

Отчёт по лабораторной работе №6

Операционные системы

Сячинова Ксения Ивановна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	13
4	Ответы на контрольные вопросы	14

Список иллюстраций

2.1	Запись в файл	6
2.2	Вывод файлов	7
2.3	Нахождение файлов по символу	7
2.4	Нахождение файлов по символу	8
2.5	Нахождение файлов по символам	8
2.6	Удаление файла	9
2.7	Редактор gedit	9
2.8	Определение идентификатора процесса	9
2.9	Опции команды kill	10
2.10	Завершение процесса	10
2.11	Опции команды df	10
2.12	Опции команды du	11
2.13	Команда df	11
2.14	Команда du	11
2.15	Опции команды find	12
2.16	Выполнение команды	12

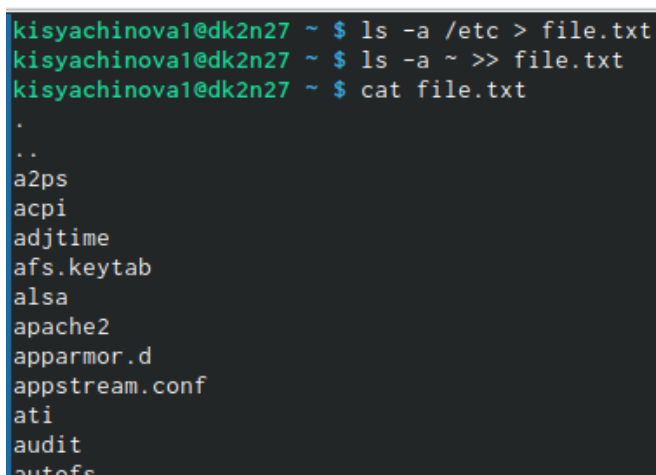
Список таблиц

1 Цель работы

Ознакомиться с инструментами поиска файлов и фильтрации текстовых данных. Приобрести практические навыки: по управлению процессами (и заданиями), по проверке использования диска и обслуживанию файловых систем.

2 Выполнение лабораторной работы

1. Осуществили вход в систему, используя наше имя.
2. Далее запишем в файл *file.txt* названия файлов, содержащихся в каталоге */etc*. Для этого используем команду *ls -a /etc >file.txt*. С помощью команды *ls -a ~ -> file.txt* дописываем в этот же файл названия файлов, содержащихся в домашнем каталоге. Для проверки действий используем команду *cat file.txt*. (рис. 2.1).



```
kisyachinova1@dk2n27 ~ $ ls -a /etc > file.txt
kisyachinova1@dk2n27 ~ $ ls -a ~ >> file.txt
kisyachinova1@dk2n27 ~ $ cat file.txt
.
..
a2ps
acpi
adjtime
afs.keytab
alsa
apache2
apparmor.d
appstream.conf
ati
audit
autofs
```

Рис. 2.1: Запись в файл

3. Нужно вывести имена всех файлов из *file.txt*, которые имеют расширение *.conf* и записать их в новый текстовый файл *conf.txt*. Для этого используем команду *grep -e '.conf\$' file.txt > conf.txt*. Проверяем выполнение действий. (рис. 2.2).

```

kisyachinova1@dk2n27 ~ $ grep -e '\.conf$' file.txt > conf.txt
kisyachinova1@dk2n27 ~ $ cat conf.txt
appstream.conf
brltty.conf
ca-certificates.conf
cachefilesd.conf
cfg-update.conf
dhcpcd.conf
dispatch-conf.conf
dleyna-server-service.conf
dnsmasq.conf
e2fsck.conf
e2scrub.conf
etc-update.conf
fluidsynth.conf
fuse.conf
gai.conf
genkernel.conf
gssapi_mech.conf
host.conf
idmapd.conf
idn2.conf
idnalias.conf

```

Рис. 2.2: Вывод файлов

4. Затем найдём файлы в домашнем каталоге, которые начинаются на *c*. Это можно сделать несколькими командами, которые представлены на рисунке. (рис. 2.3)

```

kisyachinova1@dk2n27 ~ $ find ~ -maxdepth 1 -name "c*" -print
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/conf.txt
kisyachinova1@dk2n27 ~ $ ls ~/c*
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/conf.txt
kisyachinova1@dk2n27 ~ $ ls | grep c*
conf.txt

```

Рис. 2.3: Нахождение файлов по символу

5. После этого выведем на экран (по странично) имена файлов из каталога */etc*, которые начинаются с символа *h*. Для этого я использовала команду `* find /etc -maxdepth 1 -name "h"| less`. (рис. 2.4)

```

/etc/hostname
/etc/hotplug.d
/etc/harbour
/etc/host.conf
/etc/httpd
/etc/highlight
/etc/hosts.allow
/etc/harbour.cfg
/etc/hotplug
/etc/hosts
/etc/htdig
/etc/hal
/etc/hsqldb
~
~

```

Рис. 2.4: Нахождение файлов по символу

6. Запустим в фоновом режиме процесс, который будет записывать в файл `~/logfile`, файлы, которые начинаются с `log` с помощью команды `find / -name "log" > logfile &`. Запустился непрерывный процесс записывания файла. (рис. 2.5)

```

find: '/afs/dk.sci.pfu.edu.ru/home/a/a/aadolbilen/.bashrc': Отказано в до
ступе
find: '/afs/dk.sci.pfu.edu.ru/home/a/a/aadolbilen/.profile': Отказано в до
ступе
find: '/afs/dk.sci.pfu.edu.ru/home/a/a/aadolbilen/.gconf/desktop/%gconf.xml
l': Отказано в доступе
find: '/afs/dk.sci.pfu.edu.ru/home/a/a/aadolbilen/.gconf/desktop/gnome/acc
essibility/keyboard/%gconf.xml': Отказано в доступе
find: '/afs/dk.sci.pfu.edu.ru/home/a/a/aadolbilen/.gconf/desktop/gnome/acc
essibility/%gconf.xml': Отказано в доступе
find: '/afs/dk.sci.pfu.edu.ru/home/a/a/aadolbilen/.gconf/desktop/gnome/%gc
onf.xml': Отказано в доступе
find: '/afs/dk.sci.pfu.edu.ru/home/a/a/aadolbilen/.gconf/desktop/gnome/per
ipherals/keyboard/kbd.sysbackup/%gconf.xml': Отказано в доступе

```

Рис. 2.5: Нахождение файлов по символам

7. Проверим наличие файла `logfile`, а затем с помощью команды `rm logfile` удалим его. (рис. 2.6)


```
kisyachinova1@dk2n27 ~ $ cat logfile
/run/log
/dev/log
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/puppet/modules/pu
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/puppet/modules/pu
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/modules/pupp
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/modules/pupp
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/modules/pupp
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/logs
/afs/dk.sci.pfu.edu.ru/common/files/drupal/vagrant-proxyconf/.git
/afs/dk.sci.pfu.edu.ru/common/files/drupal/vagrant-proxyconf/spec
/afs/dk.sci.pfu.edu.ru/common/files/drupal/vagrant-proxyconf/lib/
/afs/dk.sci.pfu.edu.ru/home/a/a/aadzyubenko/.mozilla/firefox/zytg
/afs/dk.sci.pfu.edu.ru/home/a/a/aaflotskij/work/java/examples/ser
/afs/dk.sci.pfu.edu.ru/home/a/a/aagluxova/.opera/images/login.yah
kisyachinova1@dk2n27 ~ $ rm logfile
kisyachinova1@dk2n27 ~ $ rm logfile
rm: невозможно удалить 'logfile': Нет такого файла или каталога
```

Рис. 2.6: Удаление файла

8. Запускаем в консоли в фоновом режиме редактор *gedit*. После ввода команды *gedit &* появляется окно редактора. (рис. 2.7)

```
kisyachinova1@dk2n27 ~ $ gedit &
[1] 7085
kisyachinova1@dk2n27 ~ $
```

Рис. 2.7: Редактор gedit

9. Для определения идентификатора процесса *gedit* используем команду *ps | grep -i "gedit"*. Из рисунка видно, что наш процесс имеет PID 7085. (рис. 2.8)

```
kisyachinova1@dk2n27 ~ $ gedit &
[1] 7664
kisyachinova1@dk2n27 ~ $ ps |grep -i "gedit"
[1]+  Завершён      gedit
kisyachinova1@dk2n27 ~ $ pgrep gedit
4507
kisyachinova1@dk2n27 ~ $ pidof gedit
4507
kisyachinova1@dk2n27 ~ $
```

Рис. 2.8: Определение идентификатора процесса

10. Далее ознакомимся со справкой команды *kill* и используем её для завершения процесса *gedit*. (рис. 2.9), (рис. 2.10)

```
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 avail-
    able signals. Particularly useful signals include HUP, INT,
    KILL, STOP, CONT, and 0. Alternate signals may be specified in
```

Рис. 2.9: Опции команды kill

```
kisyachinova1@dk2n27 ~ $ man kill
kisyachinova1@dk2n27 ~ $ kill 4507
```

Рис. 2.10: Завершение процесса

11. Далее получим более подробную информацию о командах *df* и *du*.

- *df*– утилита, показывающая список всех файловых систем по именам устройств, сообщает их размер, занятое и свободное пространство и точки монтирования.
- *du* – утилита, предназначенная для вывода информации об объеме дискового пространства, занятого файлами и директориями. Она принимает путь к элементу файловой системы и выводит информацию о количестве байт дискового пространства или блоков диска, задействованных для его хранения (рис. 2.11), (рис. 2.12), (рис. 2.13), (рис. 2.14)

```
DF(1)                                User Commands                                DF(1)

NAME
    df - report file system space usage

SYNOPSIS
    df [OPTION]... [FILE]...

DESCRIPTION
    This manual page documents the GNU version of df. df displays
    the amount of space available on the file system containing each
    file name argument. If no file name is given, the space avail-
    able on all currently mounted file systems is shown. Space is
```

Рис. 2.11: Опции команды df

```

DU(1)                                User Commands                                DU(1)

NAME
    du - estimate file space usage

SYNOPSIS
    du [OPTION]... [FILE]...
    du [OPTION]... --files0-from=F

DESCRIPTION
    Summarize device usage of the set of FILES, recursively for di-
    rectories.

```

Рис. 2.12: Опции команды du

```

kisyachinova1@dk2n27 ~ $ df
Файловая система 1К-блоков  Использовано    Доступно  Использовано%  Смонтиро
вано в
none                3999704          16808      3982896           1% /run
udev                10240             0        10240           0% /dev
tmpfs               3999704             0      3999704           0% /dev/shm
/dev/sda8           484939832      73771008      386461768        17% /
tmpfs               3999708      135864      3863844           4% /tmp
/dev/sda6           50090536      12548      47501092           1% /var/cac
he/openafs          2147483647             0  2147483647           0% /afs
tmpfs               799940           200      799740           1% /run/use
r/4700

```

Рис. 2.13: Команда df

```

t/objects/b4
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/c1
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/5d
23     ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/f6
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/b6
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/8b
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/7e
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/2c
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/68
3      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/48
7      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/46
531    ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/c8
40     ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/7d
16     ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/21
7      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/d6
6      ./work/study/2022-2023/Операционные системы/study_2022-2023_OS/.git/objects/f3

```

Рис. 2.14: Команда du

12. Выведем имена всех директорий, которые имеются в домашнем каталоге, предварительно узнаем опции команды *find*. (рис. 2.15), (рис. 2.16)

```
FIND(1)                                General Commands Manual                                FIND(1)

NAME
    find - search for files in a directory hierarchy

SYNOPSIS
    find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...]
    [expression]

DESCRIPTION
    This manual page documents the GNU version of find.  GNU find
    searches the directory tree rooted at each given starting-point
    by evaluating the given expression from left to right, according
    to the rules of precedence (see section OPERATORS), until the
    outcome is known (the left hand side is false for and operations,
    true for or), at which point find moves on to the next file name.
```

Рис. 2.15: Опции команды find

```
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.texlive2022/texmf-var
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.texlive2022/texmf-var/lu
atex-cache
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.texlive2022/texmf-var/lu
atex-cache/generic
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.texlive2022/texmf-var/lu
atex-cache/generic/names
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.texlive2022/texmf-var/lu
atex-cache/generic/fonts
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.texlive2022/texmf-var/lu
atex-cache/generic/fonts/otl
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.pki
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.pki/nssdb
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.VirtualBox
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.VirtualBox/Machines
kisyachinova1@dk2n27 ~ $ find ~ -type d
```

Рис. 2.16: Выполнение команды

3 Выводы

В ходе выполнения данной лабораторной работы я ознакомилась с инструментами поиска файлов и фильтрации текстовых данных, а также приобрела практические навыки по управлению процессами, по проверке использования диска и обслуживанию файловых систем.

4 Ответы на контрольные вопросы

1. В системе по умолчанию открыто три специальных потока:

- `stdin` – стандартный поток ввода (по умолчанию: клавиатура), файловый дескриптор 0;
- `stdout` – стандартный поток вывода (по умолчанию: консоль), файловый дескриптор 1;
- `stderr` – стандартный поток вывод сообщений об ошибках (по умолчанию: консоль), файловый дескриптор 2. Большинство используемых в консоли команд и программ записывают результаты своей работы в стандартный поток вывода `stdout`.

2. `>` Перенаправление вывода в файл `>>` Перенаправление вывода в файл и открытие файла в режиме добавления (данные добавляются в конец файла)/

3. Конвейер (`pipe`) служит для объединения простых команд или утилит в цепочки, в которых результат работы предыдущей команды передаётся последующей. Синтаксис следующий:

`команда1|команда2` (это означает, что вывод команды 1 передаётся на ввод команде 2)

4. Процесс рассматривается операционной системой как заявка на потребление всех видов ресурсов, кроме одного – процессорного времени. Этот последний важнейший ресурс распределяется операционной системой между

другими единицами работы – потоками, которые и получили свое название благодаря тому, что они представляют собой последовательности (потоки выполнения) команд. Процесс – это выполнение программы. Он считается активной сущностью и реализует действия, указанные в программе. Программа представляет собой статический набор команд, а процесс это набор ресурсов и данных, использующихся при выполнении программы.

5.

- `pid`: идентификатор процесса (PID) процесса (`processID`), к которому вызывают метод
- `gid`: идентификатор группы UNIX, в котором работает программа.

6. Любую выполняющуюся в консоли команду или внешнюю программу можно запустить в фоновом режиме. Для этого следует в конце имени команды указать знак амперсанда `&`. Запущенные фоновые программы называются задачами (`jobs`). Ими можно управлять с помощью команды `jobs`, которая выводит список запущенных в данный момент задач.

7.

- `top` – это консольная программа, которая показывает список работающих процессов в системе. Программа в реальном времени отсортирует запущенные процессы по их нагрузке на процессор.
- `htop` – это продвинутый консольный мониторинг процессов. Утилита выводит постоянно меняющийся список системных процессов, который сортируется в зависимости от нагрузки на ЦПУ. Если делать сравнение `stop`, то `htop` показывает абсолютно все процессы в системе, время их непрерывного использования, загрузку процессоров и расход оперативной памяти.

8. `find` – это команда для поиска файлов и каталогов на основе специальных условий. Ее можно использовать в различных обстоятельствах, например,

для поиска файлов по разрешениям, владельцам, группам, типу, размеру и другим подобным критериям. Команда `find` имеет такой синтаксис:

`find[папка][параметры] критерий шаблон [действие]`

Папка – каталог в котором будем искать

Параметры – дополнительные параметры, например, глубина поиска, и т.д.

Критерий – по какому критерию будем искать: имя, дата создания, права, владелец и т.д.

Шаблон – непосредственно значение по которому будем отбирать файлы.

Основные параметры: - `-P` никогда не открывать символические ссылки - `-L` - получает информацию о файлах по символическим ссылкам. Важно для дальнейшей обработки, чтобы обрабатывалась не ссылка, а сам файл. - `-maxdepth` - максимальная глубина поиска по подкаталогам, для поиска только в текущем каталоге установите 1. - `-depth` - искать сначала в текущем каталоге, а потом в подкаталогах - `-mount` искать файлы только в этой файловой системе. - `-version` - показать версию утилиты `find` - `-print` - выводить полные имена файлов - `-typef` - искать только файлы - `-typed` - поиск папки в Linux

Основные критерии: - `-name` - поиск файлов по имени - `-perm` - поиск файлов в Linux по режиму доступа - `-user` - поиск файлов по владельцу - `-group` - поиск по группе - `-mtime` - поиск по времени модификации файла - `-atime` - поиск файлов по дате последнего чтения - `-nogroup` - поиск файлов, не принадлежащих ни одной группе - `-nouser` - поиск файлов без владельцев - `-newer` - найти файлы новее чем указанный - `-size` - поиск файлов в Linux по их размеру

Примеры:

`find~ -type d` поиск директорий в домашнем каталоге

`find~ -type f -name ".*"` поиск скрытых файлов в домашнем каталоге

9. Файл по его содержимому можно найти с помощью команды `grep`:

«`grep -r` слово/выражение, которое нужно найти».

10. Утилита `df`, позволяет проанализировать свободное пространство на всех подключенных к системе разделах.
11. При выполнении команды `du` (без указания папки и опции) можно получить все файлы и папки текущей директории с их размерами. Для домашнего каталога: `du ~/`
12. Основные сигналы (каждый сигнал имеет свой номер), которые используются для завершения процесса:
- `SIGINT`–самый безобидный сигнал завершения, означает `Interrupt`. Он отправляется процессу, запущенному из терминала с помощью сочетания клавиш `Ctrl+C`. Процесс правильно завершает все свои действия и возвращает управление;
 - `SIGQUIT`–это еще один сигнал, который отправляется с помощью сочетания клавиш, программе, запущенной в терминале. Он сообщает ей что нужно завершиться и программа может выполнить корректное завершение или проигнорировать сигнал. В отличие от предыдущего, она генерирует дампы памяти. Сочетание клавиш `Ctrl+Q`;
 - `SIGHUP`–сообщает процессу, что соединение с управляющим терминалом разорвано, отправляется, в основном, системой при разрыве соединения синтернетом;
 - `SIGTERM`–немедленно завершает процесс, но обрабатывается программой, поэтому позволяет ей завершить дочерние процессы и освободить все ресурсы;
 - `SIGKILL`–тоже немедленно завершает процесс, но, в отличие от предыдущего варианта, он не передается самому процессу, а обрабатывается ядром. Поэтому ресурсы и дочерние процессы остаются запущенными. Также для передачи сигналов процессам в Linux используется утилита `kill`, её синтаксис: `kill [-сигнал] [pid_процесса]`

(PID – уникальный идентификатор процесса). Сигнал представляет собой один

из выше перечисленных сигналов для завершения процесса. Перед тем, как выполнить остановку процесса, нужно определить его PID. Для этого используют команды `ps` и `grep`. Команда `ps` предназначена для вывода списка активных процессов в системе и информации о них. Команда `grep` запускается одновременно с `ps` (в канале) и будет выполнять поиск по результатам команды `ps`.

Утилита `kill` – это оболочка для `kill`, она ведет себя точно так же, и имеет тот же синтаксис, только в качестве идентификатора процесса ей нужно передать его имя.

`killall` работает аналогично двум предыдущим утилитам. Она тоже принимает имя процесса в качестве параметра и ищет его PID в директории `/proc`. Но эта утилита обнаружит все процессы с таким именем и завершит их.