

Отчёт по лабораторной работе №7

Анализ файловой структуры UNIX. Команды для работы с файлами и каталогами

Видмаер Егор Романович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	13
4	Контрольные вопросы	14

List of Figures

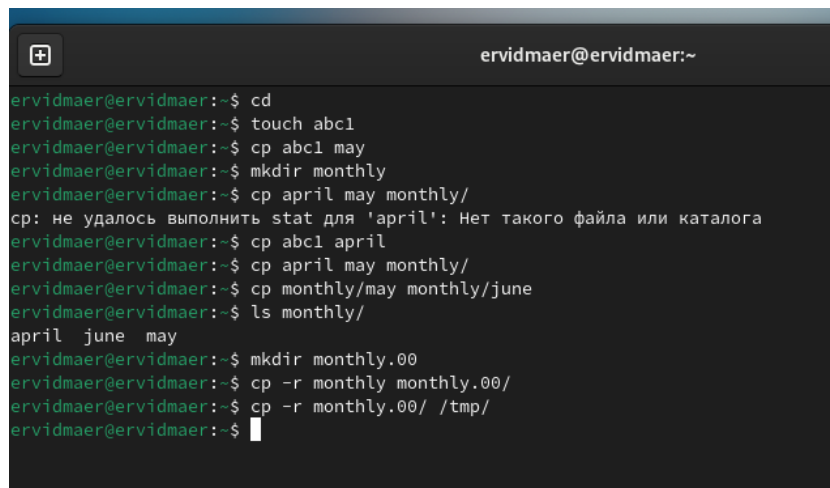
2.1	Выполнение примеров	5
2.2	Выполнение примеров	5
2.3	Выполнение примеров	6
2.4	Работа с каталогами	6
2.5	Настройка прав доступа	7
2.6	Файл /etc/passwd	8
2.7	Работа с файлами и правами доступа	8
2.8	Команда mount	9
2.9	Команда fsck	10
2.10	Команда mkfs	11
2.11	Команда kill	12

1 Цель работы

Ознакомление с файловой системой Linux, её структурой, именами и содержанием каталогов. Приобретение практических навыков по применению команд для работы с файлами и каталогами, по управлению процессами, по проверке использования диска и обслуживанию файловой системы.

2 Выполнение лабораторной работы

1. Выполним примеры, приведённые в первой части описания лабораторной работы.



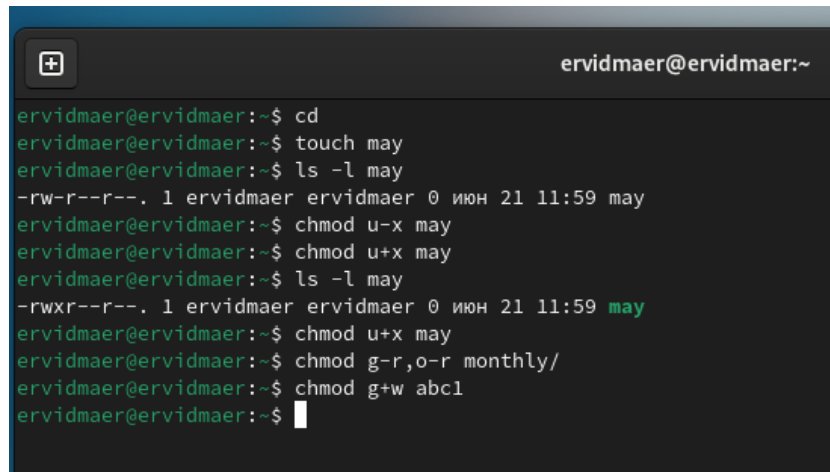
```
eravidmaer@eravidmaer:~  
eravidmaer@eravidmaer:~$ cd  
eravidmaer@eravidmaer:~$ touch abc1  
eravidmaer@eravidmaer:~$ cp abc1 may  
eravidmaer@eravidmaer:~$ mkdir monthly  
eravidmaer@eravidmaer:~$ cp april may monthly/  
cp: не удалось выполнить stat для 'april': Нет такого файла или каталога  
eravidmaer@eravidmaer:~$ cp abc1 april  
eravidmaer@eravidmaer:~$ cp april may monthly/  
eravidmaer@eravidmaer:~$ cp monthly/may monthly/june  
eravidmaer@eravidmaer:~$ ls monthly/  
april  june  may  
eravidmaer@eravidmaer:~$ mkdir monthly.00  
eravidmaer@eravidmaer:~$ cp -r monthly monthly.00/  
eravidmaer@eravidmaer:~$ cp -r monthly.00/ /tmp/  
eravidmaer@eravidmaer:~$
```

Figure 2.1: Выполнение примеров



```
eravidmaer@eravidmaer:~$  
eravidmaer@eravidmaer:~$  
eravidmaer@eravidmaer:~$ cd  
eravidmaer@eravidmaer:~$ mv april july  
eravidmaer@eravidmaer:~$ mv july monthly.00/  
eravidmaer@eravidmaer:~$ ls monthly.00/  
july  monthly  
eravidmaer@eravidmaer:~$ mv monthly.01 reports  
mv: не удалось выполнить stat для 'monthly.01': Нет такого файла или каталога  
eravidmaer@eravidmaer:~$ mv monthly.00/ monthly.01  
eravidmaer@eravidmaer:~$ mv monthly.01 reports  
eravidmaer@eravidmaer:~$ mv reports/monthly/  
april  june  may  
eravidmaer@eravidmaer:~$ mv reports/  
july    monthly/  
eravidmaer@eravidmaer:~$ mv reports/
```

Figure 2.2: Выполнение примеров


A terminal window titled 'ervidmaer@ervidmaer:~' showing a series of commands and their outputs. The user creates a file 'may' with 'touch may', checks its permissions with 'ls -l may' (showing -rw-r--r--), removes execute permissions with 'chmod u-x may', adds them back with 'chmod u+x may', and checks again (showing -rwxr--r--). Then, they set permissions for a directory 'monthly/' with 'chmod g-r,o-r monthly/' and for a file 'abc1' with 'chmod g+w abc1'.

```
ervidmaer@ervidmaer:~$ cd
ervidmaer@ervidmaer:~$ touch may
ervidmaer@ervidmaer:~$ ls -l may
-rw-r--r--. 1 ervidmaer ervidmaer 0 июн 21 11:59 may
ervidmaer@ervidmaer:~$ chmod u-x may
ervidmaer@ervidmaer:~$ chmod u+x may
ervidmaer@ervidmaer:~$ ls -l may
-rwxr--r--. 1 ervidmaer ervidmaer 0 июн 21 11:59 may
ervidmaer@ervidmaer:~$ chmod u+x may
ervidmaer@ervidmaer:~$ chmod g-r,o-r monthly/
ervidmaer@ervidmaer:~$ chmod g+w abc1
ervidmaer@ervidmaer:~$
```

Figure 2.3: Выполнение примеров

2.1. Скопируем файл /usr/include/sys/io.h в домашний каталог и переименуем его equipment. Такого нет, взяли другой файл.

2.2. - 2.5. В домашнем каталоге создаем директорию ski.places. и перемещаем в него файл equipment. Переименовываем файл equipment в equiplist. После этого создаем в домашнем каталоге файл abc1 и копируем его в каталог ski.places. и переименовываем в equiplist2. 2.6. - 2.7. Создаем каталог с именем equipment в каталоге ski.places. Перемещаем файлы equiplist и equiplist2 в каталог equipment. 2.8. Создаем и перемещаем каталог newdir в каталог ski.places и называем его plans.

A terminal window showing a sequence of commands for directory and file management. The user copies /usr/include/linux/sysinfo.h to the home directory and renames it to 'equipment'. They create a directory 'ski.places' and move 'equipment' into it. Then, they rename 'ski.places/equipment' to 'ski.places/equiplist', create a file 'abc1', and copy it to 'ski.places/equiplist2'. They then change to the 'ski.places' directory, create a subdirectory 'equipment', and move 'equiplist' and 'equiplist2' into it. Finally, they create a new directory 'newdir' in the home directory, move it to 'ski.places/newdir', and rename it to 'ski.places/plans'.

```
ervidmaer@ervidmaer:~$ cp /usr/include/linux/sysinfo.h ~
ervidmaer@ervidmaer:~$ mv sysinfo.h equipment
ervidmaer@ervidmaer:~$ mkdir ski.places
ervidmaer@ervidmaer:~$ mv equipment ski.places/
ervidmaer@ervidmaer:~$ mv ski.places/equipment ski.places/equiplist
ervidmaer@ervidmaer:~$ touch abc1
ervidmaer@ervidmaer:~$ cp abc1 ski.places/equiplist2
ervidmaer@ervidmaer:~$ cd ski.places/
ervidmaer@ervidmaer:~/ski.places$ mkdir equipment
ervidmaer@ervidmaer:~/ski.places$ mv equiplist equipment/
ervidmaer@ervidmaer:~/ski.places$ mv equiplist2 equipment/
ervidmaer@ervidmaer:~/ski.places$ cd
ervidmaer@ervidmaer:~$ mkdir newdir
ervidmaer@ervidmaer:~$ mv newdir/ ski.places/
ervidmaer@ervidmaer:~$ mv ski.places/newdir/ ski.places/plans
ervidmaer@ervidmaer:~$
```

Figure 2.4: Работа с каталогами

3. Определим опции команды `chmod`, необходимые для того, чтобы присвоить файлам из хода работы нужные права доступа.

a) Australia (`drwxr-r-`)

b) play (`drwx-x-x`)

c) My_os (`-r-xr-r-`)

d) feathers (`-rw-rw-r-`)

```
eravidmaer@eravidmaer:~$  
eravidmaer@eravidmaer:~$ mkdir australia play  
eravidmaer@eravidmaer:~$ touch my_os feathers  
eravidmaer@eravidmaer:~$ chmod 744 australia/ chmod 711 play/  
chmod: невозможно получить доступ к 'chmod': Нет такого файла или каталога  
chmod: невозможно получить доступ к '711': Нет такого файла или каталога  
eravidmaer@eravidmaer:~$ chmod 744 australia/  
eravidmaer@eravidmaer:~$ chmod 711 play/  
eravidmaer@eravidmaer:~$ chmod 544 my_os  
eravidmaer@eravidmaer:~$ chmod 664 feathers  
eravidmaer@eravidmaer:~$ ls -l  
итого 0  
-rw-rw-r--. 1 eravidmaer eravidmaer 0 июн 21 12:03 abc1  
drwxr--r--. 1 eravidmaer eravidmaer 0 июн 21 12:06 australia  
-rw-rw-r--. 1 eravidmaer eravidmaer 0 июн 21 12:06 feathers  
drwxr-xr-x. 1 eravidmaer eravidmaer 74 июн 21 10:21 git-extended  
-rwxr--r--. 1 eravidmaer eravidmaer 0 июн 21 11:59 may  
drwx--x--x. 1 eravidmaer eravidmaer 24 июн 21 11:49 monthly  
-r-xr--r--. 1 eravidmaer eravidmaer 0 июн 21 12:06 my_os  
drwx--x--x. 1 eravidmaer eravidmaer 0 июн 21 12:06 play  
drwxr-xr-x. 1 eravidmaer eravidmaer 22 июн 21 11:56 reports  
drwxr-xr-x. 1 eravidmaer eravidmaer 28 июн 21 12:04 ski.places  
drwxr-xr-x. 1 eravidmaer eravidmaer 10 июн 21 09:23 work  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 Видео  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 Документы  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 Загрузки  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 Изображения  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 Музыка  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 Общедоступные  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 'Рабочий стол'  
drwxr-xr-x. 1 eravidmaer eravidmaer 0 июн 21 08:59 Шаблоны  
eravidmaer@eravidmaer:~$
```

Figure 2.5: Настройка прав доступа

4.1. Просмотрим содержимое файла `/etc/passwd`.

```
ervidmaer@ervidmaer:~ — less /etc/passwd
root:x:0:0:Super User:/root:/bin/bash
bin:x:1:1:bin:/bin:/usr/sbin/nologin
daemon:x:2:2:daemon:/sbin:/usr/sbin/nologin
adm:x:3:4:adm:/var/adm:/usr/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/usr/sbin/nologin
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:/var/spool/mail:/usr/sbin/nologin
operator:x:11:0:operator:/root:/usr/sbin/nologin
games:x:12:100:games:/usr/games:/usr/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/usr/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/usr/sbin/nologin
dbus:x:81:81:System Message Bus:/usr/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
tss:x:59:59:Account used for TPM access:/usr/sbin/nologin
systemd-coredump:x:998:998:systemd Core Dumper:/usr/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/usr/sbin/nologin
systemd-oom:x:997:997:systemd Userspace OOM Killer:/usr/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/usr/sbin/nologin
systemd-timesync:x:996:996:systemd Time Synchronization:/usr/sbin/nologin
qemu:x:107:107:qemu user:/sbin/nologin
polkitd:x:114:114>User for polkitd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
geoclue:x:995:994>User for geoclue:/var/lib/geoclue:/sbin/nologin
nm-openconnect:x:994:993:NetworkManager user for OpenConnect:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/sbin/nologin
gluster:x:993:992:GlusterFS daemons:/run/gluster:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
```

Figure 2.6: Файл /etc/passwd

4.2 - 4.12. Выполним все указанные действия по перемещению файлов и каталогов

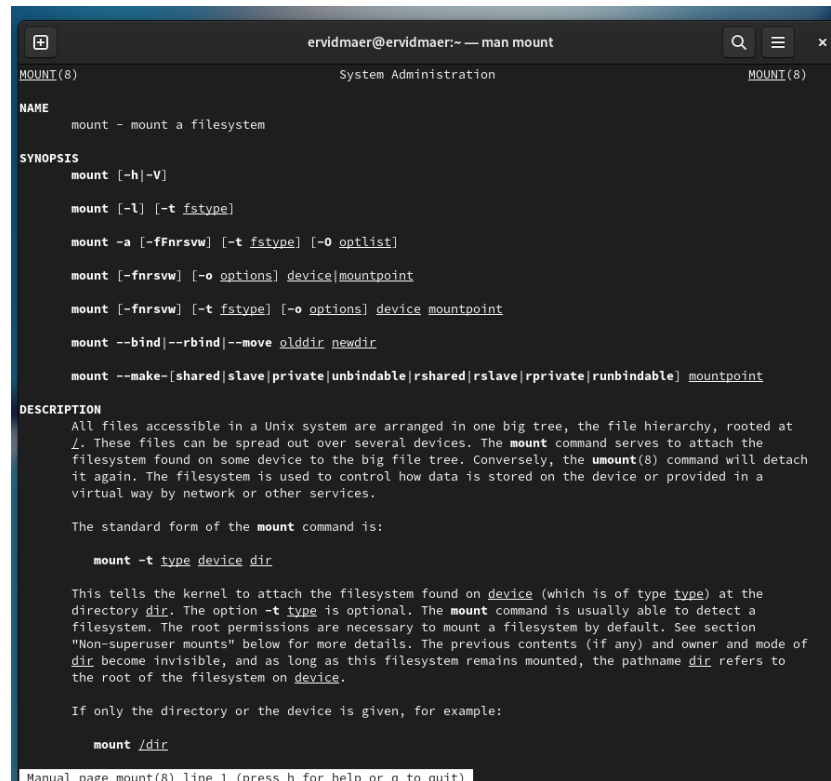
```
ervidmaer@ervidmaer:~
ervidmaer@ervidmaer:~$ cp feathers file.old
ervidmaer@ervidmaer:~$ mv file.old play/
ervidmaer@ervidmaer:~$ mkdir fun
ervidmaer@ervidmaer:~$ cp -R play/ fun/
ervidmaer@ervidmaer:~$ mv fun/ play/games
ervidmaer@ervidmaer:~$ chmod -r feathers
ervidmaer@ervidmaer:~$ cat feathers
cat: feathers: Отказано в доступе
ervidmaer@ervidmaer:~$ cp feathers feathers2
cp: невозможно открыть 'feathers' для чтения: Отказано в доступе
ervidmaer@ervidmaer:~$ chmod u+r feathers
ervidmaer@ervidmaer:~$ chmod -x play/
ervidmaer@ervidmaer:~$ cd play/
bash: cd: play/: Отказано в доступе
ervidmaer@ervidmaer:~$ chmod +x play/
ervidmaer@ervidmaer:~$
```

Figure 2.7: Работа с файлами и правами доступа

4.7. Если мы попытаемся просмотреть файл feathers командой cat, то нам будет отказано в доступе.

4.8. Если мы попытаемся скопировать файл `feathers` то у нас не получится это сделать так как мы ограничили себя в доступе для чтения.

5. Прочитаем `man` по командам `mount`, `fsck`, `mkfs`, `kill` и кратко их охарактеризуем, приведя примеры.



```
ervidmaer@ervidmaer:~ — man mount
MOUNT(8)                                System Administration                                MOUNT(8)

NAME
    mount - mount a filesystem

SYNOPSIS
    mount [-h|-V]

    mount [-l] [-t fstype]

    mount -a [-fFnrsvw] [-t fstype] [-O optlist]

    mount [-fnrsvw] [-o options] device|mountpoint

    mount [-fnrsvw] [-t fstype] [-o options] device mountpoint

    mount --bind|--rbind|--move olddir newdir

    mount --make-[shared|slave|private|unbindable|rshared|rslave|rprivate|runbindable] mountpoint

DESCRIPTION
    All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the filesystem found on some device to the big file tree. Conversely, the umount(8) command will detach it again. The filesystem is used to control how data is stored on the device or provided in a virtual way by network or other services.

    The standard form of the mount command is:

        mount -t type device dir

    This tells the kernel to attach the filesystem found on device (which is of type type) at the directory dir. The option -t type is optional. The mount command is usually able to detect a filesystem. The root permissions are necessary to mount a filesystem by default. See section "Non-superuser mounts" below for more details. The previous contents (if any) and owner and mode of dir become invisible, and as long as this filesystem remains mounted, the pathname dir refers to the root of the filesystem on device.

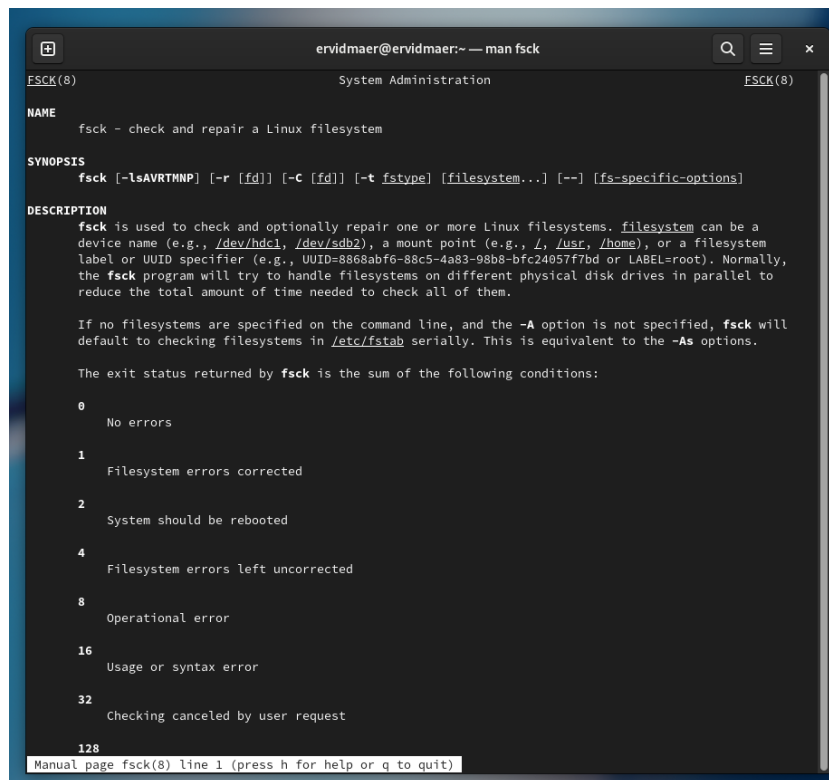
    If only the directory or the device is given, for example:

        mount /dir

Manual page mount(8) line 1 (press h for help or q to quit)
```

Figure 2.8: Команда `mount`

Монтирование файловой системы к общему дереву каталогов. Для размонтирования используется команда `umount`.

A screenshot of a terminal window titled "ervidmaer@ervidmaer:~ — man fsck". The window displays the manual page for the `fsck` command. The content is as follows:

```
FSCK(8)                                System Administration                                FSCK(8)

NAME
    fsck - check and repair a Linux filesystem

SYNOPSIS
    fsck [-lsAVRTMNP] [-r [fd]] [-C [fd]] [-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/hdc1, /dev/sdb2), a mount point (e.g., /, /usr, /home), or a filesystem
    label or UUID specifier (e.g., UUID=8868abf6-88c5-4a83-98b8-bfc24057ff7bd 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:

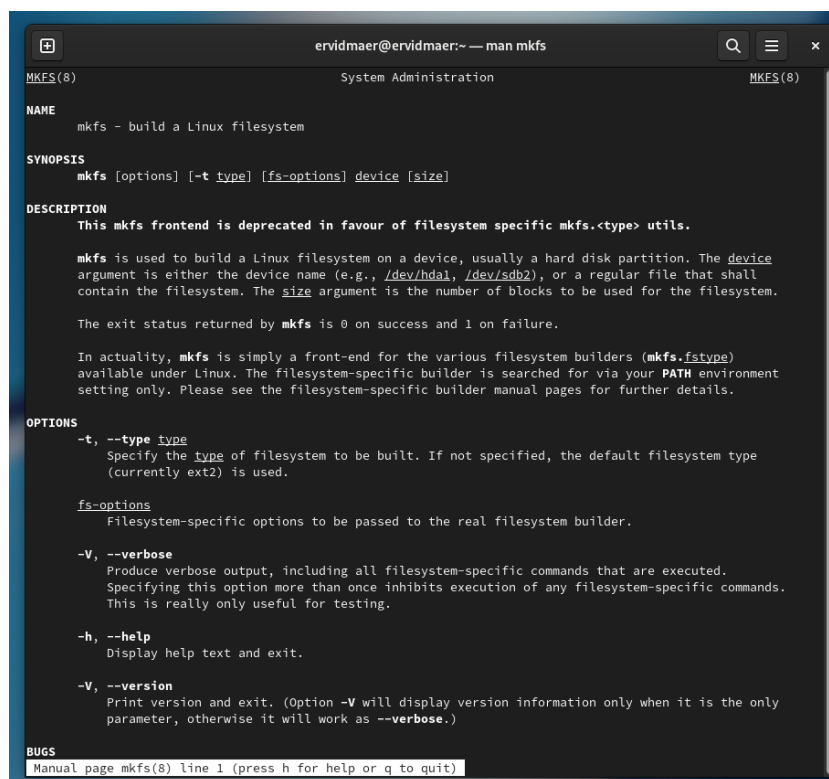
    0      No errors
    1      Filesystem errors corrected
    2      System should be rebooted
    4      Filesystem errors left uncorrected
    8      Operational error
    16     Usage or syntax error
    32     Checking canceled by user request

    128

Manual page fsck(8) line 1 (press h for help or q to quit)
```

Figure 2.9: Команда fsck

`fsck` (проверка файловой системы) – это утилита командной строки, которая позволяет выполнять проверки согласованности и интерактивное исправление в одной или нескольких файловых системах Linux. Она использует программы, специфичные для типа файловой системы, которую она проверяет. Вы можете использовать команду `fsck` для восстановления поврежденных файловых систем в ситуациях, когда система не загружается или раздел не может быть смонтирован.



```
eravidmaer@eravidmaer:~ -- man mkfs
MKFS(8)                                     System Administration                                     MKFS(8)

NAME
    mkfs - build a Linux filesystem

SYNOPSIS
    mkfs [options] [-t ] [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.

OPTIONS
    -t, --type type
        Specify the type of filesystem to be built. If not specified, the default filesystem type (currently ext2) is used.

    fs-options
        Filesystem-specific options to be passed to the real filesystem builder.

    -V, --verbose
        Produce verbose output, including all filesystem-specific commands that are executed. Specifying this option more than once inhibits execution of any filesystem-specific commands. This is really only useful for testing.

    -h, --help
        Display help text and exit.

    -V, --version
        Print version and exit. (Option -V will display version information only when it is the only parameter, otherwise it will work as --verbose.)

BUGS
    Manual page mkfs(8) line 1 (press h for help or q to quit)
```

Figure 2.10: Команда mkfs

Буквы в mkfs значке означают “make file system” (создать файловую систему). Команда обычно используется для управления устройствами хранения в Linux. Вы можете рассматривать mkfs как инструмент командной строки для форматирования диска в определенной файловой системе.

```
ervidmaer@ervidmaer:~ -- man kill
KILL(1) User Commands KILL(1)

NAME
  kill - terminate a process

SYNOPSIS
  kill [-signal|-s signal|-p] [-q value] [-a] [--timeout milliseconds signal] [--] pid|name...
  kill -l [number] | -L

DESCRIPTION
  The command kill sends the specified signal to the specified processes or process groups.

  If no signal is specified, the TERM signal is sent. The default action for this signal is to
  terminate the process. This signal should be used in preference to the KILL signal (number 9),
  since a process may install a handler for the TERM signal in order to perform clean-up steps before
  terminating in an orderly fashion. If a process does not terminate after a TERM signal has been
  sent, then the KILL signal may be used; be aware that the latter signal cannot be caught, and so
  does not give the target process the opportunity to perform any clean-up before terminating.

  Most modern shells have a builtin kill command, with a usage rather similar to that of the command
  described here. The --all, --pid, and --queue options, and the possibility to specify processes by
  command name, are local extensions.

  If signal is 0, then no actual signal is sent, but error checking is still performed.

ARGUMENTS
  The list of processes to be signaled can be a mixture of names and PIDs.

  pid
    Each pid can be expressed in one of the following ways:

    n
      where n is larger than 0. The process with PID n is signaled.

    0
      All processes in the current process group are signaled.

    -1
      All processes with a PID larger than 1 are signaled.

    -n
      Manual page kill(1) line 1 (press h for help or q to quit)
```

Figure 2.11: Команда kill

Системный вызов `kill` может быть использован для посылки какого-либо сигнала какому-либо процессу или группе процесса.

3 Вывод

В ходе данной работы мы ознакомились с файловой системой Linux, её структурой, именами и содержанием каталогов. Научились совершать базовые операции с файлами, управлять правами их доступа для пользователя и групп. Ознакомились с Анализом файловой системы. А также получили базовые навыки по проверке использования диска и обслуживанию файловой системы.

4 Контрольные вопросы

1. Дайте характеристику каждой файловой системе, существующей на жёстком диске компьютера, на котором вы выполняли лабораторную работу.

Ответ: Ext2FS (расширенная файловая система номер два). Многие годы ext2 была файловой системой по умолчанию в GNU/Linux. Ext2 заменила собой Extended File System (вот откуда появилось “Second” в названии). В “новой” файловой системе были исправлены некоторые проблемы, а также убраны ограничения. Отличная стабильность, комплексные инструментальные средства для спасения удаленных файлов, очень долгое время перезагрузки после аварии, есть вероятность частичной или полной потери данных после аварии. Одним из главных недостатков “традиционных” файловых систем, подобных Ext2FS, является низкая сопротивляемость к резким системным сбоям (сбой питания или авария программного обеспечения)

Ext3 (Расширенная файловая система номер три) - является наследником файловой системы Ext2FS. Ext3 совместима с Ext2, но обладает одной новой и очень интересной особенностью –запись. Процесс сохранения объекта происходит прежде чем запись в журнал. В результате мы получаем всегда последовательную файловую систему. Это приводит к тому, что при появлении проблем, проверка и восстановление происходят очень быстро. Время, потраченное на то, чтобы проверить файловую систему таким образом, пропорционально его фактическому использованию и не больше его размера.

ReiserFS (Это тоже журналируемая файловая система подобно Ext3FS, но их

внутренняя структура радикально отличается. В ReiserFS используется концепция бинарных деревьев (binary-tree), позаимствованная из программного обеспечения баз данных.

JFS (журналируемая файловая система). JFS была разработана и использовалась IBM. Вначале JFS была закрытой системой, но недавно IBM решила открыть доступ для движения свободного программного обеспечения. Внутренняя структура JFS близка к ReiserFS. Средняя стабильность, нет комплексных инструментальных средств для спасения удаленных файлов, очень быстрая перезагрузка после аварии, очень хорошее восстановление данных после аварии.

2. Приведите общую структуру файловой системы и дайте характеристику каждой директории первого уровня этой структуры. Ответ:

- Загрузочный блок занимает первый блок файловой системы. Только корневая файловая система имеет активный загрузочный блок, хотя место для него резервируется в каждой файловой системе.
- Суперблок располагается непосредственно за загрузочным блоком и содержит самую общую информацию о ФС (размер ФС, размер области индексных дескрипторов, их число, список свободных блоков, свободные индексные дескрипторы и т. д.). Суперблок всегда находится в оперативной памяти. Различные версии ОС Unix способны поддерживать разные типы файловых систем. Поэтому у структуры суперблока могут быть варианты (сведения о свободных блоках, например, часто хранятся не как список, а как шкала бит), но суперблок всегда располагается за загрузочным блоком. При монтировании файловой системы в оперативной памяти создается копия ее суперблока. Все последующие операции по созданию и удалению файлов влекут изменения копии суперблока в оперативной памяти. Эта копия периодически записывается на магнитный диск. Обычно причиной повреждения файловой системы является отключение электропитания (или зависание

ОС) в тот момент, когда система производит копирование суперблока из оперативной памяти на магнитный диск.

- Область индексных дескрипторов содержит описатели файлов (inode). С каждым файлом связан один inode, но одному inode может соответствовать несколько файлов. В inode хранится вся информация о файле, кроме его имени. Область индексных дескрипторов имеет фиксированный формат и располагается непосредственно за суперблоком. Общее число описателей и, следовательно, максимальное число файлов задается в момент создания файловой системы. Описатели нумеруются натуральными числами. Первый описатель используется ОС для описания специального файла (файла «Плохих блоков»). То есть поврежденные блоки раздела рассматриваются ОС как принадлежащие к специальному файлу и поэтому считаются «занятыми». Второй – описывает корневой каталог файловой системы.
- В области данных расположены как обычные файлы, так и файлы каталогов (в том числе корневой каталог). Специальные файлы представлены в ФС только записями в соответствующих каталогах и индексными дескрипторами специального формата, т. е. места в области памяти не занимают.

3. Какая операция должна быть выполнена, чтобы содержимое некоторой файловой системы было доступно операционной системе? Ответ: Команда `cat` - позволяет вывести на экран содержимое любого файла, однако в таком виде эта команда практически не используется. Если файл слишком большой, то его содержимое пролистается на экране, а Вы увидите только последние строки файла. С помощью этой команды можно комбинировать и объединять копии файлов, а также создавать новые файлы. Если набрать просто в командной строке `cat` и нажать `Enter`, то можно вводить (и соответственно видеть) текст на экране. Повторное нажатие клавиши `Enter` удвоит строку и позволит начать следующую. Когда текст набран, следует одно-

временно нажать клавиши Ctrl и d.

4. Назовите основные причины нарушения целостности файловой системы.

Как устранить повреждения файловой системы? Ответ: Некорректность файловой системы может возникать:

- В результате насильственного прерывания операций ввода-вывода, выполняемых непосредственно с диском.
- В результате нарушения работы дискового кэша. Кэширование данных с диска предполагает, что в течение некоторого времени результаты операций ввода-вывода никак не сказываются на содержимом диска — все изменения происходят с копиями блоков диска, временно хранящихся в буферах оперативной памяти (в этих буферах оседают данные из пользовательских файлов и служебная информация файловой системы, такая как каталоги, индексные дескрипторы, списки свободных, занятых и поврежденных блоков и т. п.)

5. Как создаётся файловая система? Ответ: Общее дерево файлов и каталогов системы Linux формируется из отдельных “ветвей”, соответствующих различным физическим носителям. В UNIX нет понятия “форматирования диска” (и команды форматирования), а используется понятие “создание файловой системы”. Когда мы получаем новый носитель, например, жесткий диск, мы должны создать на нем файловую систему. То есть каждому носителю ставится в соответствие отдельная файловая система. Чтобы эту файловую систему использовать для записи в нее файлов, надо ее вначале подключить в общее дерево каталогов (“смонтировать”). Вот и получается, что можно говорить о монтировании файловых систем или о монтировании носителей (с созданными на них файловыми системами). Например, создается файловая система типа ext2fs. Создание файловой системы типа ext2fs подразумевает создание в данном разделе на диске суперблока, таблицы индексных дескрипторов и совокупности блоков данных. Делает-

ся все это все с помощью команды `mkfs`. В простейшем случае достаточно дать эту команду в следующем формате:

`[root]# mkfs -t ext2 /dev/hda5`, где `/dev/hda5` надо заменить указанием на соответствующее устройство или раздел. Например, если вы хотите создать файловую систему на дискете, то команда примет вид:

`[root]# mkfs -t ext2 /dev/fd0`

После выполнения команды `mkfs` в указанном разделе будет создана файловая система `ext2fs`. В новой файловой системе автоматически создается один каталог с именем `lost+found`. Он используется в экстренных случаях программой `fsck`, поэтому не удаляйте его. Для того, чтобы начать работать с новой файловой системой, необходимо подключить ее в общее дерево каталогов, что делается с помощью команды `mount`. В качестве параметров команде `mount` надо, как минимум, указать устройство и “точку монтирования”. Точкой монтирования называется тот каталог в уже существующем и известном системе дереве каталогов, который будет теперь служить корневым каталогом для подключаемой файловой системы. После монтирования файловой системы в каталог `/mnt/disk2` прежнее содержимое этого каталога станет для вас недоступно до тех пор, пока вы не размонтируете вновь подключенную файловую систему. Прежнее содержимое не уничтожается, а просто становится временно недоступным. Поэтому в качестве точек монтирования лучше использовать пустые каталоги (заранее заготовленные).

6. Дайте характеристику командам, которые позволяют просмотреть текстовые файлы. Ответ: Для просмотра небольших файлов удобно пользоваться командой `cat`. Формат команды: `cat имя-файла`

Для просмотра больших файлов используйте команду `less` — она позволяет осуществлять постраничный просмотр файлов (длина страницы соответствует размеру экрана). Формат команды: `less имя-файла`

Для управления процессом просмотра можно использовать следующие управляющие клавиши: - Space — переход на следующую страницу, - ENTER — сдвиг вперёд на одну строку, - b — возврат на предыдущую страницу, - h — обращение за подсказкой, - q — выход в режим командной строки.

Для просмотра начала файла можно воспользоваться командой head. По умолчанию она выводит первые 10 строк файла. Формат команды: head [-n] имя-файла, где n — количество выводимых строк.

Команда tail выводит несколько (по умолчанию 10) последних строк файла. Формат команды: tail [-n] имя-файла, где n — количество выводимых строк.

7. Приведите основные возможности команды cp в Linux. Ответ: Копирование отдельных файлов Для копирования файла следует использовать утилиту cp с аргументами, представленными путями к исходному и целевому файлам.

Копирование файлов в другую директорию В том случае, если в качестве пути к целевому файлу используется путь к директории, исходные файлы будут скопированы в эту целевую директорию.

Команда cp -r Для копирования директорий целиком следует использовать команду cp -r (параметр -r позволяет осуществлять рекурсивное копирование всех файлов из всех поддиректорий).

Копирование множества файлов в директорию Вы также можете использовать утилиту cp для копирования множества файлов в одну директорию. В этом случае последний аргумент (аргумент, указывающий на цель) должен быть представлен путем к директории.

Команда cp -i Для предотвращения перезаписи существующих файлов в ходе использования утилиты cp следует использовать параметр -i (для активации интерактивного режима копирования).

8. Назовите и дайте характеристику командам перемещения и переименования файлов и каталогов. Ответ: Команды mv и mvdir предназначены для

перемещения и переименования файлов и каталогов. Формат команды mv: mv [-опции] старый_файл новый_файл Примеры:

- Переименование файлов в текущем каталоге. Изменить название файла april на july в домашнем каталоге: cd mv april july
- Перемещение файлов в другой каталог. Переместить файл july в каталог monthly.00: mv july monthly.00 ls monthly.00 Результат: april july june may. Если необходим запрос подтверждения о перезаписи файла, то нужно использовать опцию i.
- Переименование каталогов в текущем каталоге. Переименовать каталог monthly.00 в monthly.01 mv monthly.00 monthly.01
- Перемещение каталога в другой каталог. Переместить каталог monthly.01 в каталог reports: mkdir reports mv monthly.01 reports
- Переименование каталога, не являющегося текущим. Переименовать каталог reports/monthly.01 в reports/monthly: mv reports/monthly.01 reports/monthly

9. Что такое права доступа? Как они могут быть изменены? Ответ: Права доступа — совокупность правил, регламентирующих порядок и условия доступа субъекта к объектам информационной системы (информации, её носителям, процессам и другим ресурсам). Права доступа к файлу или каталогу можно изменить, воспользовавшись командой chmod. Сделать это может владелец файла (или каталога) или пользователь с правами администратора. Формат команды: chmod режим имя_файла Режим (в формате команды) 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 В работе с правами доступа можно использовать их цифровую запись (восьмеричное значение) вместо символической