Министерство образования Республики Беларусь

Учреждение образование

«Белорусский государственный технологический университет»

Кафедра программной инженерии

**Отчет к лабораторной работе**:

«Командная оболочка ОС UNIX Shell»

Выполнил:

Студент 4 курса 5 группы ФИТ

Почиковской Юлии

Проверил:

Копыток Д. В.

Минск 2022

**Теоретические сведения**

1. **Командная оболочка ОС UNIX**

Интерфейсом для ОС UNIX служит shell (*оболочка),* являющаяся программным уровнем, который обеспечивает среду для ввода команд и параметров для получения желаемого результата. Для удовлетворения различных потребностей в UNIX существуют различные оболочки. Например, оболочки Bourne (sh), Bourne Again (bash), Korn (ksh) и С (csh). У каждой из них — свои свойства и способы взаимодействия с UNIX. В настоящей практической работе рассматриваются следующие темы.

Способ взаимодействия пользователя с командной оболочкой в ОС UNIX, свойства командного процессора shell, управление средой окружения shell

**Взаимодействие ядра и оболочки**

Когда система UNIX переводится в интерактивный режим, ядро системы загружается в основную память компьютера, где остается до выключения компьютера. Во время процесса загрузки программа **init** выполняется в качестве фоновой задачи и продолжает выполняться до выключения. Эта программа сканирует файл **/etc/inittab,** в котором перечисляются порты, имеющие терминалы, и их характеристики. Когда эта программа активна и открытый терминал найден, **init** вызывает программу **getty,** которая вызывает на монитор терминала приглашение **login:.** Когда этот процесс вызван и выполняется, пользователь может начинать взаимодействовать с системой.

**Вызовы оболочки UNIX при входе в систему**

Во время входа в систему после ввода имени пользователя программа **getty** выводит на монитор приглашение **password:.** После ввода пароля программа getty вызывает программу **login,** которая просматривает файл **/etc/passwd** на предмет соответствующих записей. Если соответствие установлено, **login** осуществляет переход к начальному каталогу пользователя и передает управление программе начального запуска сеанса; и имя пользователя, и пароль определяются записью в файле /etc/passwd. Хотя это может зависеть от программы приложения, например, после введения пароля может загрузится конкретное приложение, управляемое меню, обычно программа начального запуска сеанса является командным интерпретатором, например таким как оболочка Bourne **/bin/sh**.

Затем программа оболочки считывает файлы **/etc/profile** и **.profile** (из рабочего каталога пользователя)**,** которые устанавливают параметры, характерные для всей системы и для конкретного пользователя. После этого оболочка выводит приглашение на ввод команд, подобное $.

При выходе из оболочки ядро возвращает управление программе **init,** которая перезапускает процесс входа в систему. Прерывание может выполняться одним из двух способов:

- посредством команды **exit;**

**-** или когда ядро вызывает команду **kill** применительно к процессу оболочки.

При прерывании ядро освобождает все ресурсы, использовавшиеся пользователем и связанной с ним программой оболочки.

**Оболочка и дочерние процессы**

В системе UNIX существует много уровней программ, начиная с ядра и заканчивая конкретным приложением или командой. Взаимосвязь между этими уровнями показана на рис. 1.

После завершения процесса входа в систему уровень программы оболочки (уровень командного интерпретатора) непосредственно взаимодействует с ядром, как показано на рис. 1. 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. Для ознакомления с примером такого процесса введите следующую команду:

$ps

В результате выводится перечень выполняющихся в данный момент команд: программа оболочки и программа **ps.** Теперь введите следующее:

$sleep 10 & $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.

**Автоматическое выполнение оболочки**

Некоторые ресурсы UNIX, например **сгоn** (демон выполнения заданий по расписанию)**,** могут запускать программу оболочки без вмешательства пользователя. При использовании этого свойства пользователю нужно в первой строке файла с кодом программы указать оболочку, которую необходимо запустить, например, так:

#!/bin/sh

Эта строка определяет оболочку Bourne.

Необходимо также переназначить любой вывод, поскольку ни один терминал не связан с автоматическим выполнением.

**Функции и свойства оболочки**

Какую бы оболочку не выбрал пользователь, все они имеют одно и то же назначение — предоставить пользователю интерфейс к UNIX. С этой целью все оболочки обладают некоторыми общими характеристиками:

* Интерпретация командной строки.
* Зарезервированные ключевые слова (имена команд и переменных окружения).
* Метасимволы оболочки (шаблоны).
* Доступ к программам и их исполнение.
* Обработка файлов: переназначение ввода/вывода и программные каналы.
* Поддержка переменных среды окружения командной оболочки.
* Управлением средой окружения командной оболочки.
* Программирование на основе команд оболочки.

**2. Интерпретация командной строки**

При входе в систему запускается специальная версия shell, называемая *интерактивной оболочкой,* отображается приглашение оболочки, обычно в форме знака доллара ($), процента (%) или фунта (#). При вводе строки в ответ на это приглашение оболочка пытается интерпретировать ее. Ввод в ответ на приглашение оболочки иногда называют *командной строкой.* Основной формат командной строки выглядит следующим образом:

*команда аргументы*

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

**Аргументы**передаются исполняемому модулю. Большинство программ утилит UNIX предполагают, что **аргументы**будут иметь следующую форму:

*опции имена\_файлов*

Например, в командной строке

$ls -l filel file2

- имеется три аргумента команды ls; первый является опцией, а последние два — именами 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 **echo** с единственным аргументом, который в данном случае является строкой символов, содержащей пробелы.

**Зарезервированные слова**

Все версии оболочек имеют слова, обладающие специальным значением. При программировании средствами оболочек такие слова, как **do, done, for** и **while** обеспечивают управление циклом, a **if, then, else if -** условиями. Каждая версия оболочки обладает различными зарезервированными словами, присущими именно ей.

**Метасимволы оболочки (шаблоны)**

Шаблоны (таблица 1) используются для указания в команде нескольких имен обрабатываемых файлов.

|  |  |
| --- | --- |
| Все версии оболочек поддерживают метасимволы, которые дают возможность указывать имена файлов. Шаблоны приводятся ниже:*Шаблоны* | *Описание* |
| **?** | Соответствие любому символу |
| **\*** | Соответствие любой части символьной строки |
| **[a-z]** | Соответствие диапазону приведенных символов |
| **[0-9]** | Соответствие диапазону приведенных цифр |

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

$ls t\*

Эта команда выводит перечень всех файлов, 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.

**Команды запуска программ**

При вводе команды оболочка считывает переменную среды $PATH содержащую список каталогов с файлами программ. Оболочка просматривает этот набор каталогов, отыскивая файл программы, соответствующий команде. Затем оболочка передает ядру действительное имя файла.

**Обработка файлов - переправления ввода/вывода и прграммные каналы**

Большинство команд UNIX принимают ввод с клавиатуры терминала и отправляют вывод на его дисплей, если иное не указано аргументами. Для переназначения вывода в файл используется символ '>'. Например, команда

$ls > myfiles

выводит список файлов текущего каталога и помещает его в файл, названный myfiles. Аналогично можно переназначить ввод с помощью символа <. Например, команда

$wc -l < myfiles

осуществляет ввод в команду wc из файла myfiles. Хотя такой же результат можно было бы получить, использовав имя файла в качестве аргумента, необходимость в переназначении ввода становится более очевидной при программировании на языке оболочки shell.

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. Команда канала — одно из наиболее используемых инструментальных средств при создании конструкций команд.

**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.

**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 ($). Рассмотрите следующие примеры:

$ echo $LOOKUP /usr/mydir

$ echo $LOOKUP LOOKUP

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

$LOOKUP=/usr/mydir

$export LOOKUP

Примечание. Присвоение значений переменным в оболочке С отличается от выполнения этого в оболочках Bourn и Корн, Для присвоения переменной в оболочке С используется команда set:

% set LOOKUP = /usr/mydir

Замечание. Обратите внимание, что знак равенства с обеих сторон выделяется пробелами.

Подобно подстановке имени файла подстановка имени переменной происходит до вызова программы. Во втором примере знак доллара ($) опущен. Следовательно, оболочка просто передает строку команде **echo** в качестве аргумента. При подстановке имени переменной значение переменной замещает имя переменной.

Например, в строке

$ ls $LOOKUP/filename

программа ls вызывается с единственным аргументом **/usr/mydir/filename**.

**Запуск оболочки — управление средой**

Когда пользователь начинает сеанс работы с UNIX то запускается оболочка (если иное не определено в учетной записи пользователя), оболочка создает для пользователя специфичную рабочую среду, параметры которой сохраняются в файле профиля учетной записи пользователя (**~имя\_пользователя/.profile**).

**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 **$TERM** и путь по умолчанию для исполняемых файлов — в переменной **$РАТН.** Посмотрите следующие примеры:

$echo $term

$echo $PATH

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

Примечание. Оболочка С присваивает значения переменным среды посредством команды setenv:

%setenv TERM = vt100

**Файлы начального запуска оболочки**

Файл **.profile** — локальный файл начального запуска для оболочки Bourne. Оболочка Коrn 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.

**Опции начального запуска оболочки**

При вызове оболочки из **/etc/passwd** или из командной строки, несколько опций можно устанавливать в качестве аргументов программы оболочки. Например, оболочка Bourne имеет опцию -х, которая перед исполнением отображает команды и их аргументы. Это удобно при отладке программы оболочки.

**Практическое задание**

50 наиболее применяемых команд:

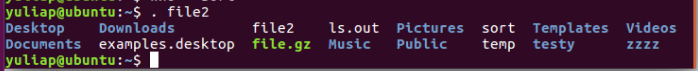


Рисунок 1 – Команда .



Рисунок 2 – Команда &

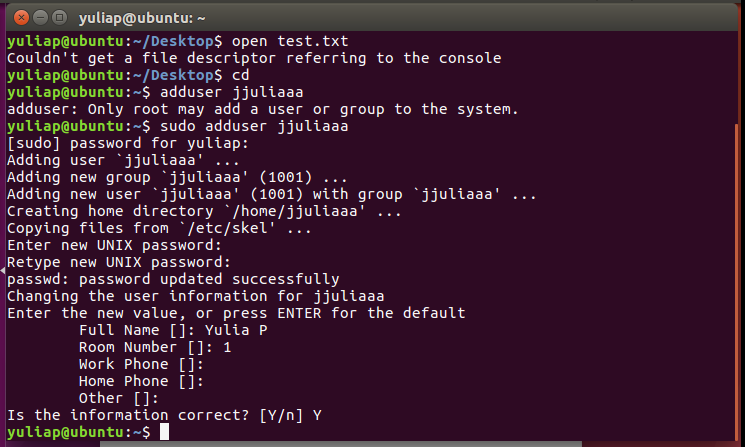


Рисунок 3 – Команда adduser

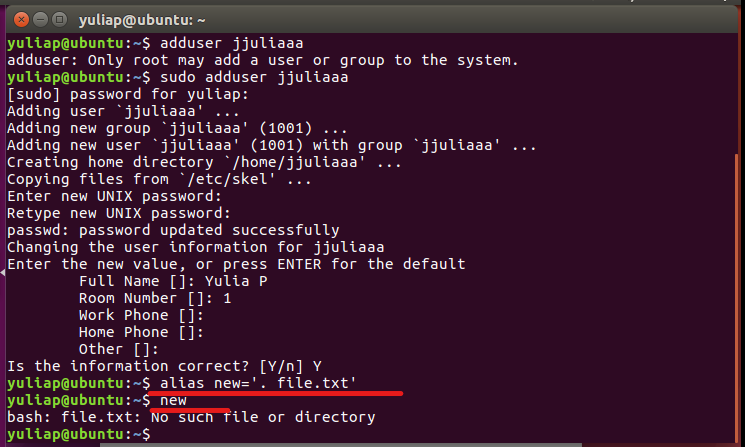


Рисунок 4 – Команда alias

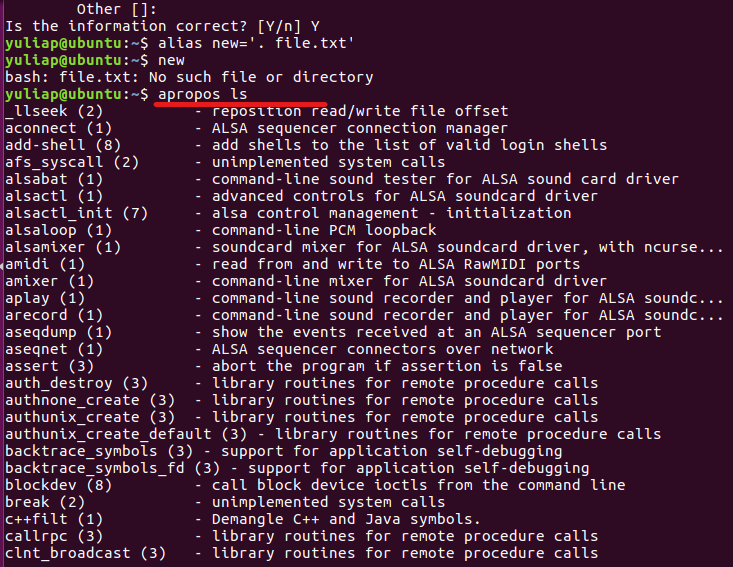


Рисунок 5 – Команда apropos

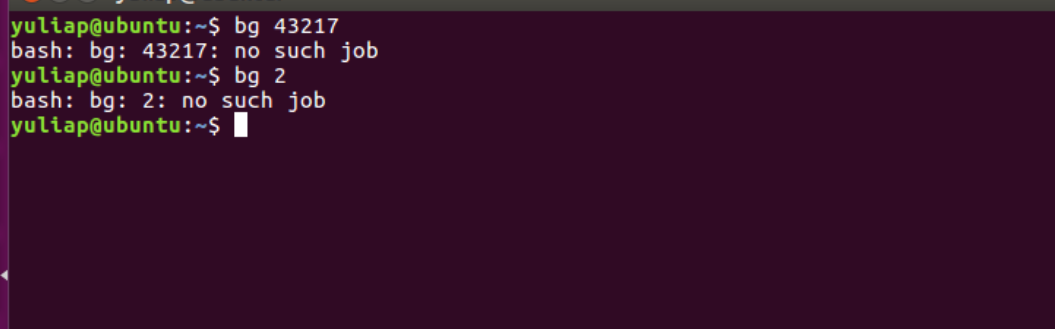


Рисунок 6– Команда br



Рисунок 7 – Команда cat

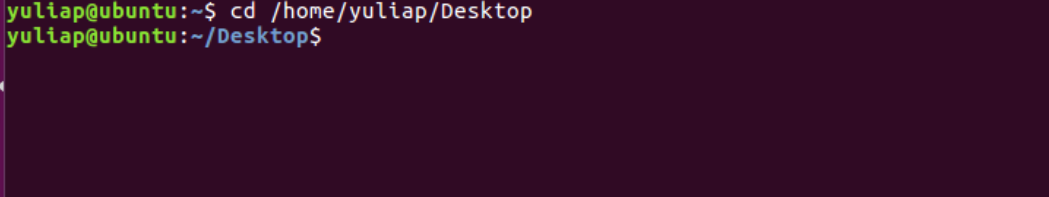


Рисунок 8 – Команда cd

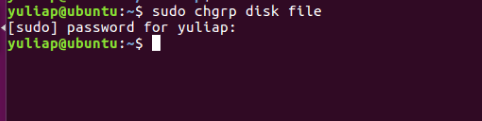


Рисунок 9 – Команда chgrp

Например, чтобы добавить полномочия на чтение и выполнение владельцу и группе файла testl, пользователь должен ввести команду

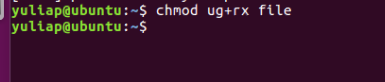


Рисунок 10 – Команда chmod

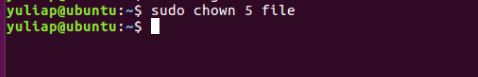


Рисунок 11 – Команда chown

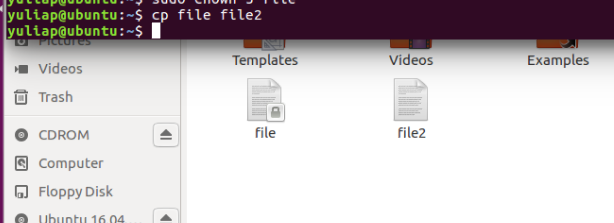


Рисунок 12 – Команда cp (copy)

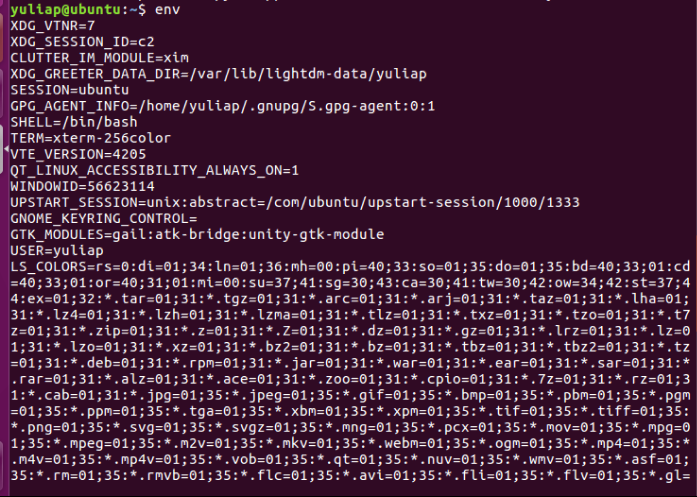


Рисунок 13 – Команда env



Рисунок 14 – Команда fc

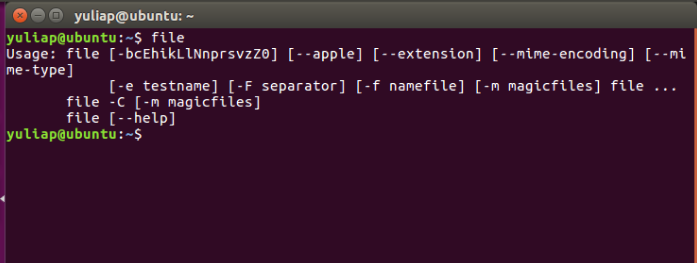


Рисунок 15 – Команда file

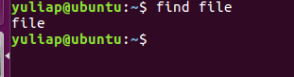


Рисунок 16 – Команда find

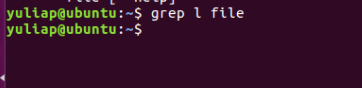


Рисунок 17– Команда grep

Это внешний интерфейс программы форматирования groff-документов. По умолчанию эта программа вызывает программу troff.

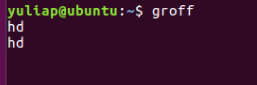


Рисунок 18 – Команда groff

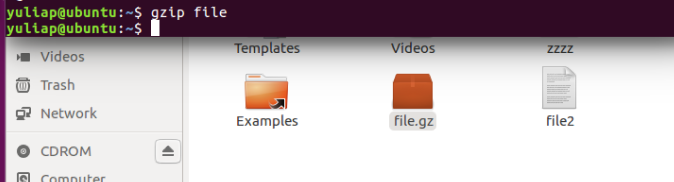


Рисунок 19 – Команда gzip

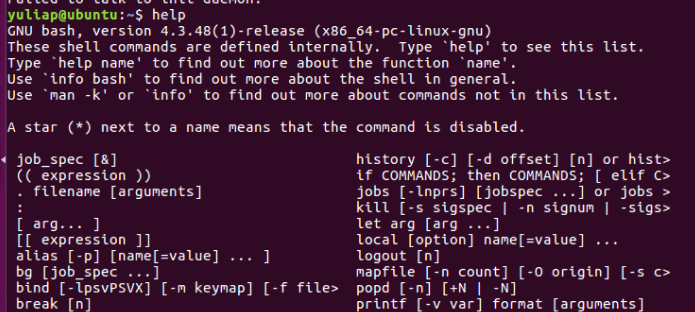


Рисунок 20 – Команда help



Рисунок 21 – Команда hostname



Рисунок 22 – Команда kill

Работает подобно программе more, но допускает перемещение в файле как в прямом, так и в обратном направлениях. Она также не должна читать весь входной файл перед стартом, поэтому для больших входных файлов программа запускается быстрее, чем текстовый редактор, такой, как vi.

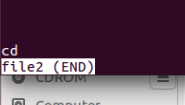


Рисунок 23 – Команда less

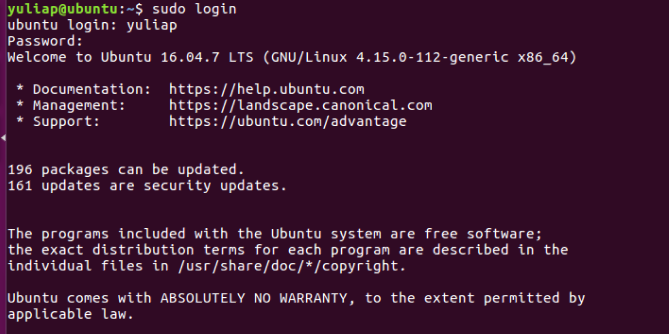


Рисунок 24 – Команда login



Рисунок 25 – Команда logout

Используется администратором системы для управления 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. Команда может быть задана для любого из принтеров, сконфигурированных в файле /etc/printcap.



Рисунок 26 – Команда lpc

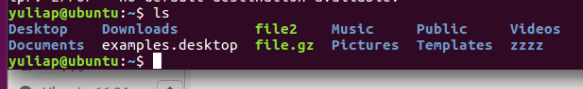


Рисунок 27 – Команда ls

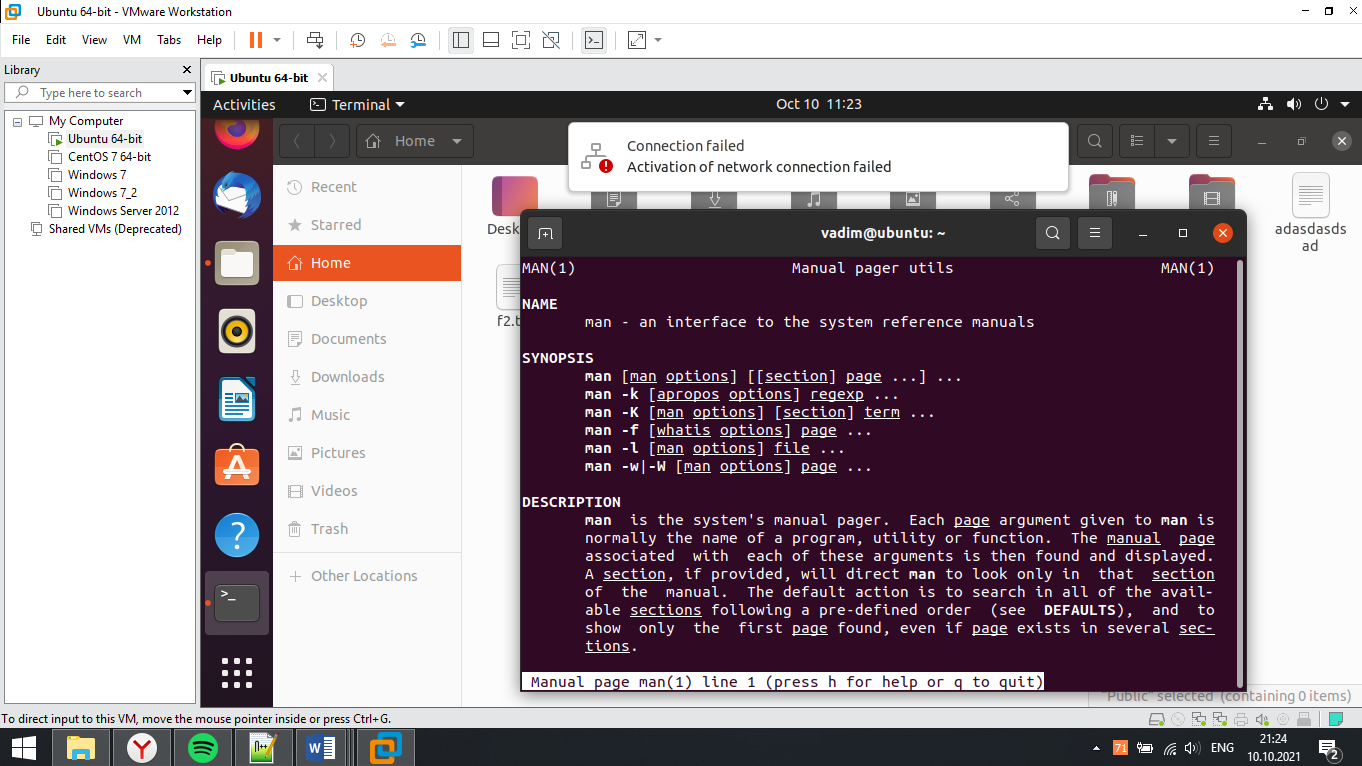


Рисунок 28 – Команда man (для man)



Рисунок 29 – Команда mesg



Рисунок 30 – Команда mkdir

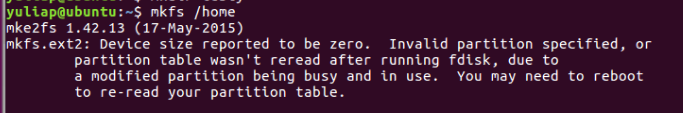


Рисунок 31 – Команда mkfs

Команда устанавливает на устройстве (как правило, раздел диска) область своппинга **Linux.** Устройство обычно задается в такой форме:



Рисунок 32 – Команда mkswap



Рисунок 33 – Команда more

Подсоединяет файловую систему, определяемую с помощью специального файла (часто является именем устройства), к задаваемому в виде параметра каталогу. Только суперпользователь может монтировать файлы. Если команда **mount** выполняется без параметров, она выводит список всех монтированных в настоящее время файловых систем.

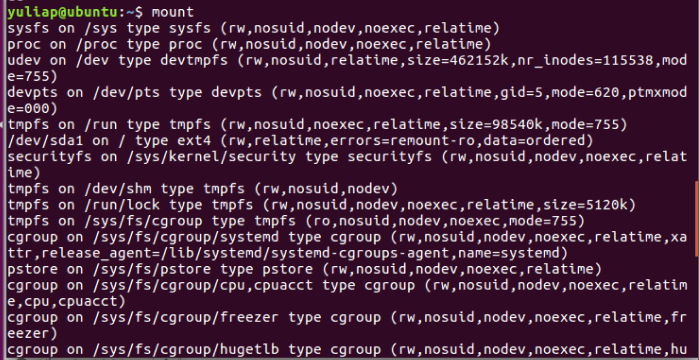


Рисунок 34 – Команда mount



Рисунок 35 – Команда mv

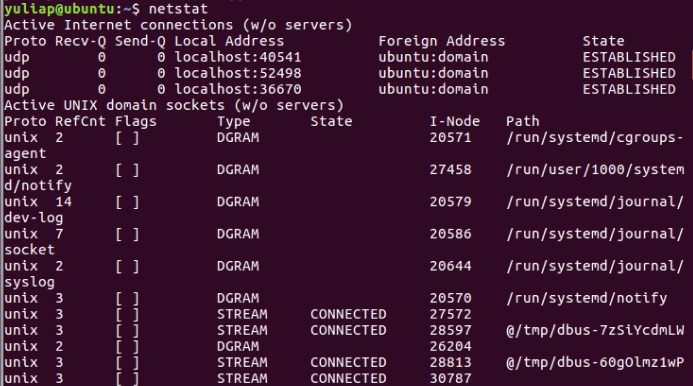


Рисунок 36 – Команда netstat

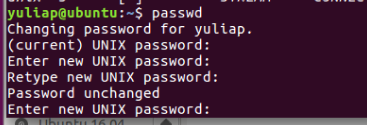


Рисунок 37 – Команда passwd

Предоставляет отчет о текущих **процессах.**

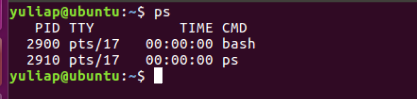


Рисунок 38 – Команда ps

Выводит имя текущего каталога

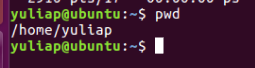


Рисунок 39 – Команда pwd



Рисунок 40 – Команда rm



Рисунок 41 – Команда set

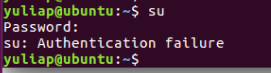


Рисунок 42 – Команда su

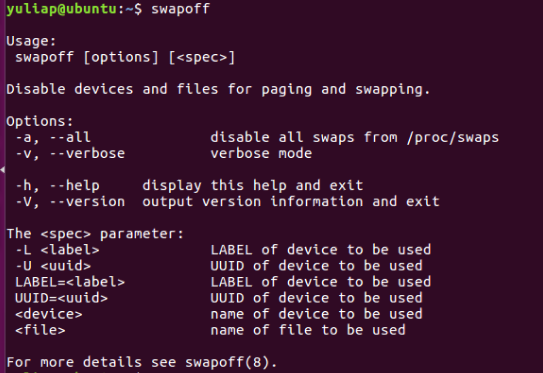


Рисунок 43 – Команда swapoff

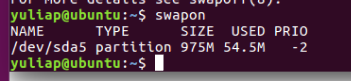


Рисунок 44 – Команда swapon

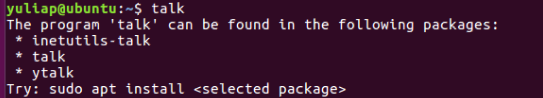


Рисунок 45 – Команда talk

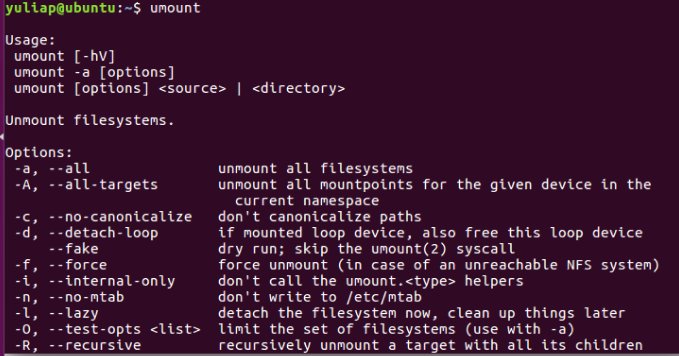


Рисунок 46 – Команда umount



Рисунок 47 – Команда unzip



Рисунок 48 – Командаn wall



Рисунок 49 – Команда who

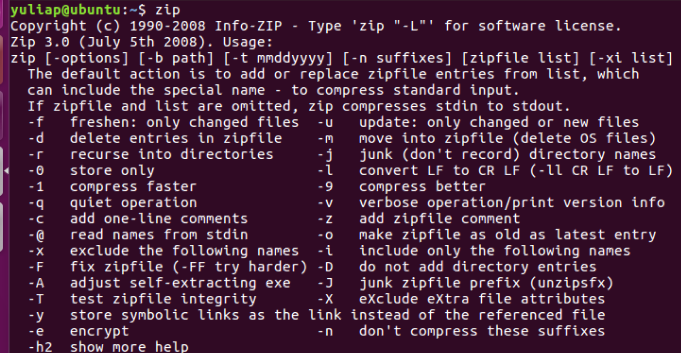


Рисунок 50 – Команда zip

Упражнения:

$date \@

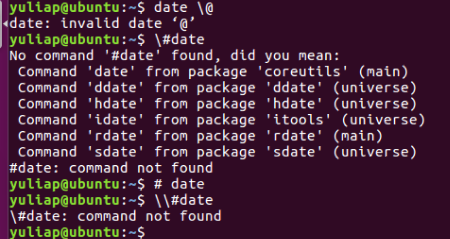


Рисунок 1 – Команда date

$ls junk

$ls /

$ls

$ls \*

$ls’\*’

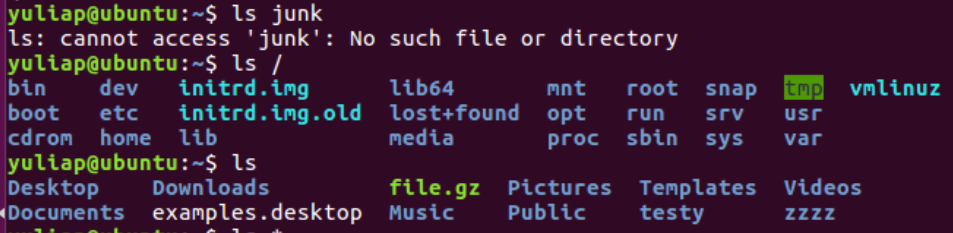


Рисунок 2 – Запуск игр с помощью команды ls

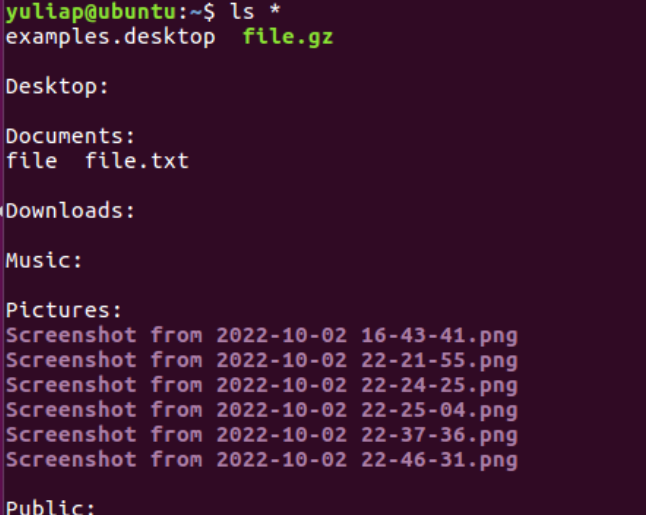


Рисунок 2 – Запуск команды ls

$echo junk

$echo /

$echo

$echo \*

$echo ‘\*’

$‘\*’

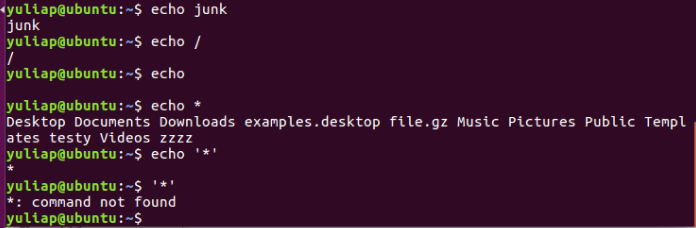


Рисунок 3 – Запуск команды echo



Рисунок 5 – Работа с командой ls

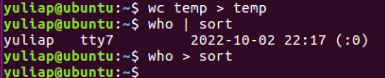


Рисунок 6- Команды wc и who