

Отчёт по лабораторной работе №6

Операционные системы

Скрипникова София Дмитриевна

Содержание

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

Список иллюстраций

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

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. ??).

Вывод файлов

4. Затем найдём файлы в домашнем каталоге, которые начинаются на *c*. Это можно сделать несколькими командами, которые представлены на рисунке. (рис. 2.1)

```
[sdscripnikova@sdscripnikova ~]$ find ~ -maxdepth 1 -name "c*" -print
/home/sdscripnikova/cd ~
/home/sdscripnikova/cd ~/.pub
/home/sdscripnikova/conf.txt
[sdscripnikova@sdscripnikova ~]$ ls ~/c*
'/home/sdscripnikova/cd ~'      /home/sdscripnikova/conf.txt
'/home/sdscripnikova/cd ~/.pub'
[sdscripnikova@sdscripnikova ~]$ ls | grep c*
```

Рис. 2.1: 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 -maxdepth1 -name "h"| less`. (рис. 2.2)



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

Рис. 2.2: 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. 2.3)

```
find: '/etc/nftables': Отказано в доступе
find: '/etc/openvpn/client': Отказано в доступе
find: '/etc/openvpn/server': Отказано в доступе
find: '/etc/polkit-1/localauthority': Отказано в доступе
find: '/etc/polkit-1/rules.d': Отказано в доступе
find: '/etc/sos/cleaner': Отказано в доступе
find: '/etc/ssh/sshd_config.d': Отказано в доступе
find: '/etc/sss': Отказано в доступе
find: '/etc/sudoers.d': Отказано в доступе
find: '/lost+found': Отказано в доступе
find: '/usr/libexec/initialscripts/legacy-actions/auditd': Отказано в доступе
```

Рис. 2.3: Нахождение файлов по символам

7. Проверим наличие файла *logfile*, а затем с помощью команды *rm logfile* удалим его. (рис. 2.4)

```
[sdscripnikova@sdscripnikova ~]$ rm logfile
[sdscripnikova@sdscripnikova ~]$ rm logfile
rm: невозможно удалить 'logfile': Нет такого файла или каталога
```

Рис. 2.4: Удаление файла

8. Запускаем в консоли в фоновом режиме редактор *gedit*. После ввода команды *gedit &* появляется окно редактора. (рис. 2.5)

```
[sdscripnikova@sdscripnikova ~]$ gedit &
[1] 3465
```

Рис. 2.5: Редактор gedit

9. Для определения идентификатора процесса *gedit* используем команду *ps | grep -i "gedit"*. Из рисунка видно, что наш процесс имеет PID 4507. (рис. 2.6)


```
[sdscripnikova@sdscripnikova ~]$ gedit &
[1] 3465
[sdscripnikova@sdscripnikova ~]$ gedit &
[2] 3497
[1]   Завершён      gedit
[sdscripnikova@sdscripnikova ~]$ ps |grep -i "gedit"
[2]+   Завершён      gedit
[sdscripnikova@sdscripnikova ~]$ pgrep gedit
[sdscripnikova@sdscripnikova ~]$ pidof gedit
[sdscripnikova@sdscripnikova ~]$
```

Рис. 2.6: Определение идентификатора процесса

10. Далее ознакомимся со справкой команды *kill* и используем её для завершения процесса *gedit*. (рис. 2.7),(рис. 2.8)

```
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.

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

Рис. 2.7: Опции команды kill

```
[sdscripnikova@sdscripnikova ~]$ man kill
[sdscripnikova@sdscripnikova ~]$ kill 3497
```

Рис. 2.8: 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. 2.9), (рис. 2.10), (рис. 2.11), (рис. 2.12)

```

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 available on all cur-
    rently mounted file systems is shown. Space is shown in 1K blocks by
    default, unless the environment variable POSIXLY_CORRECT is set, in
    which case 512-byte blocks are used.

    If an argument is the absolute file name of a device node containing a
    mounted file system, df shows the space available on that file system
    rather than on the file system containing the device node. This version
    of df cannot show the space available on unmounted file systems, because
    on most kinds of systems doing so requires very nonportable intimate
    knowledge of file system structures.

OPTIONS
    Manual page df(1), line 1 (press h for help or q to quit)

```

Рис. 2.9: 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 directories.

    Mandatory arguments to long options are mandatory for short options too.

    -0, --null
        end each output line with NUL, not newline

    -a, --all
        write counts for all files, not just directories

    --apparent-size
        print apparent sizes rather than device usage; although the ap-
        parent size is usually smaller, it may be larger due to holes in
        ('sparse') files, internal fragmentation, indirect blocks, and
        Manual page du(1) line 1 (press h for help or q to quit)

```

Рис. 2.10: Опции команды du

```

[sdskripnikova@sdskripnikova ~]$ df
Файловая система 1K-блоков  Использовано  Доступно  Использовано%  Смонтировано в
devtmpfs           4096             0         4096             0% /dev
tmpfs              1008740         13000      995740             2% /dev/shm
tmpfs              403500          1392      402108             1% /run
/dev/sda2          82835456      22776652   57623668            29% /
tmpfs              1008744          24      1008720             1% /tmp
/dev/sda2          82835456      22776652   57623668            29% /home
/dev/sda1          996780         230280     697688            25% /boot
tmpfs              201748           252      201496             1% /run/user/1000
/dev/sr0           62390          62390          0            100% /run/media/sdskripn
ikova/VBox_GAs_6.1.38

```

Рис. 2.11: Команда df

```

12 ./sdskripnikova.github.io/.git/objects
4  ./sdskripnikova.github.io/.git/logs/refs/heads
4  ./sdskripnikova.github.io/.git/logs/refs/remotes/origin
4  ./sdskripnikova.github.io/.git/logs/refs/remotes
8  ./sdskripnikova.github.io/.git/logs/refs
12 ./sdskripnikova.github.io/.git/logs
116 ./sdskripnikova.github.io/.git
116 ./sdskripnikova.github.io
0  ./letters
0  ./memos
0  ./misk
8  ./ski.plases/equipment
0  ./ski.plases/plans
8  ./ski.plases
du: невозможно получить доступ к './play/file.old': Отказано в доступе
0  ./play

```

Рис. 2.12: Команда du

12. Выведем имена всех директорий, которые имеются в домашнем каталоге, предварительно узнаем опции команды *find*. (рис. 2.13), (рис. 2.14)

```
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...] [ex-
    pression]

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 prece-
    dence (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.  If no starting-point is specified, .'
    is assumed.

    If you are using find in an environment where security is important (for
    example if you are using it to search directories that are writable by
    other users), you should read the 'Security Considerations' chapter of
    the findutils documentation, which is called Finding Files and comes
    with findutils.  That document also includes a lot more detail and dis-
    cussion than this manual page, so you may find it a more useful source
    of information.

Manual page find(1), line 1 (press h for help or q to quit)
```

Рис. 2.13: Опции команды find

```
./sdskripnikova.github.io/.git/description
./sdskripnikova.github.io/.git/refs
./sdskripnikova.github.io/.git/refs/heads
./sdskripnikova.github.io/.git/refs/heads/main
./sdskripnikova.github.io/.git/refs/tags
./sdskripnikova.github.io/.git/refs/remotes
./sdskripnikova.github.io/.git/refs/remotes/origin
./sdskripnikova.github.io/.git/refs/remotes/origin/main
./sdskripnikova.github.io/.git/objects
./sdskripnikova.github.io/.git/objects/pack
./sdskripnikova.github.io/.git/objects/info
./sdskripnikova.github.io/.git/objects/e6
./sdskripnikova.github.io/.git/objects/e6/9de29bb2d1d6434b8b29ae775ad8c2e48c5391
./sdskripnikova.github.io/.git/objects/f9
./sdskripnikova.github.io/.git/objects/f9/3e3a1a1525fb5b91020da86e44810c87a2d7bc
./sdskripnikova.github.io/.git/objects/f2
./sdskripnikova.github.io/.git/objects/f2/f7e3702d1595bbc15bd439a391a03c362bc556
./sdskripnikova.github.io/.git/config
```

Рис. 2.14: 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.