

Команды для развлечения

sl fortune cowsay xcowsay cowthink xcowthink toilet cmatrix oneko espeak aafire bb rig xeyes

Сторонние команды для развлечения

asciiquarium

Команды с аргументами (часть информации взята из интернета и не проверена, в т.ч., там, где об этом явно не говорится)

узнать число доступных процессоров (see number of available processors):

$ nproc

добавить ключ в SSH-агент (чтобы пароль много раз не вводить):

$ ssh-add ~/.ssh/id_ed25519

инфа о файле/папке/…

$ stat /

$ stat .

$ stat ~/.bashrc

...

удалить ФС — можно через wipefs

$ gtf 1280 720 75

$ cvt 1280 720 # haven't tested

$ xrandr --newmode "1280x1024_75.00" 138.54 1280 1368 1504 1728 1024 1025 1028 1069 -HSync +Vsync # gtf's output in this command line

$ xrandr --addmode VGA-1 "1280x1024_75.00"

$ xrandr --output VGA-1 --mode "1280x1024_75.00"

$ ssh-keygen -t ed25519

$ nmcli connection modify <connection-name> ipv4.route-metric X

truncate -r /dev/sdf /mnt/backup/sdf.img

dd if=/dev/sdf of=/mnt/backup/sdf.img iflag=fullblock conv=sparse,notrunc,noerror,sync status=progress bs=512

or

ddrescue --sparse /dev/sdf /mnt/backup/sdf.img /root/sdf-backup.map

if /mnt/backup/sdf.img already has zeroes and is not sparse:

fallocate -d /mnt/backup/sdf.img

ls -s --block-size 1 /mnt/backup/sdf.img

du -sm /mnt/backup/sdf.img

du -sm --apparent-size /mnt/backup/sdf.img

lsblk -o +fstype,uuid,rota,MIN-IO,PHY-SEC,LOG-SEC

$ qemu-system-x86_64 -monitor stdio -machine accel=kvm -m 2048 -boot menu=on -rtc base=utc -name "sparky-dokuwiki" -nodefaults -vga virtio -chardev vc,id=vc0 -mon chardev=vc0 -usb -device usb-tablet -device usb-ehci,id=ehci0 -drive file=cdrom.iso,if=none,media=cdrom,readonly=on,format=raw,id=cdrom0 -drive file=Snapshots/current_state.qcow2,if=none,media=disk,format=qcow2,id=currhdd -drive file=zvC9-swap.qcow2,if=none,media=disk,format=qcow2,id=swap0 -device ide-cd,drive=cdrom0,bootindex=1,unit=1 -device ide-hd,drive=currhdd,bootindex=2,unit=0 -device virtio-blk-pci,drive=swap0,bootindex=3 -netdev user,id=usernet0,ipv6=off -device e1000,netdev=usernet0,bootindex=4 -bios /usr/share/qemu/OVMF.fd

$ qemu-system-x86_64 -monitor stdio -machine accel=tcg -m 2048 -boot menu=on -rtc base=utc -name "sparky-dokuwiki" -nodefaults -vga virtio -chardev vc,id=vc0 -mon chardev=vc0 -usb -device usb-tablet -device usb-ehci,id=ehci0 -drive file=cdrom.iso,if=none,media=cdrom,readonly=on,format=raw,id=cdrom0 -drive file=Snapshots/current_state.qcow2,if=none,media=disk,format=qcow2,id=currhdd -drive file=zvC9-swap.qcow2,if=none,media=disk,format=qcow2,id=swap0 -device ide-cd,drive=cdrom0,bootindex=1,unit=1 -device ide-hd,drive=currhdd,bootindex=2,unit=0 -device virtio-blk-pci,drive=swap0,bootindex=3 -netdev user,id=usernet0,ipv6=off -device e1000,netdev=usernet0,bootindex=4 -bios /usr/share/qemu/OVMF.fd

DOWNLOAD_KEYSERVER="hkp://keyserver.ubuntu.com" lxc-create -t download -n test

lxc-ls

lxc-start test

lxc-info test

lxc-execute test /bin./bash

lxc-monitor

lxc-wait […]

lxc-stop test

lxc-console -n test

(Ctrl-a q, to disconnect)

watch lsusb
a2enmod rewrite

$ fallocate -l 12GiB 12.bin

$ truncate --size 4GiB 4.bin

Create wrapped passphrase:

mkdir ~/.ecryptfs

(stty -echo; printf "Passphrase: " 1>&2; read PASSWORD; stty echo; echo 1>&2; head -c 48 /dev/random | base64; echo "$PASSWORD";) \

| ecryptfs-wrap-passphrase ~/.ecryptfs/wrapped-passphrase >/dev/null

Insert wrapped passphrase:

(stty -echo; printf "Passphrase: " 1>&2; read PASSWORD; stty echo; echo "$PASSWORD";) | ecryptfs-insert-wrapped-passphrase-into-keyring ~/.ecryptfs/wrapped-passphrase -

mount with unwrapped key:

KEY_SIG="0123456789abcdef"

mount -i -t ecryptfs /path/.private /path/private -o ecryptfs_sig=${KEY_SIG},ecryptfs_fnek_sig=${KEY_SIG},ecryptfs_cipher=aes,ecryptfs_key_bytes=32,ecryptfs_unlink_sigs

list keys in keyring:

keyctl list @u

udevadm info /dev/sdb

conntrack -L

распаковать arj-архив (опасность, не проверено):
$ unarj e filename.arj

кодировать/декодировать любой файл в ascii:
$ uuencode ...
$ udecode ...

редактирование текстовых файлов:
$ nano 1.txt

$ xed 1.txt
$ pico 1.txt
$ mousepad 1.txt
$ pluma 1.txt
$ gedit 1.txt
$ vi 1.txt
$ vim 1.txt

запуск X Server:
$ startx
$ startx -- :1
или без менеджера окон:
$ xinit

терминал:
$ xfce4-terminal
$ xterm

копирование на/из немонтированную DOS FS:
$ mcopy ...
измерить время выполнения:
$ time command arguments...
считать:
$ echo "2048 / 8" | bc
$ echo "2048 / 8" | calc
запуск sfdisk с исключительной блокировкой устройства:
flock /dev/sdc sfdisk /dev/sdc
дамп и восстановление таблицы разделов:
sfdisk -d /dev/nvme0n1 > gpt.txt
cat gpt.txt > sfdisk /dev/nvme0n1
partprobe /dev/nvme0n1

опасность:
blkdiscard /dev/sdb
blkdiscard /dev/sdb3
fstrim /mnt/mount_point

настройка времени в mint
$ time-admin

перевести mint linux на хранение локального времени (вместо времени UTC (GMT)) в материнской плате:
timedatectl set-local-rtc 1

установить текущее системное время в материнскую плату:
hwclock -w
обратно:
hwclock -s

настройка часового пояса:
dpkg-reconfigure tzdata

настройка комбинации переключения между латинской/национальной раскладкой:
dpkg-reconfigure keyboard-configuration

узнать, HDD или SSD:
lsblk -d -o +name,rota /dev/sda
(если rota = 1, то это HDD. Если rota = 0, это не HDD)
cat /sys/block/sda/queue/rotational
(заменить sda на имя диска. Спикок дисков получить командой lsblk)

определить, работаем ли из виртуальной машины:
$ imvirt

информация о процессоре:
$ lscpu
$ cat /proc/cpuinfo
$ cpuid
установка частоты (уточнить, в каких единицах задаётся частота. В КГц?):
$ cpufreq-set -c 0 -f 1200000
узнать частоту:
$ less /proc/cpuinfo
$ cat /proc/cpuinfo
$ cpufreq-info

опасность (не проверено):
$ before=$(set -o posix; set | sort)
опасность (не проверено, часть 2):
$ comm -13 <(printf %s "$before") <(set -o posix; set | sort | uniq)

опасность (не проверено):
$ eval "printf '%q\n' $(printf ' "${!%s@}"' _ {a..z} {A..Z})"

опасность (не проверено):
$ diff <(echo "$set_before") <(echo "$set_after") | sed -e 's/^> //' -e '/^[[:digit:]].*/d'

опасность (не проверено):
$ before=$(set -o posix; set); dosomestuff; diff <(echo "$before") <(set -o posix; set)

опасность (не проверено):
$ for i in _ {a..z} {A..Z}; do eval "echo \${!$i@}" ; done | xargs printf "%s\n"

страницы документации на kill в разных разделах:
$ man 1 kill
$ man 2 kill
$ man "kill(1)"
$ man "kill(2)"

ещё документация:
(man - способ, взятый из Unix)
$ man bash
(info - документация программой из GNU)
$ info kill

(help - документация на встроенные команды и ключевые слова bash)
$ help cd
(type - определить тип команды или ключевого слова bash или псевдонима (alias))
$ type cd
$ type man
$ type if

выдать список псевдонимов:
$ alias
задать псевдоним:
$ alias "lslapt=ls -lapt"

опасность (не проверено):
$ function dump_vars { local VARNAME ; compgen -v | while read -r VARNAME; do printf "$VARNAME=%q\n" "${!VARNAME}" ; done }

вывести переменные окружения
```
$ ( set -o posix  ; set ) | less -i
$ env
$ declare
$ declare -p
$ declare -xp
$ printenv
```

create vbox vmdk attached to HDD:
создать VirtualBox .vmdk-файл, привязанный к диску sda (найти диск через lsblk):
```
$ VBoxManage internalcommands createrawvmdk -filename
/path/to/sda.vmdk -rawdisk /dev/sda
```
для sdb, sdc, nvme0n1:
```
$ VBoxManage internalcommands createrawvmdk -filename
/path/to/disk.vmdk -rawdisk /dev/sdb
 $ VBoxManage internalcommands createrawvmdk -filename
/path/to/disk.vmdk -rawdisk /dev/sdc
 ...
 $ VBoxManage internalcommands createrawvmdk -filename
/path/to/disk.vmdk -rawdisk /dev/nvme0n1
 $ VBoxManage internalcommands createrawvmdk -filename
/path/to/disk.vmdk -rawdisk /dev/nvme1n1
 ...
```
то же, под windows:
```
 $ VBoxManage internalcommands createrawvmdk -filename C:\sda.vmdk
-rawdisk \\.\PhysicalDrive0
```
или для другого диска:
```
 $ VBoxManage internalcommands createrawvmdk -filename C:\sda.vmdk
-rawdisk \\.\PhysicalDrive1
 $ VBoxManage internalcommands createrawvmdk -filename C:\sda.vmdk
-rawdisk \\.\PhysicalDrive2
 ...
```

запуск виртуальной машины в qemu в режиме UEFI:
```
$ qemu-system-x86_64 … -bios /usr/share/qemu/OVMF.fd
```

(VirtualBox тоже умеет запускать UEFI-машины (виртуальные), см. «EFI» в настройках машины)

что-то про файл:
```
$ apt-file search ovmf
```

что-то про включение/выключение беспроводных устройств:
```
$ rfkill
```

можно смотреть диски и разделы и данные S.M.A.R.T.:
```
$ gnome-disks
```

сообщать о событиях X:
```
$ xev
```

что-то про известных пользователей в системе:
```
$ lslogins
```

сборка программы из исходного кода:
```
$ cd src-prog-1.0.1
$ ./configure --prefix=/opt/custom ; make ; sudo make install
```
или
```
$ meson --prefix=/opt/custom my-builddir ; cd my-builddir ; ninja ; sudo ninja install
```

перезагрузка
```
$ reboot
```
выключение:
```
$ poweroff
```

перезагрузка в настройки UEFI
```
# systemctl reboot --firmware-setup
```
выключение с указанием запуска настройки UEFI после включения
```
# systemctl poweroff --firmware-setup
```

узнать, какая цель (target) запускается при загрузке системы (аналог runlevel)
```
# systemctl get-default
```
установить цель при загрузке системы (текстовый режим):
```
# systemctl set-default multi-user.target
```
установить цель при загрузке системы (графический режим):
```
# systemctl set-default graphical.target
```

отключить автозапуск ssh server:
```
# systemctl disable ssh
```
включить автозапуск ssh server:
```
# systemctl enable ssh
```

остановить ssh server:
```
# systemctl stop ssh
```
запустить ssh server:
```
# systemctl start ssh
```

отключить автозапуск менеджера дисплеев (окно входа)
```
# systemctl disable lightdm.target
```
другие менеджеры дисплеев: gdm, lxdm, ...

заменить текущий интерпретатор команд bash выполнением заданной команды:
```
$ exec [-cl] [-a name] [command [arguments]]
```

узнать о видеокартах:
```
$ inxi -G
```

represent 1024 as a product of prime numbers
разложить на простые множители:
```
$ factor 1024
```

очистить экран:
```
$ clear
```
или
Ctrl+l

show current architecture
узнать архитектуру:
```
$ arch
```
узнать ядро и архитектуру:
```
$ uname -a
```
узнать ядро:
```
$ uname -r
```
узнать как запускалось ядро:
```
$ cat /proc/cmdline
```

создать блочное устройство (/dev/loop0 или /dev/loop1 или ...) с содержимым файла:
```
# losetup --find /path/to/disk_image.img
# losetup --find /path/to/disk_image.img --sector-size 512
```
создать также устройства разделов (/dev/loop0p1, /dev/loop0p2, ...):
```
# losetup --find /path/to/disk_image.img --sector-size 512 --
partscan
```
то же самое, в режиме только чтения:
```
# losetup --find /path/to/disk_image.img --sector-size 512 --
partscan --read-only
```
узнать какие устройства /dev/loopX привязаны к файлам:
```
# losetup --list
```
удалить устройство loop0:
```
# losetup --detach /dev/loop0
```
удалить все такие устройства:
```
# losetup --detach-all
```

редактирование таблицы разделов, создание файловых систем, форматирование, задание флагов, имён разделов, меток тома:
```
# gparted
# gparted /dev/sda
```

работа с таблицей разделов:
```
# parted /dev/sda
```
напечатать информацию:

```
 # parted /dev/sda print
```
"комады в интерактивном режиме parted":
```
 print
   print free
   mktable gpt
   mktable msdos
   unit MiB
   unit B
```
for MBR (msdos) partition table:
```
   mkpart primary ext4 1048576 32000MiB
```
for GPT partition table
```
   mkpart grub 1MiB 9MiB
   print
   set 1 bios_grub on
   rm 1
```
КОНЕЦ "комады в интерактивном режиме parted"
другие программы для работы с таблицей разделов:
```
 # fdisk /dev/sda
 # sfdisk /dev/sda
```
(gdisk - для GPT)
```
 # gdisk /dev/sda
```
выдать список устройств и разделов:
```
 # fdisk -l
```
или
```
 $ lsblk
 $ lsblk -f
```
только разделы (выдать список):
```
 # blkid
```

некоторые страницы документации
(иерархия файловой системы)
```
 $ man hier
 $ man file-hierarchy
```

(кодировка ascii, набор символов unicode)
$ man ascii
$ man unicode

$ man utf-8
(кодировка UTF-8)
$ man charsets
(кодировки)

$ man 7 regex
(регулярные выражения)

Архивирование и сжатие файлов
архивирование rsync:
$ rsync -aH --progress /home/user1 /home/user2 /mnt/backup/ ; sync
что-то архивное через cp:
$ cp -a dir1 dir2 /mnt/backup/

разжимает файл 'file1.gz'

$ bunzip2 file1.bz2
раздать file1.gz:
$ gunzip file1.gz
сжать file1:
$ gzip file1
$ bzip2 file1
сжать file1 с максимальным сжатием:
$ gzip -9 file1

опасность (не проверено):
rar a file1.rar test_file создать rar-архив 'file1.rar' и включить в него файл test_file
rar a file1.rar file1 file2 dir1 создать rar-архив 'file1.rar' и включить в него file1, file2 и dir1
rar x file1.rar распаковать rar-архив
unrar x file1.rar

работа с tar-архивами:
создать tar-архив archive.tar, содержащий файл file1:
$ tar -cvf archive.tar file1
создать tar-архив archive.tar, содержащий файл file1, file2 и папку dir1:
$ tar -cvf archive.tar file1 file2 dir1
показать содержимое архива archive.tar:
$ tar -tf archive.tar
распаковать архив archive.tar:
$ tar -xvf archive.tar
распаковать архив archive.tar в папку /tmp:
$ tar -xvf archive.tar -C /tmp
создать архив archive.tar.bz2 со сжатием с помощью bzip2 (Прим.переводчика. ключ -j работает не во всех *nix системах):
$ tar -cvfj archive.tar.bz2 dir1
разжать с помощью bunzip2 архив archive.tar.bz2 и распаковать его (Прим.переводчика. ключ -j работает не во всех *nix системах):
$ tar -xvfj archive.tar.bz2
создать архив archive.tar.gz со сжатием с помощью gzip:
$ tar -cvfz archive.tar.gz dir1 file1 dir2
разжать архив archive.tar.gz через gunzip и распаковать его:
$ tar -xvfz archive.tar.gz
создать архив archive.tar.xz со сжатием с помощью xz:
$ tar -cvfJ archive.tar.xz dir1file1 dir2
разжать архив archive.tar.xz через "xz -d" и распаковать его:
$ tar -xvfJ archive.tar.xz

zip-архивы
не проверено:
$ zip -r file1.zip file1 создать сжатый zip-архив
создать сжатый zip-архив и со включением в него нескольких файлов и/или директорий
$ zip -r archive.zip file1 file2 dir1
или
$ 7z -tzip a archive.zip file1 file2 dir1
распаковать:
$ unzip file1.zip
или
$ 7z x file1.zip

Специальные атрибуты файлов
опасность (не проверено):
(здесь символ #, если он не является первым непробельным символом, начинает комментарий)
$ chattr +a file1 # позволить открывать файл на запись только в режиме добавления
$ chattr +c file1 # позволяет ядру автоматически сжимать/разжимать содержимое файла.
$ chattr +d file1 # указавет утилите dump игнорировать данный файл во время выполнения backup'a
$ chattr +i file1 # делает файл недоступным для любых изменений: редактирование, удаление, перемещение, создание линков на него.
$ chattr +s file1 # позволяет сделать удаление файла безопасным, т.е. выставленный атрибут s говорит о том, что при удалении файла, место, занимаемое файлом на диске заполняется нулями, что предотвращяет возможность восстановления данных.
$ chattr +S file1 # указывает, что, при сохранении изменений, будет произведена синхронизация, как при выполнении команды sync
$ chattr +u file1 # данный атрибут указывает, что при удалении файла содержимое его будет сохранено и при необходимости пользователь сможет его восстановить
$ lsattr # показать атрибуты файлов

touch -t 0712250000 fileditest модифицировать дату и время создания файла, при его отсутствии, создать файл с указанными датой и временем (YYMMDDhhmm)
find / -name file1 найти файлы и директории с именем file1. Поиск начать с корня (/)
find / -user user1 найти файл и директорию принадлежащие пользователю user1. Поиск начать с корня (/)
find /home/user1 -name "*.bin" Найти все файлы и директории, имена которых оканчиваются на '. bin'. Поиск начать с '/ home/user1'
find /usr/bin -type f -atime +100 найти все файлы в '/usr/bin', время последнего обращения к которым более 100 дней
find /usr/bin -type f -mtime -10 найти все файлы в '/usr/bin', созданные или изменённые в течении последних 10 дней
find / -name *.rpm -exec chmod 755 '{}' \; найти все фалы и директории, имена которых оканчиваются на '.rpm', и изменить права доступа к ним
find / -xdev -name "*.rpm" найти все фалы и директории, имена которых оканчиваются на '.rpm', игнорируя съёмные носители, такие как cdrom, floppy и т.п.
locate "*.ps" найти все файлы, сожержащие в имени '.ps'. Предварительно рекомендуется выполнить команду 'updatedb'
whereis halt показывает размещение бинарных файлов, исходных кодов и руководств, относящихся к файлу 'halt'
which halt отображает полный путь к файлу 'halt'
fuser -km /mnt/hda2 принудительное размонтирование раздела. Применяется в случае, когда раздел занят каким-либо пользователем
umount -n /mnt/hda2 выполнить размонитрование без занесения информации в /etc/mtab. Полезно когда файл имеет атрибуты "только чтение" или недостаточно места на диске
df -h отображает информацию о смонтированных разделах с отображением общего, доступного и используемого пространства (Прим.переводчика. ключ -h работает не во всех *nix системах)
ls -lSr |more выдаёт список файлов и директорий рекурсивно с сортировкой по возрастанию размера и позволяет осуществлять постраничный просмотр
du -sh dir1 подсчитывает и выводит размер, занимаемый директорией 'dir1' (Прим.переводчика. ключ -h работает не во всех *nix системах)

du -sk * | sort -rn отображает размер и имена файлов и директорий, с соритровкой по размеру
rpm -q -a --qf '%10{SIZE}t%{NAME}n' | sort -k1,1n показывает размер используемого дискового пространства, занимаемое файлами rpm-пакета, с сортировкой по размеру (fedora, redhat и т.п.)
dpkg-query -W -f='${Installed-Size;10}t${Package}n' | sort -k1,1n показывает размер используемого дискового пространства, занимаемое файлами deb-пакета, с сортировкой по размеру (ubuntu, debian т.п.)
cal 2022
cat /proc/cpuinfo отобразить информацию о процессоре
cat /proc/interrupts показать прерывания
cat /proc/meminfo проверить использование памяти
cat /proc/swaps показать файл(ы) подкачки
cat /proc/version вывести версию ядра
cat /proc/net/dev показать сетевые интерфейсы и статистику по ним
cat /proc/mounts отобразить смонтированные файловые системы
date 041217002007.00 установить системные дату и время ММДДЧЧммГГГГ.СС (МесяцДеньЧасМинутыГод.Секунды)
clock -w # сохранить системное время в BIOS
shutdown -h now # Остановить систему
init 0
telinit 0
shutdown -h hours:minutes # запланировать остановку системы на указанное время
shutdown -c # отменить запланированную по расписанию остановку системы
shutdown -r now # перегрузить систему
dmidecode -q # показать аппаратные системные компоненты - (SMBIOS / DMI)
hdparm -i /dev/hda # вывести характеристики жесткого диска
hdparm -tT /dev/sda # протестировать производительность чтения данных с жесткого диска
chmod --reference=RFILE FILE...
mount -l
ifconfig -a
ip addr show
whois domain
dig domain
dig -x host
wget file
wget -c file
gzip file
tar -c -J -f 1.tar.xz DIR1 FILE2 ...
man 1 bash
apropos string
which app
whereis app
whatis app
uname -a
uname -r
finger user
locate file
ssh user@host
ssh-copy-id user@host
ssh -p PORT user@host
grep -r pattern DIR

grep pattern FILES...

fg 1
tail -f
apt-get --download-only install
apt download packageName

apt-cache pkgnames
apt-cache search vsftpd
apt-cache pkgnames vsftpd
apt-cache show netcat
apt-cache showpkg vsftpd
apt-cache stats
apt-get update
apt-get update
apt-get dist-upgrade
mintupdate-cli upgrade
apt-get install netcat vsftpd
apt-get install "netca*"
apt-get install packageName --no-upgrade
apt-get install packageName --only-upgrade
apt-get install vsftpd=2.3.5-3ubuntu1
apt-get remove vsftpd
apt-get purge vsftpd
apt-get remove --purge vsftpd
apt-get clean
apt-get --download-only source vsftpd
apt-get source vsftpd
apt-get --compile source goaccess
apt-get download nethogs
apt-get changelog vsftpd
apt-get check
apt-get build-dep netcat
apt-get autoclean
apt-get clean
apt-get autoremove vsftpd

dpkg -i packageName.deb # install
dpkg-deb -f packageName.deb # info from file
dpkg -l
dkpg -l apache2
dpkg -r apache2 # remove
dpkg -p apache2
dpkg -c packageName.deb
dpkg -s packageName # check if it's installed
dpkg -L packageName # check locations?
dpkg -R --install directory-debs # install all debs from directory
dpkg --unpack packageName.deb # unpack, don't configure
dpkg --configure packageName # configure unpacked package
dpkg --uptade-avail packageName # what?
dpkg --clear-avail # ?
dpkg --forget-old-unavail # ?

```
# dpkg --license
# dpkg --version
# dpkg --help
# dpkg --status packageName
# dpkg --search fileName # which package owns that file

# aptitude update
# aptitude install packageName
# aptitude show birthday
# aptitude search packageName
```

информация об NTFS-разделе:
(опасность, не проверено)

```
# ntfsinfo -m /dev/nvme0n1p4
```

перемещение lvm pv:
(могут быть разные размеры секторов на устройствах)
в /etc/lvm/lvm.conf (на случай разных размеров секторов):

```
allow_mixed_block_sizes = 1
```

далее:

```
# pvcreate /dev/nvme0n1p9
# vgextend vg_name /dev/nvme0n1p9
# pvmove /dev/nvme1n1p9 /dev/nvme0n1p9
# vgreduce vg_name /dev/nvme1n1p9
```

heredoc

Остальное

Установка пакетов в Mint

Основной список для mint:

apt install nmap nmapsi4 geany htop vlock pwgen screen mc qbittorrent rtorrent memtest86+ aqemu calc florence galculator qtcreator g++ gcc gcc-doc autoconf automake libtool libtool-doc autoconf-doc gimp gparted gimp-help-ru hplip-gui build-essential aptitude shotcut openshot flowblade kdenlive xplayer mplayer smplayer dragonplayer mint-meta-mate mint-meta-cinnamon vlc remmina remmina-plugin-nx remmina-plugin-spice remmina-plugin-rdp keepass2 keepass2-doc keepassx keepassxc fritzing pidgin gajim kopete kde-telepathy parted gdisk fdisk powertop iotop dnstop x11vnc linuxvnc ssvnc tigervnc-viewer tigervnc-standalone-server imagemagick imagemagick-doc ansible ansible-doc kde-full lxde lxdm lxqt enlightenment ephoto efl-doc terminology mousepad xed pluma gedit kate kwrite joe vim vim-doc shotwell gnome lxmusic kdepim kinfocenter kwave audacity flac moc qmmp timidity lame lame-doc ffmpeg ffmpeg-doc kmplayer brasero xfburn xorriso genisoimage kaffeine k3b lxlock x11vnc xscreensaver peek kazam simplescreenrecorder recordmydesktop tilda vokoscreen-ng valgrind gdb xfce4-xkb-plugin whois procinfo hwinfo x2vnc wireless-tools wget warpinator vtgrab vmtouch traceroute transmission-gtk syslinux-utils squashfs-tools squashfs-tools-ng smbclient smartmontools sweeper sudo simple-scan shotcut sensible-utils sed rsync reiserfsprogs qt5-gtk2-platformtheme pv psmisc procps procinfo phototonic pavucontrol alsamixer patch xsane testdisk gddrescue

big list:

apt install geany htop vlock pwgen screen mc qbittorrent rtorrent memtest86+ aqemu calc florence galculator qtcreator g++ gcc gcc-doc autoconf automake libtool libtool-doc autoconf-doc

gimp gparted gimp-help-ru hplip-gui build-essential aptitude shotcut openshot flowblade kdenlive xplayer mplayer smplayer dragonplayer mint-meta-mate mint-meta-cinnamon vlc remmina remmina-plugin-nx remmina-plugin-spice remmina-plugin-rdp keepass2 keepass2-doc keepassx keepassxc fritzing pidgin gajim kopete kde-telepathy parted gdisk fdisk powertop iotop dnstop x11vnc linuxvnc ssvnc tigervnc-viewer tigervnc-standalone-server imagemagick imagemagick-doc ansible ansible-doc kde-full lxde lxdm lxqt enlightenment ephoto efl-doc terminology mousepad xed pluma gedit kate kwrite joe vim vim-doc shotwell gnome lxmusic kdepim kinfocenter kwave audacity flac moc qmmp timidity lame lame-doc ffmpeg ffmpeg-doc kmplayer brasero xfburn xorriso genisoimage kaffeine k3b lxlock x11vnc xscreensaver peek kazam simplescreenrecorder recordmydesktop tilda vokoscreen-ng valgrind gdb xfce4-xkb-plugin whois procinfo hwinfo x2vnc wireless-tools wget warpinator vtgrab vmtouch traceroute transmission-gtk syslinux-utils squashfs-tools squashfs-tools-ng smbclient smartmontools sweeper sudo simple-scan shotcut sensible-utils sed rsync reiserfsprogs qt5-gtk2-platformtheme pv psmisc procps procinfo phototonic pavucontrol alsamixer patch xsane testdisk gddrescue

additional packages:
apt install
чтобы собрать qemu-7.0.0:
apt install libsdl2-dev libsdl2-image-dev libgtk3.0-cil-dev python3-sphinx python3-sphinx-bootstrap-theme libgnutls28-dev libusb-1.0-0-dev libopenusb-dev \
libvde-dev libvncserver-dev libvdeplug-dev libgtkmm-3.0-dev libusb-1.0-0-dev libcap-ng-dev libattr1-dev python3-sphinx-rtd-theme
apt-get install htop vlock pwgen screen mc xorriso aptitude vlc gimp gimp-help-ru gparted aqemu qbittorrent rtorrent hplip-gui geeqie imagemagick-doc g++ gcc-doc xscreensaver memtest86+ brasero xfburn guvcview recordmydesktop pidgin gajim kopete mint-meta-cinnamon mint-meta-mate mint-meta-codecs
apt-get install peek kazam simplescreenrecorder recordmydesktop vokoscreen-ng valgrind procinfo xviewer xreader xrdp xplayer smplayer smplayer-l10n smtube smplayer-themes xed mousepad joe nano x2vnc wireless-tools wget warpinator vtgrab vmtouch traceroute transmission-gtk syslinux-utils ssvnc squashfs-tools squashfs-tools-ng smbclient smartmontools simplescreenrecorder sweeper sudo simple-scan shotcut sensible-utils sed rsync remmina remmina-plugin-nx remmina-plugin-spice reiserfsprogs qtcreator qt5-default qt5-gtk2-platformtheme qemu qemu-utils pv psmisc procps procinfo powertop pix phototonic pavucontrol alsamixer patch parted p7zip-full openvpn openshot onboard okular ntfs-3g ntp ntpdate nmap nmapsi4 net-tools mugshot minicom make lsof lshw linuxvnc ledit laptop-detect kate kwrite kdenlive kpartx kamera
apt install geany htop vlock pwgen screen mc qbittorrent rtorrent memtest86+ aqemu calc florence galculator qtcreator g++ gcc gcc-doc autoconf automake libtool libtool-doc autoconf-doc gimp gparted gimp-help-ru hplip-gui build-essential aptitude shotcut openshot flowblade kdenlive xplayer mplayer smplayer dragonplayer mint-meta-mate mint-meta-cinnamon vlc remmina remmina-plugin-nx remmina-plugin-spice remmina-plugin-rdp keepass2 keepass2-doc keepassx keepassxc fritzing pidgin gajim kopete kde-telepathy parted gdisk fdisk powertop iotop dnstop x11vnc linuxvnc ssvnc tigervnc-viewer tigervnc-standalone-server imagemagick imagemagick-doc ansible ansible-doc kde-full lxde lxdm lxqt enlightenment ephoto efl-doc terminology mousepad xed pluma gedit kate kwrite joe vim vim-doc shotwell gnome lxmusic kdepim kinfocenter kwave audacity flac moc qmmp timidity lame lame-doc ffmpeg ffmpeg-doc kmplayer brasero xfburn xorriso genisoimage kaffeine k3b lxlock x11vnc xscreensaver peek kazam simplescreenrecorder recordmydesktop tilda vokoscreen-ng valgrind gdb xfce4-xkb-plugin whois procinfo hwinfo x2vnc wireless-tools wget warpinator vtgrab vmtouch traceroute transmission-gtk syslinux-utils squashfs-tools squashfs-tools-ng smbclient smartmontools sweeper sudo simple-scan shotcut sensible-utils sed rsync reiserfsprogs qt5-gtk2-platformtheme pv psmisc procps procinfo phototonic pavucontrol alsamixer patch xsane

apt-get --download-only install samba openssh-server vsftpd ftpd apache2 libapache2-mod-php funny-manpages xviewer xreader atril evince xrdp smtube

GrUB

Установка GrUB:

sudo grub-install /dev/sda --target i386-pc
sudo grub-install /dev/sda --target x86_64-efi

grub rescue commands:

ls
ls (hd0,3)/
ls (hd0,3)/boot/
ls (hd0,3)/boot/grub
set root=(hd0,3)
set prefix=(hd0,3)/boot/grub
insmod normal
normal

grub commands

halt
reboot
loopback loop0 (hd0,4)/1.iso
set root=(hd0,3)

/etc/default/grub:

```
GRUB_DEFAULT=0
#GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=7200
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
#GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_CMDLINE_LINUX=""
```

env

DISPLAY=:0.0 HOME LOGNAME LC_ALL

ссылки: symbolic, hard

программирование в bash
for, while, if-then-elif-else-fi, case-in-esac, until, do-while?, do-until?
'\' в конце строки в скрипте
$(()), $[], $(), ``,, ., source, :, true, false, test, [ ]
trap - захват сигналов и других событий
function f() {list;} g(){list;} function h{list;}
shift
"$*", "$@", $@, $*, $#, $0, $1, $2, ...

bash:
!! - повторить последнюю команду

запуск bash:
bash -x
bash -c "echo 123"

sed:
sed "s/to_be_replaces/new_string/g"

GNU screen. комбинации клавиш

Ctrl+A_Ctrl+D

шаблоны bash
*, ?

таблицы разделов

gpt msdos(MBR)

создание скриптов
#!/path/to/interpreter argument1

Файлы

/etc/passwd /etc/group /etc/fstab /etc/issue /etc/os-release /etc/services /etc/protocols
/etc/hostname /etc/lsb-release /etc/hosts /etc/shadow /etc/network/interfaces
/etc/resolv.conf
/dev/sda /dev/sda1 /dev/sr0 /dev/disk/by-.../...

/etc/default/grub
/etc/grub.d/40_custom

/dev/mem
/dev/md0
/dev/mapper/...
/proc/cpuinfo /proc/meminfo /proc/devices

/proc/cmdline - как запущено ядро

/sys/firmware/efi - present if booted in UEFI mode
/sys/firmware/efi/efivars - EFI variables

/sys/block/<имя_диска>/queue/rotational - 0, если не HDD, 1, если HDD
например, # cat /sys/block/sdc/queue/rotational

Кнопки при работе с bash

Tab: дописать оставшуюся часть команды или аргумента команды (путь к файлу или другое)

Tab несколько раз: показать кандидаты на дополнение текущей части командной строки

Стрелка вверх: перемещение по истории команд к более ранним

Стрелка вниз: перемещение по истории команд к более поздним

Ctrl+c: отменить ввод командной строки или прервать выполняющуюся команду (послать сигнал SigINT)

Ctrl+l (или команда clear): очистить экран

Ctrl+d: конец воода (EOF, end of file) или выход из bash

Ctrl+z: поставить текущую команду на паузу (fg НОМЕР - вернуть из паузы (например, fg 1; по команде jobs можно смотреть список задач, поставленных на паузу))

Переменные среды

LD_LIBRARY_PATH, PATH, HOME, PKG_CONFIG_PATH, CPPFLAGS, LDFLAGS

Можно определить и экспортировать LC_ALL=C (команда «export LC_ALL=C»), так будут перкрываться все заданные в других местах настройки языка и форматов вывода

LC_ALL LANG LANGUAGE

LC_ADDRESS LC_MONETARY LC_PAPER LC_IDENTIFICATION LC_NAME LC_TELEPHONE LC_MEASUREMENT LC_NUMERIC LC_TIME

Комбинации клавиш

В виртуальной текстовой консоли: Ctrl+Alt+Del = перезагрузка

В графическом режиме (виртуальная графическая консоль): Ctrl+Alt+Backspace = убить X-сервер

В терминалах и текстовой консоли (виртуальной): Ctrl+d - конец ввода (EOF) или выход из bash. Ctrl+c: прервать процесс (послать сигнал SigINT). Ctrl+Z: поставить процесс на паузу (fg 1, fg 2, fg ... - чтобы вернуть процесс из паузы)

Ctrl+l - очистить экран

В любом режиме:

От Ctrl+Alt+F1 до Ctrl+Alt+F12: переключение на vt (виртуальную консоль) с указанным номером

Ctrl+Alt+SysRq+S - сбросить кэш записи на диск. Ctrl+Alt+SysRQ+U - Отмонтировать все файловые системы и те, что нельзя отмонтировать, перемонтировать в

режим только чтение. Ctrl+Alt+SysRQ+B - мгновенная перезагрузка. Ctrl+Alt+SysRQ+O - мгновенное выключение ПК.

В графическом режиме:

Alt+Tab - переключение между окнами

Alt+F4 - закрыть текущее окно

Alt+F10 - развернуть/восстановить текущее окно

Alt+F9 - свернуть текущее окно

Alt+зажать_правой_конпкой_мыши_на_окне_и_тянуть - изменение размера окна

Alt+перемещение_левой_кнопкой_мыши - переместить окно за любую точку внутри окна

Ctrl+Shift+u, затем код символа unicode (шестнадцатиричный), затем пробел = ввод символа по коду (unicode). Например, код a0 - неразрывный пробел

(символы и коды смотреть в программе gucharmap)

Ctrl+Alt+T = запустить эмулятор терминала (работает в Xfce, cinnamon, gnome, mate, kde; не работает в enlightenment, LXDE, OpenBox)
Ctrl+C - скопировать текст в буфер обмена
Ctrl+Z - отменить предыдущее изменение в редакторе
Ctrl+D, Ctrl+L, Ctrl+W, Ctrl+U

В некоторых файлах можно писать комментарии, начинающиеся с символа '#' (решётка). Программы, обрабатывающие файлы, игнорируют комментарии.
В bash слово, начинающееся с символа '#', и всё, что идёт в строке после этого слова, - комментарий. Если символ '#' идёт внутри слова, он не начинает комментарий

Пакеты Debian/MX_Linux (не Mint Linux)

isenkram isenkram-cli

Talks about using octal mode with chmod and about permissions

you have 3 bits special and 3 groups of 3 bits each for user(owner), group, others. so, 3+3*3=12 bits:
SUID, SGID, Sticky bit, ownerR, ownerW, ownerX, groupR, groupW, gourpX, others**s**R, othersW, othersX

mode 7654 sets 7 for SUID, SGID, StickyBit, 6 for user(owner) RWX, 5 for group RWX, 4 for others RWX
7=0b111, 6=0b110, 5=0b101, 4=0b100, so bits are 111 110 101 100
Special = 111 = SUIG on, SGID on, Sticky Bit on
User(owner) = 110 = R on, W on, X off
Group = 101 = R on, W off, X on
Others = 100 = R on, W off, X off

«chmod 0640» is rw, 0440 is ro. Detailed mode:
0640 = rw-r---- (rw for user, r for group and nothing for others, 3 blocks of 3 bits (3 bits user, 3 bits group, 3 bits others))
0440 = r--r----
here, 6=0b110, 110 corresponds to rwx for owner (u), so r=1, w=1, x=0
in 0640, 4=0b100 (rwx), for group, so group r=1, group w=0, group x=0 (group mode is r--)
in 0640, 2nd 0 = 0b000 (rwx), for others, so other r=w=x=0

(r=read, w=write, x=execute file or use forlder (cd into it or work with it's contents))

in 0750, we have same order: special, user, group, others. Special = 0b000 (setuid, setgui, sticky bit),
so suid=sgit=sticky=0. user(owner)=7=0b111 (rwx), so r=w=x=1. group=5=0b101 (rwx), so r=1, w=0, x=1
so, 0750 = rwxr-x--- (see ls -l)

1st digit means SUID, SGID, StickyBit, so
7750 = rwsr-s--T
7751 = rwsr-s--t
5751 = rwsr-x--t
5741 = rwsr----t
5740 = rwsr----T

7651 = rwSr-s--t

(yu have 12 bits permissions: suid, sgid, sticky, ownerR, ownerW, ownerX, groupR, groupW, gourpX, otherxR, othersW, othersX;
suid = Set User ID, SGID = Set Group ID. If you have SUID on executable, any user able to execute it will get UID of the owner of the executable for that process (launched form the executable);
run "$ sudo find / -type f -perm /4000" to see all SUID files)

disable write for all (-R = recursively):
$ chmod -c -R a-w dir_or_file
enable write for owner (-R = recursively):
$ chmod -c -R u+w dir_or_file
change owner to $USER and group to it's primary group (-R = recursively):
$ sudo chown -c -R ${USER}: dir_or_file
(notice colon after username)
change owner to $USER (-R = recursively):
$ sudo chown -c -R ${USER} dir_or_file
change group (-R = recursively):
$ sudo chgrp -c -R sudo   dir_or_file

disable access to others:
$ chmod -c -R o= dir_or_file
or
$ chmod -c -R o-rwxst dir_or_file

gain membership in some group:
$ newgrp sudo
$ newgrp kvm
(it can ask for group password; also, your GID becomes kvm's ID (run "id"))
set group password:
$ sudo gpasswd kvm
delete group password (will be unable to newgrp with password):
$ sudo gpasswd -r kvm

also, there are other permission-related settings, I don't use them ("$ man getfacl", "$ man setfacl")

Ссылки

1. Сборник на английском: https://www.tecmint.com/free-online-linux-learning-guide-for-beginners/

2. «Что нужно знать юзверям и не только...: Повесть о вопросах и запросах». Автор acetone https://linuxmint.com.ru/viewtopic.php?f=24&t=35
3. Основные команды консоли#: Справочник команд с их описанием на русском находится по адресу http://www.zabrosov.ru так же можете посмотреть тему на нашем форуме (примечание: данный фрагмент текста откуда-то копировался) находящуюся по адресу https://linuxmint.com.ru/viewtopic.php?f=37&t=37 еще можно посмотреть http://qps.ru/rDozs https://clck.ru/HkobX
4. Основные рекоммендации по ускорению работы Linux Mint на слабых ПК: Тюнинг использования ОЗУ, настройки окружения и программ https://linuxmint.com.ru/viewtopic.php?f=45&t=173
5. Базовая установка видеодрайвера в Linux Mint: https://linuxmint.com.ru/viewtopic.php?t=46
6. Учебные материалы Uneex (в т.ч., видеозаписи лекций на факультете ВМиК (ВМК) МГУ им. М.В. Ломоносова и местами тексты к ним; в т.ч., лекции по разработке программ): https://uneex.ru/
7. Учебные материалы школы 179, г. Москва: https://server.179.ru/

Ссылки, взятые со страницы от acetone

Первая часть

1. Руководство Mint 17.3 (pdf): https://goo.gl/UNlFFn
2. Введение в Ubuntu: http://help.ubuntu.ru/wiki/руководство_по_ubuntu_desktop_14_04/введение
3. Ubuntu 14.04: http://help.ubuntu.ru/wiki/руководство_по_ubuntu_desktop_14_04

Вторая часть

1. Загрузчик GrUB: http://help.ubuntu.ru/wiki/grub
2. Восстановление GrUB: http://help.ubuntu.ru/wiki/восстановление_grub
3. Подробнее: http://forum.ubuntu.ru/?topic=74165.0/

Третья часть

1. Настройка сети (детально): http://help.ubuntu.ru/wiki/настройка_сети_вручную/
2. Настройка роутеров: http://rudevice.ru/
3. Список терминальных команд: http://help.ubuntu.ru/wiki/командная_строка?&#шпаргалка_команд

Дополнительно

4. страницы man: http://www.opennet.ru/man_1.shtml
5. пакеты Ubuntu: http://packages.ubuntu.com/
6. другие пакеты: http://pkgs.org/

7. что-то про драйвера (коллекция драйверов?): https://driverscollection.com/

что-то про Nvidia http://help.ubuntu.ru/совместимость/nvidia

Форумы Родственников Минта (дистрибутивов?)

1. Ubuntu: http://forum.ubuntu.ru/
2. Gentos: http://www.rhelforum.ru/
3. OpenSUSE: http://open-suse.ru/forum
4. Debian: http://debianforum.ru/
5. SlackWare: http://www.slackware.ru/forum/
6. Gentoo: http://forums.gentoo.org/index.php
7. ArchLinux: http://archlinux.org.ru/forum/
8. Fedora: http://russianfedora.pro/
9. Xubuntu: http://forum.xubuntu-ru.net/index.php
10. Lubuntu: http://www.lubuntu.ru/forum

Разное

1. Руководства и статьи: http://help.ubuntu.ru/wiki/howto
2. Команды терминала: http://www.f-notes.info/linux:linux_command
3. Шпаргалка: http://forum.ubuntu.ru/index.php?topic=14535.0
4. Bash: http://rus-linux.net/MyLDP/HOWTO-ru/Bash-Progr-Intro-HOWTO/Bash-Prog-Intro-HOWTO.html#toc14
5. Дистрибутивы Linux Mint: https://www.linuxmint.com/download_all.php
6. Дистрибутивы на любой вкус: http://releases.ubuntu.com/
7. Форум JAVA: http://javatalks.ru/
8. Большая шпаргалка: http://wiki.fornex.com/index.php/Команды_Linux
9. Что-то про создание загрузочных флешек: http://help.ubuntu.ru/wiki/unetbootin

О программировании

1. Shell: http://www.codenet.ru/progr/other/sh/
2. Bash: http://rus-linux.net/MyLDP/HOWTO-ru/Bash-Progr-Intro-HOWTO/Bash-Prog-Intro-HOWTO.html
3. C: http://cpp.com.ru/

4. C++: http://cpp.com.ru/
5. PHP: http://i-vd.org.ru/books/php/
6. Vala: https://wiki.gnome.org/Projects/Vala/Tutorial/ru
7. PHP: http://www.php-spravka.ru/