Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Практическое занятие № 2. Знакомство с командной оболочкой shell

Студент: Валдайцев А. Д.

ФИТ 4 курс 5 группа

Преподаватель: Сазонова Д. В.

Минск 2023

# Теоретические сведения

# Оболочка и дочерние процессы

При вводе команды, например, $**ls,** оболочка находит файл программы **/bin/ls** и передает его ядру для исполнения. Ядро выделяет память для нового дочернего процесса, загружает программу и выполняет инструкции, содержащиеся в **/bin/ls.** По завершении программы ядро освобождает выделенную процессу память и возвращает управление родительской программе оболочки. Пример процесса представлен на рисунке 1.1.

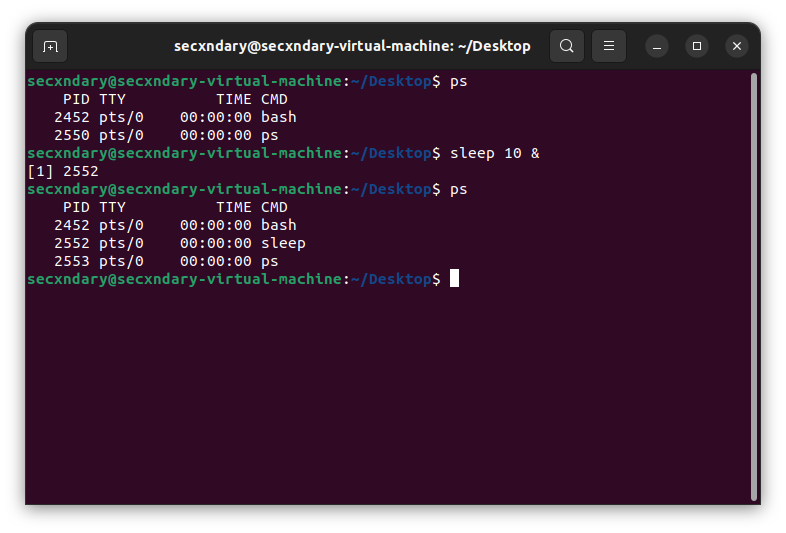


Рисунок 1.1 – Вывод команд ps и sleep 10 &

В результате выводится перечень выполняющихся в данный момент команд: программа оболочки и программа **ps.** Первая команда создает дочерний процесс sleep, подлежащий выполнению в фоновом режиме, который отображается посредством команды ps. При каждом вводе команды создается дочерний процесс, который выполняется независимо от родительского процесса или оболочки. Это оставляет родительский процесс неизменным при выполнении другой задачи.

# Интерпретация командной строки

Основной формат командной строки выглядит следующим образом:

*команда аргументы*

Команда— это исполняемая команда UNIX, программа, утилита или программа оболочки.

Аргументыпередаются исполняемому модулю. Большинство программ утилит UNIX предполагают, что аргументыбудут иметь следующую форму:

*опции имена\_файлов*

Например, в командной строке

*$ ls -l file1 file2*

имеется три аргумента команды ls; первый является опцией, а последние два — именами файлов.

Вывод данной команды представлен на рисунке 1.2.

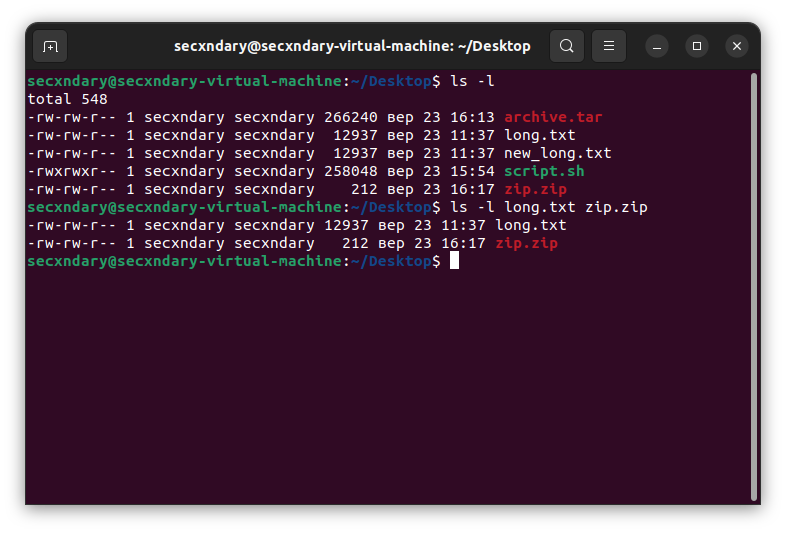


Рисунок 1.2 – Вывод команды ls с параметрами в виде имен файлов

Одно из действий, выполняемых shell для ядра — исключение необязательной информации. Так, применительно к компьютеру одним из типов необязательной информации являются пробелы; следовательно, важно знать, что делает оболочка, встречая пробел. Пробелами являются символы пробелов, горизонтальной табуляции и символы новой строки. Рассмотрим следующий пример:

$ echo part A part В part С part A part В part С

В данном случае оболочка интерпретирует командную строку как команду echo**,** имеющую шесть аргументов, и удаляет пробелы между ними. Например, при печати заголовков отчета и при необходимости сохранить пробелы, нужно было бы заключить данные в символы кавычек, как в следующем примере:

$echo 'part A part В part С part A part B part С'

Символ одинарных кавычек мешает оболочке “заглядывать” внутрь кавычек. Теперь оболочка интерпретирует эту строку как команду echoс единственным аргументом, который в данном случае является строкой символов, содержащей пробелы.

Пример выполнения данных команд представлен на рисунке 1.3.

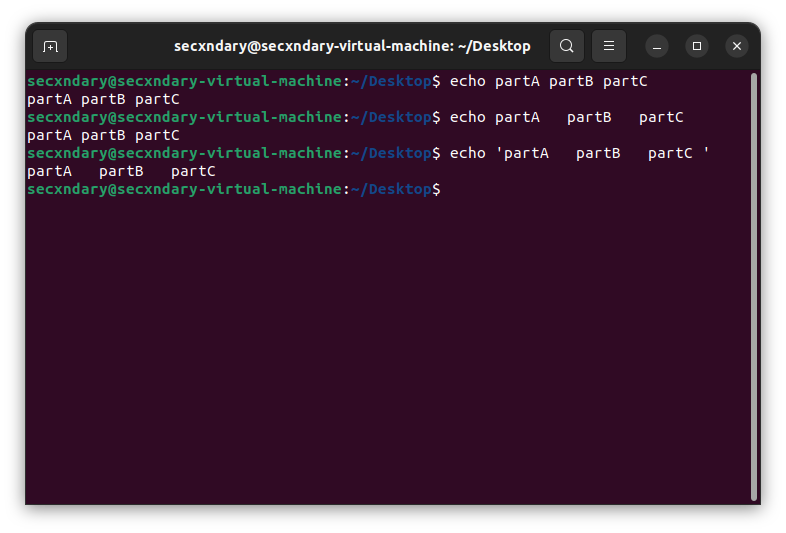


Рисунок 1.3 – Вывод команды echo

# Метасимволы оболочки

Шаблоны могут быть удобны при обработке ряда специфичных файлов. Ниже приведен ряд примеров:

$ls t\*

Эта команда выводит все файлы, чьи имена начинаются с t.

$ls test?5.dat

Эта команда выводит перечень всех файлов, имена которых начинаются с test, далее следует любой одиночный символ, а затем — 5.dat.

$ls [а-с]\*

Эта команда выводит перечень всех файлов, имена которых начинаются с символов от а до с.

$ls [e,m,t]\*

В результате выполнения этой команды отображается список всех файлов, имена которых начинаются с е, m или t.

Пример выполнения данных команд представлен на рисунке 1.4.

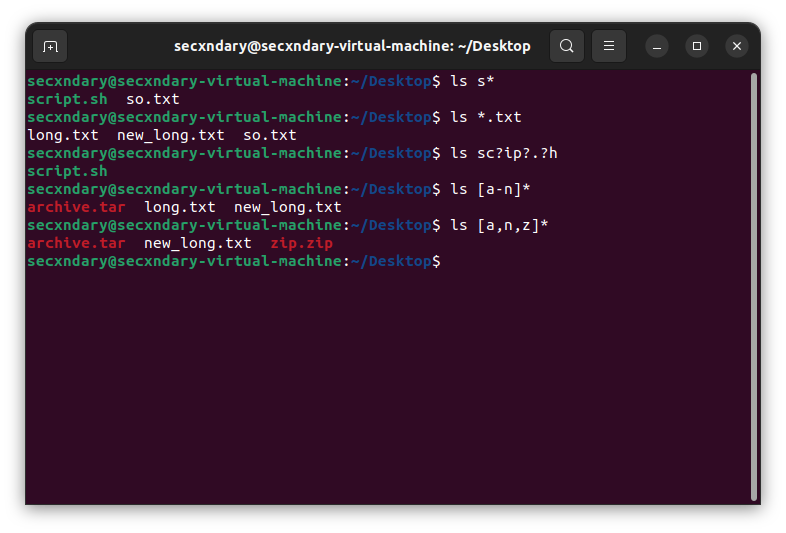


Рисунок 1.4 – Вывод команд ls с различными параметрами

# Перенаправления ввода/вывода и программные каналы

Большинство команд принимают ввод с клавиатуры и отправляют вывод на его дисплей. Для переназначения вывода в файл используется символ '>'. Например, команда $ls > desktop\_files выводит список файлов текущего каталога и помещает его в файл, названный desktop\_files. Пример вывода данной команды представлен на рисунке 1.5.

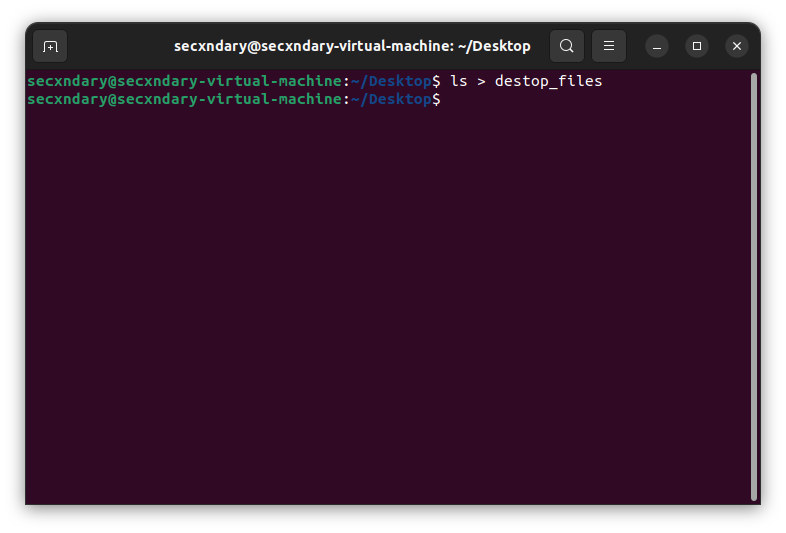


Рисунок 1.5 – Вывод команды ls > desktop\_files

Результатом выполнения данной команды стало создание в папке Desktop/ файла desktop\_files с содержанием, представленным на рисунке 1.6.

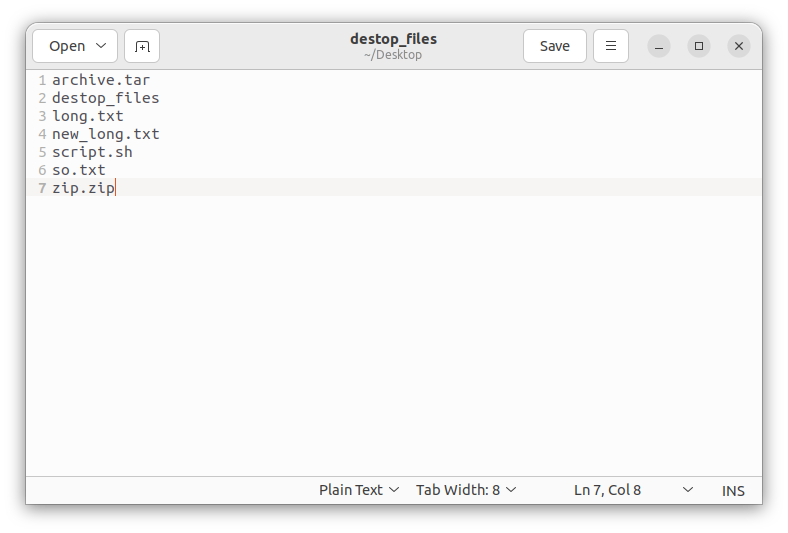


Рисунок 1.6 – Содержимое файла desktop\_files

Для направления вывода одной команды во ввод следующей можно использовать символ | (канал). Например, команда $ls -s | -nr | pg выдает список файлов текущего каталога и их размер в блоках, направляет вывод в команду sort, которая упорядочивает файлы в порядке уменьшения номеров, а затем направляет вывод в команду формирования страниц pg для окончательного отображения на мониторе терминала.

Пример вывода данной команды представлен на рисунке 1.7.

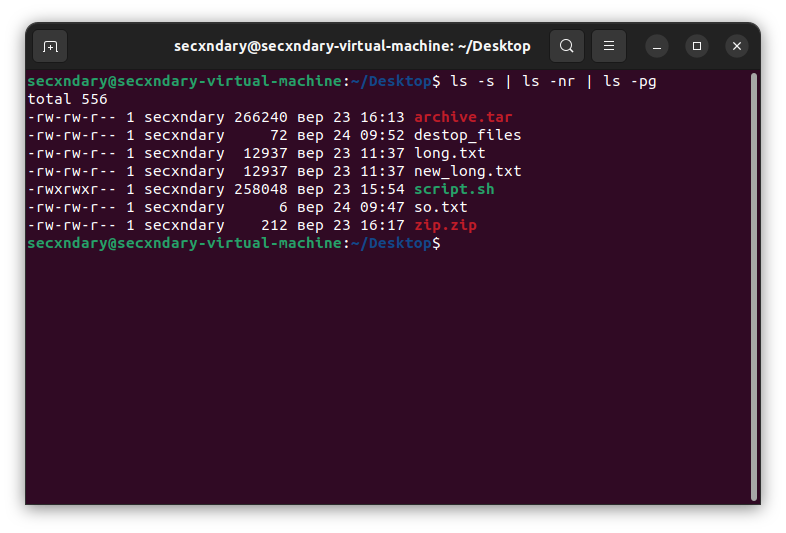


Рисунок 1.7 – Вывод команд с символом канала

# Подстановка команд

Подстановка команд аналогична переназначению, за исключением того, что она используется для получения аргументов команды из вывода другой команды. Например, команда

$grep 'wc -l myfiles` \*'

берет число строк в файле myfiles из команды wc и помещает это число в качестве аргумента в команду grep для отыскания в текущем каталоге всех файлов, соответствующих этому критерию.

Пример вывода данной команды представлен на рисунке 1.8.

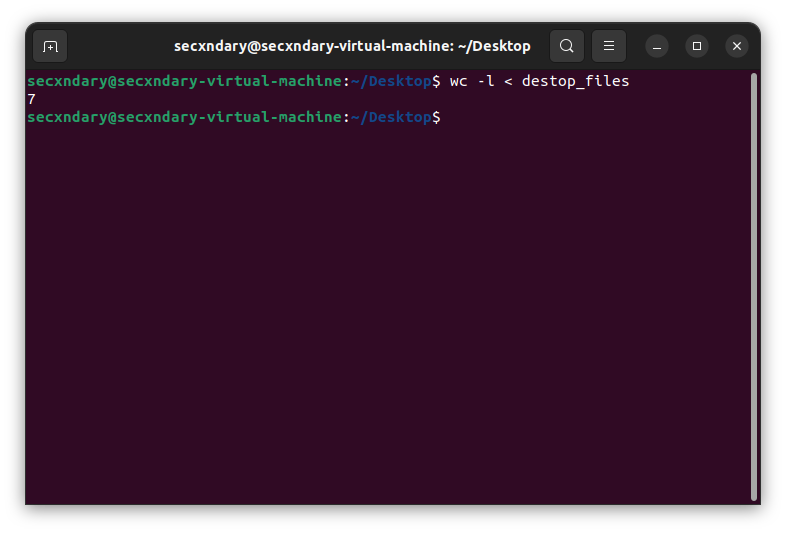


Рисунок 1.8 – Вывод команды wc

# Поддержка переменных окружения оболочки

Оболочка может поддерживать переменные. Переменные— это области памяти, в которых можно хранить данные для последующего использования. Значение переменной присваивается с помощью знака равенства (=):

LOOKUP=/usr/mydir

В этой строке оболочка устанавливает LOOKUP в качестве переменной и присваивает ей значение /usr/mydir. В дальнейшем можно использовать значение, сохраненное в LOOKUP, в командной строке, предваряя имя переменной знаком доллара ($).

Чтобы сделать переменную доступной дочерним процессам, можно использовать команду export, например:

$ export LOOKUP

Пример вывода данных команд представлен на рисунках 1.9 и 1.10.

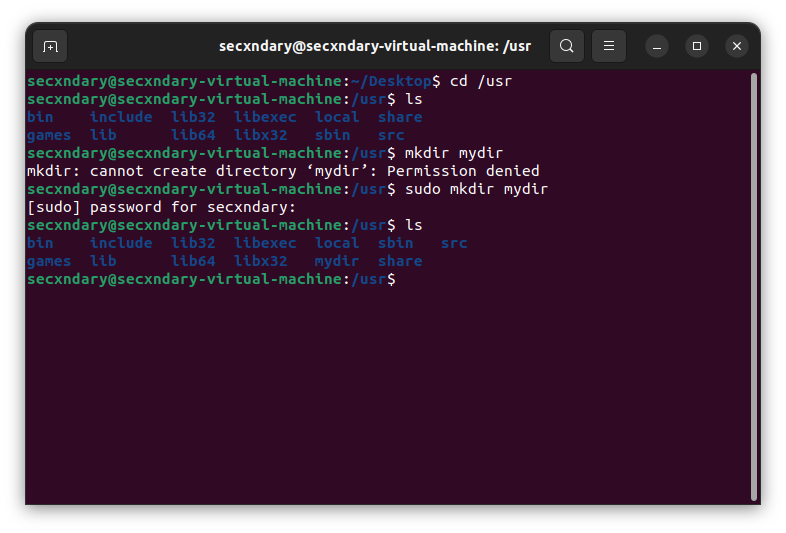


Рисунок 1.9 – Создание директории mydir

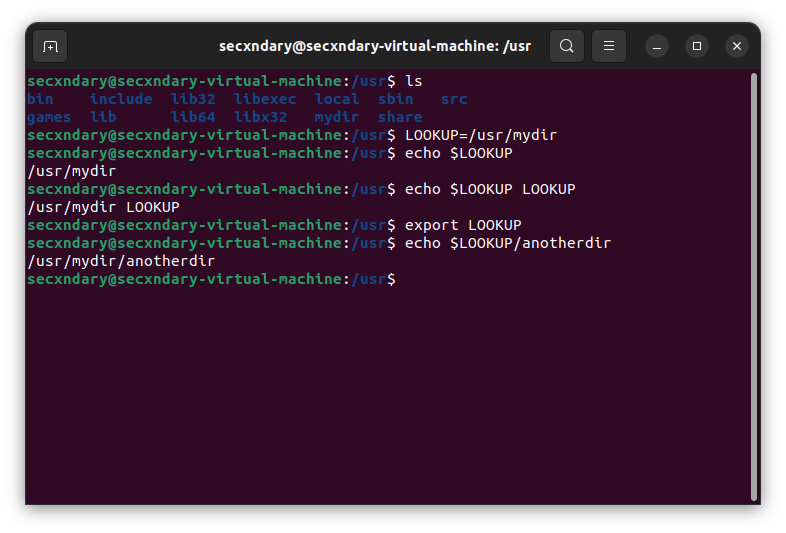


Рисунок 1.10 – Создание переменной окружения LOOKUP

# Переменные среды оболочки

Когда программа login вызывает оболочку, она устанавливает переменные среды, которые считываются из файлов инициализации оболочки **/etc/profile** и **.profile.** Обычно эти файлы устанавливают тип терминала в переменной $TERMи путь по умолчанию для исполняемых файлов — в переменной $РАТН.Посмотрим следующие примеры:

$ echo $term

$ echo $PATH

Переменные легко можно изменить так же, как присваиваются значения любой переменной оболочки.

Пример изменения данных переменных представлен на рисунке 1.11.

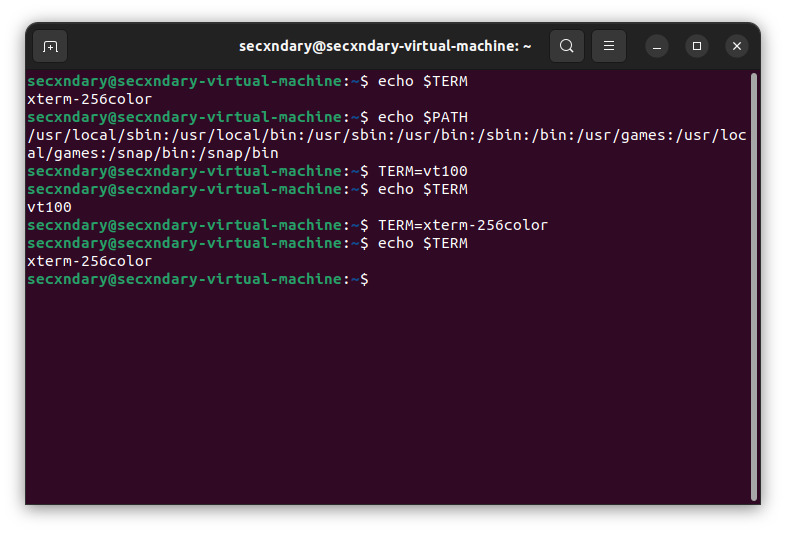


Рисунок 1.11 – Изменение системных переменных среды

# Файлы начального запуска оболочки

Файл **.profile** — локальный файл начального запуска для оболочки Bourne. Оболочка Korn использует файл **.kshrc**, а оболочка С — файл **.cshrc.** Для манипулирования средой начального запуска эти файлы можно редактировать. При необходимости можно добавлять дополнительные переменные. Можно также добавлять программирование оболочки для получения установок среды, зависящих от различных условий.

Местоположение файла **.profile** в командной оболочке mc представлено на рисунке 1.12.

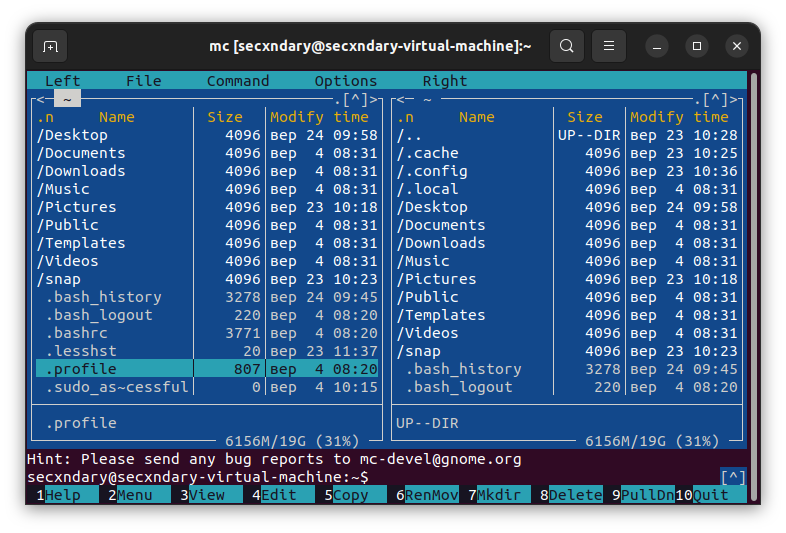


Рисунок 1.12 – Вывод файла .profile в командной оболочке mc

# Наиболее часто используемые команды UNIX

|  |  |  |
| --- | --- | --- |
| Команда Linux | Краткое описание команды | Снимок экрана |
| . | Задаёт командному процессору выполнить все команды в файле, который передан в качестве параметра команды. |  |
| & | Запуск команды в фоновом режиме |  |
| adduser | Создание пользователя (необходимы права суперпользователя) |  |
| alias | Создание алиасов. alias выведет список созданных сокращений.  alias alias\_name=’command’ создаст новый алиас. |  |
| apropos | Поиск команды по фрагменту ее названия или описанию |  |
| banner | Вывод на экран символьного баннера |  |
| bg | Перевод приостановленного процесса (с помощью Ctrl+Z) в фоновый режим (по аналогии с &) |  |
| cat | Вывод в консоль содержимого файла. При вводе двух файлов как параметров выведет их содержимое подряд.  “-n число“ выведет определенное кол-во строк. |  |
| cd | Переход в каталог |  |
| chgrp | Изменение группы и владельца файла. Позволяет работать также с символьными ссылками |  |
| cgmod | Изменение прав доступа к файлу.  4 основных группы прав: r (read), w (write), x (exec), s (root exec).  3 группы пользователей: u (владелец файла), g (группа файла), o (остальные юзеры).  +/- включает/отключает права.  Например, u+x – разрешить выполнение для владельца, o-w – запретить запись для других юзеров, ugo+rwx – разрешить все для всех. Можно записать в шестнадцатеричном виде: 0 – без прав, 4 – readonly, 7 – rwx.  Можно записать цифры подряд [ugo], например, 777 – всем разрешить все.  -R так же поддерживает рекурсию, -v выводит максимум информации. |  |
| chown | Изменение владельца файла. Может выполнять только суперпользователь.  -c – подробный вывод изменений, -f – тихий режим.  chown user filename.  chown root:root filename для изменения и владельца, и группы. -R применяется не только к каталогу, но и ко всем вложенных подкаталогам и файлам. |  |
| chroot | Создаёт каталог / (называемый корневым), чтобы он отличался в файловой системе от любого другого. |  |
| cp | Копирование файлов в каталогов:  cp options source dest.  При копировании папок добавляется опция -r для рекурсивного копирования. |  |
| dd | Преобразование форматов файлов. |  |
| env | Вывод переменных окружения. Результат выполнения команды предоставляется в виде списка из двух столбцов, где имя переменной помещено слева, а соответствующее ей значение находится справа. |  |
| fc | Редактирование файла хронологии (содержит протокол команд). Передаваемые параметры используются для выбора диапазона команд из файла хронологии. |  |
| fg | Выполнение приостановленного процесса в основном режиме (не фоновом) |  |
| file | Тестирование файловой системы, проверка магического (magic) номера, а также проверка языка программирования. |  |
| find | Поиск файлов и каталогов по разрешениям, группе, типу, правам, имени, дате, размеру и т.д., а также выполнение определенных действий с результатами поиска.  Синтаксис: find папка параметры критерий шаблон действие.  -type f – искать только файлы,  -type d – искать только папки.  -maxdepth число – глубина поиска по вложенным папкам.  -not – найти файлы, не соответствующие шаблону.  find ./’Рабочий стол’ – найти все файлы на рабочем столе.  find . -name “\*.png” – найти файлы с разрешением png.  find . -maxdepth 1 -type f -name “.\*” – найти скрытые файлы в директории.  find . -cmin 60 – найти файлы, изменённые в течение часа.  find / -size +50M -size -100M – найти файлы размером от 50 до 100 МБ. |  |
| grep | Поиск текста по шаблону: строке или регулярному выражению.  По умолчанию принимает стандартный ввод, но можно настроить и поиск в файлах, записав не путь для поиска, а путь к файлу.  -w – поиск слова. -c – количество вхождений строки регулярного выражения. |  |
| groff | Внешний интерфейс программы форматирования groff-документов. |  |
| gzip | Наиболее часто используемая утилита для создания архивов.  -c для создания;  -d для распаковки,;  -f для принудительной операции;  -r для рекурсии;  -l для подробного вывода;  от -1 до -9 – уровни сжатия по возрастанию. |  |
| halt | Предложение ядру завершить работу системы. |  |
| help | Выводит подсказку по синтаксису некоторых наиболее важных команд среды программирования с помощью команд оболочки. |  |
| kill | Посылает определённый сигнал заданному процессу. Если никакой сигнал не определён, посылается сигнал TERM.  kill <опция> <pid>  kill -9 1438 |  |
| less | Просмотр длинных текстов, не вмещающихся в экран терминала.  less filename откроет в терминале файл, который можно прокручивать и читать. Это более новый аналог команды more, которая теперь отпала. |  |
| login | Команда входа в систему login используется для регистрации. Может также применяться в любое время для переключения с учетных данных одного пользователя на другого. |  |
| logout | Используется для выхода из системы текущего пользователя. Если в системе зарегистрирован единственный пользователь, тогда работа системы завершается. |  |
| lpc | Управление работой устройства построчной печати. Её можно применять, чтобы отключать или запускать устройство печати или очередь спула печати, перестраивать порядок заданий в спуле, выяснять состояние принтеров, очередей спула и демонов печати. |  |
| lpd | Демон устройства построчной печати обычно вызывается во время начальной загрузки из файла rс. Он делает один проход по файлу /etc/printcap, чтобы найти существующие принтеры и напечатать любые файлы, оставшиеся после аварийного завершения. |  |
| lpq | Просмотр очереди печати.  -P для очереди конкретного принтера.  -l для вывода списков всей очереди печати на все принтеры |  |
| lpr | Печать текстового файла.  lpr filename.  -P для ввода имени принтера. Можно поместить несколько файлов в очередь на печать.  lpr \*.txt для печати всех текстовых документов. |  |
| ls | Содержимое каталога  ls -l для более удобного вывода;  ls -a для вывода скрытых файлов. |  |
| make | Автоматическое определение частей большой программы, которые нужно перекопировать, и затем задает необходимые для этого команды. |  |
| man | Отображение руководства man pages. |  |
| mesg | Утилита запускается пользователем, чтобы управлять приемом сообщений от других пользователей на его терминал. Если доступ по записи открыт, программы типа talk и write имеют разрешение отображать сообщения на терминале. |  |
| mkdir | Создать каталог. Можно создать папку по другому пути, если прописать абсолютный путь.  mkdir {dir1, dir2} создаст несколько каталогов.  mkdir -p позволяет создать вложенную строктуру:  mkdir -p dir1/dir2/{dir3,dir4} |  |
| mkfs | Используется для построения файловой системы Linux на устройстве, обычно в разделе жесткого диска. |  |
| mkswap | Команда устанавливает на устройстве (как правило, раздел диска) область своппинга Linux**.** |  |
| more | Является фильтром для листания текста поэкранно. Это команда "двигается" только вниз по тексту, в противоположность команде less**,** которая может "перемещаться" вверх и вниз. |  |
| mount | Подсоединяет файловую систему, определяемую с помощью специального файла (часто является именем устройства), к задаваемому в виде параметра каталогу. Если выполняется без параметров, то выводит список всех монтированных в настоящее время файловых систем. |  |
| mv | Перемещение или, при указании той же директории, переименование файлов. |  |
| netstat | Состояние сетевых соединений на протоколах TCP, UDP, а также низкоуровневых сетевых соединений или через сокеты UNIX.  -r – получение информации о таблице маршрутизации. |  |
| passwd | Ввод старого пароля и ввод нового пароля пользователя. Суперпользователь также может заменить пароль любого пользователя. |  |
| ps | Вывод запущенных процессов и их PID (TTY – терминал, из которого запущен процесс, TIME – общее время, CMD – команда его запуска).  -aux выведет полную информацию о всех процессах.  -A – все процессы, кроме лидеров групп.  -d – все процессы, включая фоновые и лидеров групп. |  |
| pwd | Выводит имя текущего каталога, полезно при bash-программировании |  |
| rm | Удаляет файлы и папки, причем не в корзину, а безвозвратно.  -i запросит дополнительное подтверждение.  rm file\* удалит все файлы, начинающиеся со слова file.  -R удаляет директории со всеми вложенными файлами и папками. |  |
| rmdir | Удалить каталог. |  |
| set | Временное изменение переменной среды. |  |
| shutdown | Закрытие системы. |  |
| su | Запустить программу от имени другого пользователя. Похожая команда sudo, но su переключает на другого юзера, а sudo только выполняет команду от его имени. |  |
| swapoff | Остановка своппинга. |  |
| swapon | Команда устанавливает область своппинга (подкачки) в виде файла или блочного устройства, указывая путь. Обычно выполняется при запуске системы |  |
| tail | Выводит несколько последних строк файла, аналогична head.  -f позволяет следить за файлом: при изменении, например, файла логов, после команды head -f filename нужный файл будет сразу выводиться в консоль. |  |
| talk | Переписка с другим пользователем в терминале в реальном времени. |  |
| tar | Стандартная утилита сжатия с множеством опций, например:  -c – создать архив;  -j – с помощью Bzip;  -z – с помощью Gzip;  -A – добавить файл к архиву;  -t – показать содержимое архива;  -x – извлечь файлы из архива;  Для сжатия используется -cvf;  Для распаковки используется -xvf. |  |
| umount | Применяется для выполнения демонтирования файловых систем с места их расположения. |  |
| unalias | Удаление созданных алиасов: unalias aliasname |  |
| unzip | Распаковать zip-архив. |  |
| wall | Отображает содержимое стандартного ввода на терминалах всех зарегистрированных в настоящее время пользователей. |  |
| who | Выводит имя пользователя, находящегося в консоли в данный момент. |  |
| xhost + | Позволяет отображать в системе xterms. Вероятно, наиболее общая причина, по которой дистанционный терминал не может быть открыт – команда xhost + не была выполнена. Чтобы отключить данную возможность, используется команда xhost - |  |
| xmkmf | Используется, чтобы создавать Imakefiles для исходных модулей системы X. Она фактически выполняет команду intake с набором аргументов. |  |
| xset | Устанавливает некоторые опции в сеансе X Window. Пользователь может применять эти опции, чтобы устанавливать свой звонок, скорость мыши и другие параметры. |  |
| zip | Кроссплатформерная утилита для создания zip-архивов. С опцией -e можно настроить шифрование |  |

# Контрольные вопросы

*1. Объясните, что произойдет в случае ввода команды $ date \@:*

date: invalid date ‘@’.

Данный результат вывелся потому, что \ – это символ перехода на новую строку в терминале, и команда выглядит как «date @», что вызывает ошибку.

*2. Объясните приведенный ниже диалог. Предполагается, что для стирания также используется символ #.*

$date

Mon Sep 26 12:39:56 EET 2006

$# date

Mon Sep 26 12:39:56 EET 2006

$\#date

$\\#date

#date : not found

date – команда вывода текущей даты.

# – начало комментария, поэтому #date не выведет ничего.

\ – символ экранирования, поэтому # считывается как строковый литерал, и в оболочку подается команда «#date», что приводит к выводу ошибки «#date: command not found».

В четвертом примере также экранируется всё, что стоит после первого обратного слеша, поэтому мы получаем ошибку «\#date: command not found».

*3. Попробуйте выполнить команду $ls /usr/games, а затем что-либо из предложенного ею. Вы довольны? Остановите выполнение запущенной программы, одним из известных Вам способов. Объясните, что выполнила система в ответ на ваши команды.*

При выводе команды ls /usr/games происходит вывод содержимого каталога /usr/games/, в котором находятся системные игры. При вводе названия любой из них запускается сама игра в новом окне. Да, я доволен.

*4. В чем состоит разница между следующими командами:*

$ls junk вывод файла или директории «junk»

$ls / вывод корневой директории системы

$ls вывод всех файлов и папок в директории

$ls \* вывод также вложенных файлов и папок

$ls ’\*’ вывод файла или директории «\*»

$echo junk вывод на консоль слова «junk»

$echo \ переход на новую строку в терминале

$echo вывод пустой строки

$echo \* вывод списка файлов директории

$echo ‘\*’ вывод строкового литерала «\*»

$‘\*’ ошибка: «\*: command not found»

*5. Объясните, почему команда $ ls > ls.out включает ls.out в список имен []?*

Данная комбинация команд перенаправит вывод команды ls в файл ls.out. Пустые квадратные скобки [] могут появиться в файле ls.out, если он был ранее создан и уже существует.

*6. Объясните результат выполнения команды $ wc temp > temp.*

Сначала выполняется команда wc temp, которая записывает информацию о файле temp – количество в нем строк, слов и размер в байтах. Так как файл пустой, то выведется 0 0 0 temp. Данный вывод перенаправляется в сам файл temp и записывается туда. В итоге получаем в файле temp запись 0 0 0 temp.

*7. Что произойдет, если вы ошибетесь в имени команды, задав   
$ who > temp?*

Если ошибиться в имени команды, то произойдет сообщение об ошибке.

*8. Объясните разницу между командами $ who | sort и   
$ who > sort.*

Команда who вывод пользователей, которые в данный момент находятся в системе.

Команда who | sort сначала выводит список активных пользователей, а затем передает его в команду sort для их сортировки.

Команда who > sort перенаправит вывод команды who в файл sort.