

# **Отчёт по лабораторной работе №5**

*дисциплина:Операционные системы*

Бондаренко Елизавета Валентиновна

# Содержание

1	Цель работы	5
2	Последовательности выполнения работы	6
3	Выполнение лабораторной работы	8
4	Контрольные вопросы	17
5	Выводы	24
	Список литературы	25

## Список иллюстраций

3.1	Создание текстового файла и копирование . . . . .	8
3.2	Копирование каталогов в текущем каталоге. Копируем каталог monthly в другой каталог . . . . .	8
3.3	Переименование файлов в текущем каталоге и перемещение . . .	8
3.4	Копируем файл /usr/include/sys/io.h в домашний каталог и называ- ем его equipment . . . . .	9
3.5	В домашнем каталоге создаем директорию ~/ski.plases . . . . .	9
3.6	Перемещаем файл equipment в каталог ~/ski.plases . . . . .	9
3.7	Переименование . . . . .	9
3.8	Создание и копирование . . . . .	9
3.9	Создание и перемещение . . . . .	10
3.10	Создание и перемещение . . . . .	10
3.11	Опции команды chmod . . . . .	11
3.12	содержимое файла /etc/password . . . . .	11
3.13	Копирование . . . . .	11
3.14	Перемещение . . . . .	12
3.15	Копирование . . . . .	12
3.16	Перемещение и переименование . . . . .	12
3.17	Лишение и получение прав владельцаам . . . . .	13
3.18	mount . . . . .	14
3.19	fsck . . . . .	14
3.20	kill . . . . .	15
3.21	mkfs . . . . .	16

## Список таблиц

# 1 Цель работы

В ходе выполнения лабораторной работы я должна ознакомиться с файловой системой Linux, её структурой, именами и содержанием каталогов. А также приобрести практические навыки по применению команд для работы с файлами и каталогами, по управлению процессами (и работами), по проверке использования диска и обслуживанию файловой системы.

## 2 Последовательности выполнения работы

1. Выполняем все примеры, приведённые в первой части описания лабораторной работы.
2. Выполняем следующие действия, зафиксировав в отчёте по лабораторной работе используемые при этом команды и результаты их выполнения:
  - 2.1. Копируем файл `/usr/include/sys/io.h` в домашний каталог и называем его `equipment`. Если файла `io.h` нет, то используем любой другой файл в каталоге `/usr/include/sys/` вместо него.
  - 2.2. В домашнем каталоге создаем директорию `~/ski.plases`.
  - 2.3. Перемещаем файл `equipment` в каталог `~/ski.plases`.
  - 2.4. Переименуем файл `~/ski.plases/equipment` в `~/ski.plases/equiplist`.
  - 2.5. Создаем в домашнем каталоге файл `abc1` и копируем его в каталог `~/ski.plases`, называем его `equiplist2`.
  - 2.6. Создаем каталог с именем `equipment` в каталоге `~/ski.plases`.
  - 2.7. Перемещаем файлы `~/ski.plases/equiplist` и `equiplist2` в каталог `~/ski.plases/equipment`.
  - 2.8. Создаем и перемещаем каталог `~/newdir` в каталог `~/ski.plases` и называем его `plans`.
3. Определяем опции команды `chmod`, необходимые для того, чтобы присвоить перечисленным ниже файлам выделенные права доступа, считая, что в начале таких прав нет. При необходимости создаем нужные файлы.
4. Проведем приведённые ниже упражнения, записывая в отчёт по лабораторной работе используемые при этом команды:
  - 4.1. Просмотрим содержимое

- файла `/etc/passwd`. 4.2. Скопируем файл `~/feathers` в файл `~/file.old`. 4.3. Переместим файл `~/file.old` в каталог `~/play`. 4.4. Скопируем каталог `~/play` в каталог `~/fun`. 4.5. Переместим каталог `~/fun` в каталог `~/play` и назовите его `games`. 4.6. Лишим владельца файла `~/feathers` права на чтение. 4.7. Что произойдёт, если мы попытаемся просмотреть файл `~/feathers` командой `cat`? 4.8. Что произойдёт, если мы попытаемся скопировать файл `~/feathers`? 4.9. Даем владельцу файла `~/feathers` право на чтение. 4.10. Лишите владельца каталога `~/play` права на выполнение. 4.11. Переходим в каталог `~/play`. Что произошло? 4.12. Даем владельцу каталога `~/play` право на выполнение.
5. Прочитаем ман по командам `mount`, `fsck`, `mkfs`, `kill` и кратко их охарактеризуем.

### 3 Выполнение лабораторной работы

1. Выполняем все примеры, приведённые в первой части описания лабораторной работы. Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию. Рисунки 3.1; 3.2; 3.3.

```
evbondarenko@dk3n60 ~ $ cd
evbondarenko@dk3n60 ~ $ touch abc1
evbondarenko@dk3n60 ~ $ cp abc1 april
evbondarenko@dk3n60 ~ $ cp abc1 may
evbondarenko@dk3n60 ~ $ mkdir monthly
evbondarenko@dk3n60 ~ $ cp april may monthly
evbondarenko@dk3n60 ~ $ cp monthly/may monthly/june
evbondarenko@dk3n60 ~ $ ls monthly
april  june  may
```

Рис. 3.1: Создание текстового файла и копирование

```
evbondarenko@dk3n60 ~ $ mkdir monthly.00
evbondarenko@dk3n60 ~ $ cp -r monthly monthly.00
evbondarenko@dk3n60 ~ $ cp -r monthly.00 /tmp
```

Рис. 3.2: Копирование каталогов в текущем каталоге. Копируем каталог monthly в другой каталог

```
evbondarenko@dk3n60 ~ $ cd
evbondarenko@dk3n60 ~ $ mv april july
evbondarenko@dk3n60 ~ $ mv july monthly.00
evbondarenko@dk3n60 ~ $ ls monthly.00
july  monthly
evbondarenko@dk3n60 ~ $ mv monthly.00 monthly.01
evbondarenko@dk3n60 ~ $ mkdir reports
evbondarenko@dk3n60 ~ $ mv monthly.01 reports
evbondarenko@dk3n60 ~ $ mv reports/monthly.01 reports/monthly
```

Рис. 3.3: Переименование файлов в текущем каталоге и перемещение



2. Копируем файл /usr/include/sys/io.h в домашний каталог и называем его equipment. Если файла io.h нет, то используем любой другой файл в каталоге /usr/include/sys/ вместо него.(рис.3.4).

```
evbondarenko@dk3n60 ~ $ cp /usr/include/sys/io.h equipment
evbondarenko@dk3n60 ~ $ ls
-          GNUstep    public      Документы   'Рабочий стол'
1          may        public_html Загрузки     Шаблоны
abc1       monthly   reports     Изображения
Architecture_PC newdir    tmp         Музыка
equipment  os-intro  Видео       Общедоступные
```

Рис. 3.4: Копируем файл /usr/include/sys/io.h в домашний каталог и называем его equipment

В домашнем каталоге создаем директорию ~/ski.plases (рис.3.5).

```
evbondarenko@dk3n60 ~ $ mkdir ski.plases
```

Рис. 3.5: В домашнем каталоге создаем директорию ~/ski.plases

Перемещаем файл equipment в каталог ~/ski.plases (рис.3.6).

```
evbondarenko@dk3n60 ~ $ mv equipment ski.plases
```

Рис. 3.6: Перемещаем файл equipment в каталог ~/ski.plases

Переименуем файл ~/ski.plases/equipment в ~/ski.plases/equiplist(рис.3.7).

```
evbondarenko@dk3n60 ~ $ mv ski.plases/equipment ski.plases/equiplist
```

Рис. 3.7: Переименование

Создаем в домашнем каталоге файл abc1 и копируем его в каталог ~/ski.plases, называем его equiplist2.(рис.3.8).

```
evbondarenko@dk3n60 ~ $ touch abc1
evbondarenko@dk3n60 ~ $ cp abc1 ski.plases
evbondarenko@dk3n60 ~ $ mv ski.plases/abc1 ski.plases/equiplist2
```

Рис. 3.8: Создание и копирование

Создаем каталог с именем equipment в каталоге ~/ski.plases и перемещаем его(рис.3.9).

```
evbondarenko@dk3n60 ~ $ mv equipment ski.plases
evbondarenko@dk3n60 ~ $ mv ski.plases/equipment ski.plases/equiplist
```

Рис. 3.9: Создание и перемещение

Создаем и перемещаем каталог ~/newdir в каталог ~/ski.plases и называем его plans(рис.3.10).

```
evbondarenko@dk3n60 ~ $ mv newdir plans
evbondarenko@dk3n60 ~ $ mv plans ski.plases
evbondarenko@dk3n60 ~ $ ls
```

abc1	monthly	reports	Документы	Общедоступные
Architecture_PC	os-intro	ski.plases	Загрузки	'Рабочий стол'
GNUstep	public	tmp	Изображения	Шаблоны
may	public_html	Видео	Музыка	

Рис. 3.10: Создание и перемещение

3. Определяем опции команды chmod, необходимые для того, чтобы присвоить перечисленным ниже файлам выделенные права доступа, считая, что в начале таких прав нет.(рис.3.11)

```

evbondarenko@dk3n60 ~ $ mkdir australia
evbondarenko@dk3n60 ~ $ mkdir play
evbondarenko@dk3n60 ~ $ touch ty_os
evbondarenko@dk3n60 ~ $ touch feathers
evbondarenko@dk3n60 ~ $ chmod 744 australia
evbondarenko@dk3n60 ~ $ chmod 711 play
evbondarenko@dk3n60 ~ $ chmod 544 my_os
evbondarenko@dk3n60 ~ $ chmod 664 feathers
evbondarenko@dk3n60 ~ $ ls -l
итого 41
-rw-rw-r-- 1 evbondarenko studsci 0 мая 4 15:30 abc1
drwxr-xr-x 10 evbondarenko studsci 2048 окт 21 2021 Architecture_PC
drwxr--r-- 2 evbondarenko studsci 2048 мая 4 15:42 australia
-rw-rw-r-- 1 evbondarenko studsci 0 мая 4 15:43 feathers
drwxr-xr-x 3 evbondarenko studsci 2048 ноя 11 14:15 GNUstep
-rw-r--r-- 1 evbondarenko studsci 0 мая 4 14:38 mcu
drwx--x--x 2 evbondarenko studsci 2048 мая 4 14:25 monthly
-r-xr--r-- 1 evbondarenko studsci 0 мая 4 15:42 my_os
drwxr-xr-x 9 evbondarenko studsci 2048 апр 29 17:53 os-intro
drwx--x--x 2 evbondarenko studsci 2048 мая 4 15:42 play
drwxr-xr-x 3 evbondarenko studsci 2048 сен 1 2021 public
lrwxr-xr-x 1 evbondarenko root 18 апр 14 21:03 public_html -> public/public_html
drwxr-xr-x 3 evbondarenko studsci 2048 мая 4 14:34 reports
drwxr-xr-x 4 evbondarenko studsci 2048 мая 4 15:39 ski.places
drwxr-xr-x 2 evbondarenko studsci 2048 сен 30 2021 tmp
drwxr-xr-x 2 evbondarenko studsci 2048 сен 2 2021 Видео
drwxr-xr-x 2 evbondarenko studsci 2048 апр 29 16:14 Документы
drwxr-xr-x 2 evbondarenko studsci 4096 мая 4 14:21 Загрузки
drwxr-xr-x 2 evbondarenko studsci 4096 мая 4 14:19 Изображения
drwxr-xr-x 2 evbondarenko studsci 2048 сен 2 2021 Музыка
drwxr-xr-x 2 evbondarenko studsci 2048 сен 2 2021 Общедоступные
drwxr-xr-x 2 evbondarenko studsci 2048 апр 29 16:27 'Рабочий стол'
drwxr-xr-x 2 evbondarenko studsci 2048 сен 2 2021 Шаблоны

```

Рис. 3.11: Опции команды chmod

#### 4. Просмотрим содержимое файла /etc/passwd.(рис.3.12)

```

evbondarenko@dk3n60 ~ $ cat /etc/passwd
root:x:0:0:System user:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/bin/false
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:Mail program user:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/bin/false
uucp:x:10:14:uucp:/var/spool/uucppublic:/bin/false

```

Рис. 3.12: содержимое файла /etc/passwd

#### Скопируем файл ~/feathers в файл ~/file.old.(рис.3.13)

```

evbondarenko@dk3n60 ~ $ cp feathers file.old

```

Рис. 3.13: Копирование

Переместим файл ~/file.old в каталог ~/play.(рис.3.14)

```
evbondarenko@dk3n60 ~ $ mv file.old play
```

Рис. 3.14: Перемещение

Скопируем каталог ~/play в каталог ~/fun.(рис.3.15)

```
evbondarenko@dk3n60 ~ $ cp -r play fun
```

Рис. 3.15: Копирование

Переместим каталог ~/fun в каталог ~/play и называем его games.(рис.3.16)

```
evbondarenko@dk3n60 ~ $ mv fun play
evbondarenko@dk3n60 ~ $ mv play/fun games
```

Рис. 3.16: Перемещение и переименование

1)Лишим владельца файла ~/feathers права на чтение (команда «chmod u-r feathers»)..(рис.3.17) 2) Если мы попытаемся просмотреть файл ~/feathers командой cat, то получим отказ в доступе, т.к. в предыдущем пункте лишили владельца права на чтение данного файла..(рис.3.17) 3) Если мы попытаемся скопировать файл ~/feathers, например, в каталог monthly, то получим отказ в доступе, по причине, описанной в предыдущем пункте.(рис.3.17) 4)Дадим владельцу файла ~/feathers право на чтение (команда «chmod u+r feathers»)(рис.3.17) 5) Лишим владельца каталога ~/play права на выполнение (команда «chmod u-x play»).(рис.3.17) 6) Перейдем в каталог ~/play (команда «cd play»). Получим отказ в доступе, т.к. в предыдущем пункте лишили владельца права на выполнение данного каталога..(рис.3.17) 7) Дадим владельцу каталога ~/play право на выполнение (команда «chmod u+x play»).(рис.3.17)

```

evbondarenko@dk3n60 ~ $ chmod u-r feathers
evbondarenko@dk3n60 ~ $ chmod u+r feathers
evbondarenko@dk3n60 ~ $ chmod u-r play
evbondarenko@dk3n60 ~ $ cd play
evbondarenko@dk3n60 ~/play $ cd
evbondarenko@dk3n60 ~ $ chmod u-x play
evbondarenko@dk3n60 ~ $ cd play
evbondarenko@dk3n60 ~/play $ cd
evbondarenko@dk3n60 ~ $ chmod u-x play
evbondarenko@dk3n60 ~ $ chmod u+r play

```

Рис. 3.17: Лишение и получение прав владельца

#### 5. Прочитаем man по командам mount, fsck, mkfs, kill. Рисунки 3.18;3.19;3.20;3.21

Команда `mount`: предназначена для монтирования файловой системы. Все файлы, доступные в Unix системах, составляют иерархическую файловую структуру, которая имеет ветки (каталоги) и листья (файлы в каталогах). Корень этого дерева обозначается как `/`. Физически файлы могут располагаться на различных устройствах. Команда `mount` служит для подключения файловых систем разных устройств к этому большому дереву. Наиболее часто встречающаяся форма команды `mount` выглядит следующим образом: «`mount -t vfstype device dir`». Такая команда предлагает ядру смонтировать (подключить) файловую систему указанного типа `vfstype`, расположенную на устройстве `device`, к заданному каталогу `dir`, который часто называют точкой монтирования. (рис.3.18)

```

MKFS(8)                                System Administration                                MKFS(8)

NAME
    mkfs - build a Linux filesystem

SYNOPSIS
    mkfs [options] [-t type] [fs-options] device [size]

DESCRIPTION
    This mkfs frontend is deprecated in favour of filesystem specific
    mkfs.<type> utils.

    mkfs is used to build a Linux filesystem on a device, usually a hard
    disk partition. The device argument is either the device name (e.g.,
    /dev/hda1, /dev/sdb2), or a regular file that shall contain the
    filesystem. The size argument is the number of blocks to be used for
    the filesystem.

    The exit status returned by mkfs is 0 on success and 1 on failure.

```

Рис. 3.18: mount

Команда `fsck`: это утилита командной строки, которая позволяет выполнять проверки согласованности и интерактивное исправление в одной или нескольких файловых системах Linux. Он использует программы, специфичные для типа файловой системы, которую он проверяет. У команды `fsck` следующий синтаксис: `fsck [параметр] – [параметры ФС] [ . . . ]` Например, если нужно восстановить («починить») файловую систему на некотором устройстве `/dev/sdb2`, следует воспользоваться командой: «`sudo fsck -y /dev/sdb2`» Опция `-y` необходима, т. к. при её отсутствии придётся слишком часто давать подтверждение. (рис.3.19)

```

FSCK(8)                                System Administration                                FSCK(8)

NAME
    fsck - check and repair a Linux filesystem

SYNOPSIS
    fsck [-lsAVRTMNP] [-r [fd]] [-C [fs]] [-t fstype] [filesystem...] [--]
    [fs-specific-options]

DESCRIPTION
    fsck is used to check and optionally repair one or more Linux filesystems.
    filesystem can be a device name (e.g., /dev/hda1, /dev/sdb2), a
    mount point (e.g., /, /usr, /home), or an filesystem label or UUID
    specifier (e.g., UUID=8868abf6-88c5-4a83-98b8-bfc24057f7bd or LABEL=root).
    Normally, the fsck program will try to handle filesystems on different
    physical disk drives in parallel to reduce the total amount of time needed
    to check all of them.

    If no filesystems are specified on the command line, and the -A option is
    not specified, fsck will default to checking filesystems in /etc/fstab
    serially. This is equivalent to the -As options.

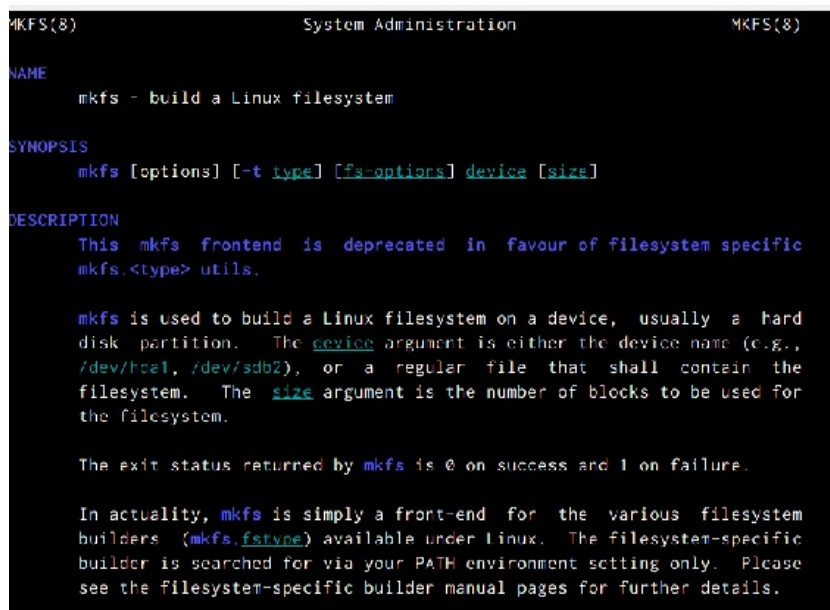
    The exit status returned by fsck is the sum of the following conditions:

```

Рис. 3.19: fsck



Команда `kill`: посылает сигнал процессу или выводит список допустимых сигналов. Имеет следующий синтаксис: `kill [опции] PID`, где PID – это PID (числовой идентификатор) процесса или несколько PID процессов, если требуется послать сигнал сразу нескольким процессам. Например, команда «`kill -KILL 3121`» посылает сигнал KILL процессу с PID 3121, чтобы принудительно завершить процесс.(рис.3.20)



```

MKFS(8)                                     System Administration                                     MKFS(8)

NAME
    mkfs - build a Linux filesystem

SYNOPSIS
    mkfs [options] [-t type] [fs-options] device [size]

DESCRIPTION
    This mkfs frontend is deprecated in favour of filesystem specific
    mkfs.<type> utils.

    mkfs is used to build a Linux filesystem on a device, usually a hard
    disk partition. The device argument is either the device name (e.g.,
    /dev/hda1, /dev/sdb2), or a regular file that shall contain the
    filesystem. The size argument is the number of blocks to be used for
    the filesystem.

    The exit status returned by mkfs is 0 on success and 1 on failure.

    In actuality, mkfs is simply a front-end for the various filesystem
    builders (mkfs.fstype) available under Linux. The filesystem-specific
    builder is searched for via your PATH environment setting only. Please
    see the filesystem-specific builder manual pages for further details.
  
```

Рис. 3.20: kill

Команда `mkfs`: создаёт новую файловую систему Linux. Имеет следующий синтаксис: `mkfs [ -V ] [ -t fstype ] [ fs-options ] filesys [ blocks ]` `mkfs` используется для создания файловой системы Linux на некотором устройстве, обычно в разделе жёсткого диска. В качестве аргумента `filesys` для файловой системы может выступать или название устройства (например, `/dev/hda1`, `/dev/sdb2`) или точка монтирования (например, `/`, `/usr`, `/home`). Аргументом `blocks` указывается количество блоков, которые выделяются для использования этой файловой системой. По окончании работы `mkfs` возвращает 0 - в случае успеха, а 1 - при неудачной операции. Например, команда «`mkfs -t ext2 /dev/hdb1`» создаёт файловую систему типа `ext2` в разделе `/dev/hdb1` (второй жёсткий диск).(рис.3.21)

```
KILL(1)                                User Commands                                KILL(1)

NAME
    kill - send a signal to a process

SYNOPSIS
    kill [options] <pid> [...]

DESCRIPTION
    The default signal for kill is TERM. Use -l or -L to list available
    signals. Particularly useful signals include HUP, INT, KILL, STOP,
    CONT, and 0. Alternate signals may be specified in three ways: -9,
    -SIGKILL or -KILL. Negative PID values may be used to choose whole
    process groups; see the PGID column in ps command output. A PID of -1
    is special; it indicates all processes except the kill process itself
    and init.
```

Рис. 3.21: mkfs



## 4 Контрольные вопросы

- 1) Чтобы узнать, какие файловые системы существуют на жёстком диске моего компьютера, использую команду «df -Th». Из рисунка видно, что на моем компьютере есть следующие файловые системы: devtmpfs, tmpfs, ext4, iso9660. devtmpfs позволяет ядру создать экземпляр tmpfs с именем devtmpfs при инициализации ядра, прежде чем регистрируется какое-либо устройство с драйверами. Каждое устройство с майором / минором будет предоставлять узел устройства в devtmpfs. devtmpfs монтируется на /dev и содержит специальные файлы устройств для всех устройств. tmpfs – временное файловое хранилище во многих Unix-подобных ОС. Предназначена для монтирования файловой системы, но размещается в ОЗУ вместо ПЗУ. Подобная конструкция является RAM диском. Данная файловая система также предназначена для быстрого и ненадёжного хранения временных данных. Хорошо подходит для /tmp и массовой сборки пакетов/образов. Предполагает наличие достаточного объёма виртуальной памяти. Файловая система tmpfs предназначена для того, чтобы использовать часть физической памяти сервера как обычный дисковый раздел, в котором можно сохранять данные (чтение и запись). Поскольку данные размещены в памяти, то чтение или запись происходят во много раз быстрее, чем с обычного HDD диска. ext4 – имеет обратную совместимость с предыдущими версиями ФС. Эта версия была выпущена в 2008 году. Является первой ФС из «семейства» Ext, использующая механизм «extent file system», который позволяет добиться меньшей фрагментации файлов и увеличить общую производительность

файловой системы. Кроме того, в Ext4 реализован механизм отложенной записи (delayed allocation – delalloc), который так же уменьшает фрагментацию диска и снижает нагрузку на CPU. С другой стороны, хотя механизм отложенной записи и используется во многих ФС, но в силу сложности своей реализации он повышает вероятность утери данных. Характеристики:

- максимальный размер файла: 16 TB;
- максимальный размер раздела: 16 TB;
- максимальный размер имени файла: 255 символов. Рекомендации по использованию:
- наилучший выбор для SSD;
- наилучшая производительность по сравнению с предыдущими Ext-системами;
- она так же отлично подходит в качестве файловой системы для серверов баз данных, хотя сама система и моложе Ext3. ISO 9660 – стандарт, выпущенный Международной организацией по стандартизации, описывающий файловую систему для дисков CDRом. Также известен как CDFS (Compact Disc File System). Целью стандарта является обеспечить совместимость носителей под разными операционными системами, такими, как Unix, Mac OS, Windows.

2) Файловая система Linux/UNIX физически представляет собой пространство раздела диска разбитое на блоки фиксированного размера, кратные размеру сектора – 1024, 2048, 4096 или 8120 байт. Размер блока указывается при создании файловой системы. В файловой структуре Linux имеется один корневой раздел – / (он же root, корень). Все разделы жесткого диска (если их несколько) представляют собой структуру подкаталогов, “примонтированных” к определенным каталогам. / – корень Это главный каталог в системе Linux. По сути, это и есть файловая система Linux. Адреса всех файлов начинаются с корня, а дополнительные разделы, флешки или оптические диски

подключаются в папки корневого каталога. Только пользователь root имеет право читать и изменять файлы в этом каталоге.

- /BIN – бинарные файлы пользователя Этот каталог содержит исполняемые файлы. Здесь расположены программы, которые можно использовать в однопользовательском режиме или режиме восстановления.
- /SBIN – системные исполняемые файлы Так же как и /bin, содержит двоичные исполняемые файлы, которые доступны на ранних этапах загрузки, когда не примонтирован каталог /usr. Но здесь находятся программы, которые можно выполнять только с правами суперпользователя.
- /ETC – конфигурационные файлы В этой папке содержатся конфигурационные файлы всех программ, установленных в системе. Кроме конфигурационных файлов, в системе инициализации Init Scripts, здесь находятся скрипты запуска и завершения системных демонов, монтирования файловых систем и автозагрузки программ.
- /DEV – файлы устройств В Linux все, в том числе внешние устройства являются файлами. Таким образом, все подключенные флешки, клавиатуры, микрофоны, камеры – это просто файлы в каталоге /dev/. Выполняется сканирование всех подключенных устройств и создание для них специальных файлов.
- /PROC – информация о процессах По сути, это псевдофайловая система, содержащая подробную информацию о каждом процессе, его Pid, имя исполняемого файла, параметры запуска, доступ к оперативной памяти и так далее. Также здесь можно найти информацию об использовании системных ресурсов.
- /VAR – переменные файлы Название каталога /var говорит само за себя, он должен содержать файлы, которые часто изменяются. Размер этих файлов постоянно увеличивается. Здесь содержатся файлы системных журналов, различные кешы, базы данных и так далее.
- /TMP – временные файлы В этом каталоге содержатся временные файлы,

созданные системой, любыми программами или пользователями. Все пользователи имеют право записи в эту директорию.

- /USR – программы пользователя Это самый большой каталог с большим количеством функций. Здесь находятся исполняемые файлы, исходники программ, различные ресурсы приложений, картинки, музыку и документацию.
  - /HOME – домашняя папка В этой папке хранятся домашние каталоги всех пользователей. В них они могут хранить свои личные файлы, настройки программ и т. д.
  - /BOOT – файлы загрузчика Содержит все файлы, связанные с загрузчиком системы. Это ядро vmlinuz, образ initrd, а также файлы загрузчика, находящиеся в каталоге /boot/grub.
  - /LIB – системные библиотеки Содержит файлы системных библиотек, которые используются исполняемыми файлами в каталогах /bin и /sbin.
  - /OPT – дополнительные программы В эту папку устанавливаются проприетарные программы, игры или драйвера. Это программы созданные в виде отдельных исполняемых файлов самими производителями.
  - /MNT – монтирование В этот каталог системные администраторы могут монтировать внешние или дополнительные файловые системы.
  - /MEDIA – съемные носители В этот каталог система монтирует все подключаемые внешние накопители –USB флешки, оптические диски и другие носители информации.
  - /SRV – сервер В этом каталоге содержатся файлы серверов и сервисов.
  - /RUN - процессы Каталог, содержащий PID файлы процессов, похожий на /var/run, но в отличие от него, он размещен в TMPFS, а поэтому после перезагрузки все файлы теряются.
- 3) Чтобы содержимое некоторой файловой системы было доступно операционной системе необходимо воспользоваться командой mount.
- 4) Целостность файловой системы может быть нарушена из-за перебоев в

питании, неполадок в оборудовании или из-за некорректного/внезапного выключения компьютера. Чтобы устранить повреждения файловой системы необходимо использовать команду `fsck`.

5) Файловую систему можно создать, используя команду `mkfs`. Ее краткое описание дано в пункте 5) в ходе выполнения заданий лабораторной работы.

6) Для просмотра текстовых файлов существуют следующие команды:

- `cat` Задача команды `cat` очень проста – она читает данные из файла или стандартного ввода и выводит их на экран. Синтаксис утилиты: `cat [опции] файл1 файл2 ...` Основные опции: `-b` – нумеровать только непустые строки `-E` – показывать символ `$` в конце каждой строки `-n` – нумеровать все строки `-s` – удалять пустые повторяющиеся строки `-T` – отображать табуляции в виде `^I` `-h` – отобразить справку `-v` – версия утилиты
- `nl` Команда `nl` действует аналогично команде `cat`, но выводит еще и номера строк в столбце слева.
- `less` Существенно более развитая команда для пролистывания текста. При чтении данных со стандартного ввода она создает буфер, который позволяет листать текст как вперед, так и назад, а также искать как по направлению к концу, так и по направлению к началу текста. Синтаксис аналогичный синтаксису команды `cat`. Некоторые опции: `-g` – при поиске подсвечивать только текущее найденное слово (по умолчанию подсвечиваются все вхождения) `-N` – показывать номера строк
- `head` Команда `head` выводит начальные строки (по умолчанию – 10) из одного или нескольких документов. Также она может показывать данные, которые передает на вывод другая утилита. Синтаксис аналогичный синтаксису команды `cat`. Основные опции: `-c` (`-bytes`) – позволяет задавать количество текста не в строках, а в байтах `-n` (`-lines`) – показывает заданное количество строк вместо 10, которые выводятся по умолчанию `-q` (`-quiet`, `-silent`) – выводит только текст, не добавляя к нему название файла `-v` (`-verbose`) – перед текстом выводит название файла `-z` (`-zero-terminated`) – символы

перехода на новую строку заменяет символами завершения строк

- **tail** Эта команда позволяет выводить заданное количество строк с конца файла, а также выводить новые строки в интерактивном режиме. Синтаксис аналогичный синтаксису команды **cat**. Основные опции: **-c** – выводить указанное количество байт с конца файла **-f** – обновлять информацию по мере появления новых строк в файле **-n** – выводить указанное количество строк из конца файла **-pid** – используется с опцией **-f**, позволяет завершить работу утилиты, когда завершится указанный процесс **-q** – не выводить имена файлов **-retry** – повторять попытки открыть файл, если он недоступен **-v** – выводить подробную информацию о файле

- 7) Утилита **cp** позволяет полностью копировать файлы и директории. Синтаксис: **cp [опции] файл-источник файл-приемник** После выполнения команды файл-источник будет полностью перенесен в файл-приемник. Если в конце указан слэш, файл будет записан в заданную директорию с оригинальным именем. Основные опции: **-attributes-only** – не копировать содержимое файла, а только флаги доступа и владельца **-f, -force** – перезаписывать существующие файлы **-i, -interactive** – спрашивать, нужно ли перезаписывать существующие файлы **-L** – копировать не символические ссылки, а то, на что они указывают **-n** – не перезаписывать существующие файлы **-P** – не следовать символическим ссылкам **-r** – копировать папку Linux рекурсивно **-s** – не выполнять копирование файлов в Linux, а создавать символические ссылки **-u** – скопировать файл, только если он был изменён **-x** – не выходить за пределы этой файловой системы **-p** – сохранять владельца, временные метки и флаги доступа при копировании **-t** – считать файл-приемник директорией и копировать файл-источник в эту директорию
- 8) Команда **mv** используется для перемещения одного или нескольких файлов (или директорий) в другую директорию, а также для переименования файлов и директорий. Синтаксис: **mv [-опции] старый\_файл новый\_файл** Основные опции: **-help** – выводит на экран официальную документацию

об утилите `-version` – отображает версию `mv -b` – создает копию файлов, которые были перемещены или перезаписаны `-f` – при активации не будет спрашивать разрешение у владельца файла, если речь идет о перемещении или переименовании файла `-i` – наоборот, будет спрашивать разрешение у владельца `-n` – отключает перезапись уже существующих объектов `-strip-trailing-slashes` – удаляет завершающий символ `/` у файла при его наличии `-t [директория]` – перемещает все файлы в указанную директорию `-u` – осуществляет перемещение только в том случае, если исходный файл новее объекта назначения `-v` – отображает сведения о каждом элементе во время обработки команды Команда `rename` также предназначена, чтобы переименовать файл. Синтаксис: `rename [опции] старое_имя новое_имя файлы` Основные опции: `-v` – вывести список обработанных файлов `-n` – тестовый режим, на самом деле никакие действия выполнены не будут `-f` – принудительно перезаписывать существующие файлы

- 9) Права доступа – совокупность правил, регламентирующих порядок и условия доступа субъекта к объектам информационной системы (информации, её носителям, процессам и другим ресурсам) установленных правовыми документами или собственником, владельцем информации. Права доступа к файлу или каталогу можно изменить, воспользовавшись командой `chmod`. Сделать это может владелец файла (или каталога) или пользователь с правами администратора. Синтаксис команды: `chmod режим имя_файла` Режим имеет следующие компоненты структуры и способ записи: `=` установить право

- лишить права
- дать право `r` чтение `w` запись `x` выполнение `u` (user) владелец файла `g` (group) группа, к которой принадлежит владелец файла `o` (others) все остальные

## 5 Выводы

В ходе выполнения лабораторной работы я ознакомилась с файловой системой Linux, её структурой, именами и содержанием каталогов. А также приобрела практические навыки по применению команд для работы с файлами и каталогами, по управлению процессами (и работами), по проверке использования диска и обслуживанию файловой системы.



## **Список литературы**