

# **Отчёт по лабораторной работе №6**

**Операционные системы**

Попова Елизавета Сергеевна

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	14
4	Ответы на контрольные вопросы	15

## 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 . . . . .	9
2.6	Удаление файла . . . . .	9
2.7	Редактор gedit . . . . .	9
2.8	Определение идентификатора процесса . . . . .	10
2.9	Опции команды kill . . . . .	10
2.10	Завершение процесса . . . . .	10
2.11	Опции команды df . . . . .	11
2.12	Опции команды du . . . . .	11
2.13	Команда df . . . . .	12
2.14	Команда du . . . . .	12
2.15	Опции команды find . . . . .	12
2.16	Выполнение команды . . . . .	13

## **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).

```
[espopova@espopova ~]$ ls -a /etc > file.txt
[espopova@espopova ~]$ ls -a ~ >> file.txt
[espopova@espopova ~]$ cat file.txt
.
..
abrt
adjtime
aliases
alsa
alternatives
anaconda
anthy-unicode.conf
appstream.conf
asound.conf
audit
authselect
avahi
bash_completion.d
bashrc
bindresvport.blacklist
binfmt.d
bluetooth
brlapi.key
brltty
brltty.conf
ceph
chkconfig.d
chromium
chrony.conf
chrony.keys
```

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).

```
Шаблоны
[espopova@espopova ~]$ grep -e '\.conf$' file.txt > conf.txt
[espopova@espopova ~]$ cat conf.txt
anthy-unicode.conf
appstream.conf
asound.conf
brltty.conf
chrony.conf
dleyna-renderer-service.conf
dleyna-server-service.conf
dnsmasq.conf
dracut.conf
fprintd.conf
fuse.conf
host.conf
idmapd.conf
kdump.conf
krb5.conf
ld.so.conf
libaudit.conf
libuser.conf
locale.conf
logrotate.conf
man_db.conf
mke2fs.conf
mttools.conf
nfs.conf
nfsmount.conf
nsswitch.conf
opensc.conf
```

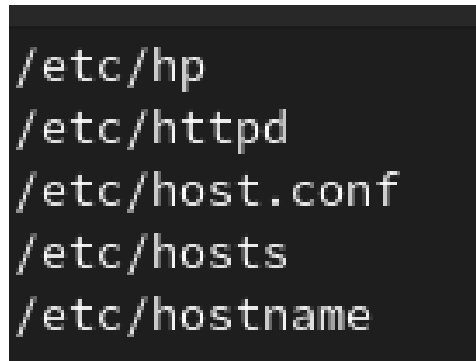
Рис. 2.2: Вывод файлов

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

```
кэтер.conf
[espopova@espopova ~]$ find ~ -maxdepth 1 -name "c*" -print
/home/espopova/conf.txt
[espopova@espopova ~]$ ls ~|c*
bash: conf.txt: команда не найдена...
[espopova@espopova ~]$ ls ~/c*
/home/espopova/conf.txt
[espopova@espopova ~]$ 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 -maxdepth1 -name "h"| less`. (рис. 2.4)



```
/etc/hp
/etc/httpd
/etc/host.conf
/etc/hosts
/etc/hostname
```

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)



```
[esporova@esporova ~]$ find/-name "log">logfile&>>
bash: *: неоднозначное перенаправление
[esporova@esporova ~]$ find / -name "log" > ~/logfile
find: '/boot/loader/entries': Отказано в доступе
find: '/boot/grub2': Отказано в доступе
find: '/boot/lost+found': Отказано в доступе
find: '/boot/efi': Отказано в доступе
find: '/proc/tty/driver': Отказано в доступе
find: '/proc/1/task/1/fd': Отказано в доступе
find: '/proc/1/task/1/fdinfo': Отказано в доступе
find: '/proc/1/task/1/ns': Отказано в доступе
find: '/proc/1/fd': Отказано в доступе
find: '/proc/1/map_files': Отказано в доступе
find: '/proc/1/fdinfo': Отказано в доступе
find: '/proc/1/ns': Отказано в доступе
find: '/proc/2/task/2/fd': Отказано в доступе
find: '/proc/2/task/2/fdinfo': Отказано в доступе
find: '/proc/2/task/2/ns': Отказано в доступе
find: '/proc/2/fd': Отказано в доступе
find: '/proc/2/map_files': Отказано в доступе
find: '/proc/2/fdinfo': Отказано в доступе
find: '/proc/2/ns': Отказано в доступе
find: '/proc/3/task/3/fd': Отказано в доступе
find: '/proc/3/task/3/fdinfo': Отказано в доступе
find: '/proc/3/task/3/ns': Отказано в доступе
find: '/proc/3/fd': Отказано в доступе
find: '/proc/3/map_files': Отказано в доступе
find: '/proc/3/fdinfo': Отказано в доступе
find: '/proc/3/ns': Отказано в доступе
find: '/proc/4/task/4/fd': Отказано в доступе
find: '/proc/4/task/4/fdinfo': Отказано в доступе
find: '/proc/4/task/4/ns': Отказано в доступе
find: '/proc/4/fd': Отказано в доступе
find: '/proc/4/map_files': Отказано в доступе
find: '/proc/4/fdinfo': Отказано в доступе
find: '/proc/4/ns': 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)

```
[esporova@esporova ~]$ rm logfile
[esporova@esporova ~]$ 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)

```
[esporova@esporova ~]$ gedit &
[1] 4013
```

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 4507. (рис. 2.8)

```
[espopova@espopova ~]$ gedit &
[1] 4013
[espopova@espopova ~]$ ps |grep -i "gedit"
[1]+  Завершён      gedit
[espopova@espopova ~]$ pgrep gedit
[espopova@espopova ~]$ pidof gedit
[espopova@espopova ~]$
```

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
NAME
  kill - terminate a process

SYNOPSIS
  kill [-signal|--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 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 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 possible 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:

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

    0
      All processes in the current process group are signaled.

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

    -p
      where p is larger than 1. All processes in process group p are signaled. When an argument of the form '-n' is given, and it is meant to denote a process group, it must be preceded by a '--' option, otherwise it will be taken as the signal to send.

  name
    All processes invoked using this name will be signaled.

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

Рис. 2.9: Опции команды kill

```
[espopova@espopova ~]$ man kill
[espopova@espopova ~]$ man kill
[espopova@espopova ~]$ kill 4013
```

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
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. 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. If no argument is given, df shows the space available on all mounted file systems. df does not show the space available on unmounted file systems, because on most kinds of systems doing so requires very nonportable intimate knowledge of the system.

OPTIONS
    Show information about the file system on which each FILE resides, or all file systems by default.

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

    -a, --all
        include pseudo, duplicate, inaccessible file systems

    -B, --block-size=SIZE
        scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below

    --direct
        show statistics for a file instead of mount point

    -h, --human-readable
        print sizes in powers of 1024 (e.g., 1023M)

    -H, --si
        print sizes in powers of 1000 (e.g., 1.1G)

    -i, --inodes
        list inode information instead of block usage

    -k
        like --block-size=K

    -l, --local
        limit listing to local file systems
  
```

Рис. 2.11: Опции команды df

```

du(1)                                                    User Commands
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 apparent size is usually smaller, it may be larger due to holes in ('')

    -B, --block-size=SIZE
        scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below

    -b, --bytes
        equivalent to '--apparent-size --block-size=1'

    -c, --total
        produce a grand total

    -D, --dereference-args
        dereference only symlinks that are listed on the command line

    -d, --max-depth=N
        print the total for a directory (or file, with --all) only if it is N or fewer levels below the command line argument; --max-depth=0

    --files0-from=F
        summarize device usage of the NUL-terminated file names specified in file F; if F is -, then read names from standard input

    -H
        equivalent to --dereference-args (-D)
  
```

Рис. 2.12: Опции команды du

```

@espopovae@espopova ~$ df
Файловая система  IK-блоков  Использовано  Доступно  Использовано%  Смонтировано в
devtmpfs           4096          0      4096          0% /dev
tmpfs              2005856      0  2005856          0% /dev/shm
tmpfs              802344       1420   800924          1% /run
/dev/sda3          82834432    14541068  65227332         19% /
tmpfs             2005860      20  2005840          1% /tmp
/dev/sda2          82834432    14541068  65227332         19% /home
/dev/sda2          996780      245812   682156         27% /boot
tmpfs             401168      148   401020          1% /run/user/1000
/dev/sr0           51716       51716          0        100% /run/media/espopova/VBox_GAs_7.0.2
@espopovae@espopova ~$

```

Рис. 2.13: Команда df

```

38648 ./work
3900 ./texlive2021/texmf-var/web2c/luahbtex
3900 ./texlive2021/texmf-var/web2c
1556 ./texlive2021/texmf-var/luatex-cache/generic/names
25736 ./texlive2021/texmf-var/luatex-cache/generic/fonts/otl
25736 ./texlive2021/texmf-var/luatex-cache/generic/fonts
27292 ./texlive2021/texmf-var/luatex-cache/generic
27292 ./texlive2021/texmf-var/luatex-cache
30592 ./texlive2021/texmf-var
30592 ./texlive2021
54516 ./bin
4 ./ssh

```

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	<p>This manual page documents the GNU version of <b>find</b>. GNU <b>find</b> 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 <b>and</b> operations, true for <b>or</b>), at which point <b>find</b> moves on to the next file name. If no starting-point is specified, "." is assumed.</p> <p>If you are using <b>find</b> 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 <b>findutils</b> documentation, which is called <b>Finding Files</b> and comes with <b>findutils</b>. That document also includes a lot more detail and discussion than this manual page, so use any <b>find</b> it is a more useful source of information.</p>	
OPTIONS	<p>The <b>-H</b>, <b>-L</b>, and <b>-P</b> options control the treatment of symbolic links. Command-line arguments following these are taken to be names of files or directories to be examined, up to the first argument that begins with "-", or the argument <b>-path</b> is used (but you should probably consider using <b>-xtype</b> instead, anyway).</p> <p>This manual page talks about "options" within the expression tree. These options control the behaviour of <b>find</b> but are specified immediately after the last path name. The file "real" options <b>-H</b>, <b>-L</b>, <b>-P</b>, <b>-D</b> and <b>-d</b> must appear before the first path name, if at all. A double dash "--" could theoretically be used to signal that any remaining arguments are not options, but this does not really work due to the way <b>find</b> determines the end of the following path arguments. If done that by adding until an expression argument comes (which also starts with a "-"). Now, if a path argument would start with a "-", then <b>find</b> would treat it as expression argument instead. Thus, to ensure that all start points are taken as such, and especially to prevent that wildcard patterns expanded by the calling shell are not mistakenly treated as expression arguments, it is generally safer to prefix wildcards or define path names with either <b>./</b> or to use absolute path names starting with <b>/</b>. Alternatively, it is generally safe though non-portable to use the GNU option <b>-find-root</b> to pass arbitrary starting points to <b>find</b>.</p> <p><b>-P</b> Never follow symbolic links. This is the default behaviour. When <b>find</b> examines or prints information about files, and the file is a symbolic link, the information used shall be taken from the properties of the symbolic link itself.</p> <p><b>-L</b> Follow symbolic links. When <b>find</b> examines or prints information about files, the information used shall be taken from the file to which the link points, not from the link itself (unless it is a broken symbolic link or <b>find</b> is unable to examine the file to which the link points). Use of this option implies <b>-xtype</b>. If you later use the <b>-d</b> option, <b>-xtype</b> will still be in effect. If <b>-L</b> is in effect and <b>find</b> discovers a symbolic link to a subdirectory during the search, the subdirectory pointed to by the symbolic link will be searched.</p> <p>When the <b>-L</b> option is in effect, the <b>-type</b> predicate will always match against the type of the file that a symbolic link points to rather than the link itself (unless the symbolic link is broken). Actions that can cause symbolic links to become broken while <b>find</b> is executing (for example <b>-delete</b>) can give rise to confusing behaviour. Using <b>-L</b> causes the <b>-lname</b> and <b>-flname</b> predicates always to return false.</p> <p><b>-H</b> Do not follow symbolic links, except while processing the command line arguments. When <b>find</b> examines or prints information about files, the information used shall be taken from the properties of the symbolic link itself. The only exception to this behaviour is when a file specified on the command line is a symbolic link, and the link can be resolved. For that situation, the information used is taken from whatever the link points to (that is, the link is followed). The information about the link itself is used as a fallback (if the file pointed to by the symbolic link cannot be examined. If <b>-H</b> is in effect and one of the paths specified on the command line is a symbolic link to a directory, the contents of that directory will be examined (though of course <b>execdepth</b> would prevent this).</p> <p>If more than one of <b>-H</b>, <b>-L</b> and <b>-P</b> is specified, each overrides the others; the last one appearing on the command line takes effect. Since it is the default, the <b>-P</b> option should be considered to be in effect unless either <b>-H</b></p>	
Manual page find(1)	Use <b>press h</b> for help or <b>q</b> to quit.	

Рис. 2.15: Опции команды find

```
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/pts56f.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/pts56f.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/ptm55f.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/ptm55f.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/ptm75f.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/ptm75f.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/latinmodern-math.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/latinmodern-math.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/lmroman10-regular.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/lmroman10-regular.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/lmsans10-regular.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/lmsans10-regular.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/lmsans10-bold.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/lmsans10-bold.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-light.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-light.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-lightitalic.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-lightitalic.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-regular.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-regular.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-italic.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firasans-italic.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-regular.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-regular.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-bold.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-bold.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-medium.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-medium.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-boldoblique.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-boldoblique.luc
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-oblique.lua
./texlive2021/texmf-var/luatex-cache/generic/fonts/otl/firamono-oblique.luc
./bin
./bin/hugo
./ssh
./ssh/known_hosts
./vboxclient-clipboard.pid
./vboxclient-seamless.pid
./vboxclient-draganddrop.pid
./vboxclient-vmsvga-session-tty2.pid
```

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+;`;
  - `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.