

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Операционная система Linux»

Управление процессами в Linux

Студент

Посаднев В.В.

Группа АС-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Оглавление

Цель работы	3
Задание кафедры.....	4
Ход работы.....	5
Задание 1	5
Задание 2	12
Задание 3	14
Задание 4	17
Задание 5	31
Вывод.....	32

Цель работы

Приобрести опыт и навыки управления процессами в операционной системе Linux.

Задание кафедры

1. Повторить команды `cat`, `head`, `tail`, `more`, `less`, `grep`, `find`
2. Разобраться с понятиями конвейер, перенаправление ввода-вывода.
3. Ознакомиться с информацией из рекомендованных источников и других про конвейеризации.
4. Повторить назначение прав доступа. Команды `chmod`, `chown`.
5. Ознакомиться с информацией по теме процессы, посмотреть и опробовать примеры наиболее распространенных команд, изучить возможность запуска процессов в `supervisor`.
6. Изучить возможность автоматического запуска программ по расписанию.

Ход работы

Задание 1

Команда `cat` выводит информацию из указанного файла на экран. Для использования данной команды создадим файлы, используя команду `touch`, и заполним его случайным содержанием, используя текстовый редактор `vim`. После инициализации данного файла выведем его содержимое с помощью команды `cat cat_file.txt`, выполнение данной команды изображено на рисунке 1.



```
elemabor@ubuntu:~/lab3$ cat cat_file.txt
1312rdfgvfdx
trgdsfvsdrgtretwres
dfdhg jfgydgrdvug
chmvhgdxvstgrdtj
yfg jnbv of dsfgbf nh jyug
itrtydgdrcfxvfnh jmutyh
gfdgbhc juki ytf defsvuoyitf vdgbnh jkio
uylhtgf hukioy tgrf sdvgbhuy i8tf guuythgrf ewarfgth
fyf hfytrdfgvhyu7tf hgbhfyt
elemabor@ubuntu:~/lab3$ _
```

Рисунок 1 – Выполнение команды `cat`

Следующей командой, которую необходимо рассмотреть, является `head`. Данная команда выводит строки указанного файла начиная сначала. Для применения данной команды воспользуемся следующей комбинацией: `head -n 3 cat_file.txt`. Пример выполнения данной команды изображен на рисунке 2. Данная команда принимает следующий набор аргументов:

- с – указывать количество текста не в строках, а в байтовом представлении;

- n – вывести определенное количество строк сначала файла, изначально данный аргумент равен 10;

- q – не печатать название файла;

- v – выводить название файла;

- z – разделение между новыми строками.

```

elemabor@ubuntu:~/lab3$ head --help
Usage: head [OPTION]... [FILE]...
Print the first 10 lines of each FILE to standard output.
With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.
-c, --bytes=[-]NUM      print the first NUM bytes of each file;
                        with the leading '-', print all but the last
                        NUM bytes of each file
-n, --lines=[-]NUM      print the first NUM lines instead of the first 10;
                        with the leading '-', print all but the last
                        NUM lines of each file
-q, --quiet, --silent   never print headers giving file names
-v, --verbose           always print headers giving file names
-z, --zero-terminated   line delimiter is NUL, not newline
--help                 display this help and exit
--version              output version information and exit

NUM may have a multiplier suffix:
b 512, kB 1000, K 1024, MB 1000*1000, M 1024*1024,
GB 1000*1000*1000, G 1024*1024*1024, and so on for T, P, E, Z, Y.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/head>
or available locally via: info '(coreutils) head invocation'
elemabor@ubuntu:~/lab3$ head -n 3 cat_file.txt
1312rdfgvfdx
trgdsfvsdrgtretures
dfdhgjfydgrdgvg
elemabor@ubuntu:~/lab3$ _

```

Рисунок 2 – Выполнение команды head с выводом первых трех строк

Следующая команда необходимая для повторения это tail. Данная команда похожа на команду cat и head. В сравнении с head она отличается тем, что идет не с начала файла, а с конца. Для применения данной команды воспользуемся следующим: *tail -n 3 cat_file.txt*. После выполнения данной команды будет выведены 3 последние строки файла cat_file.txt. Как можно заметить аргументы для выполнения данной команды аналогичны команде head. Результат выполнения данной команды изображено на рисунке 3.

```

--max-unchanged-stats=N      with --follow=name, reopen a FILE which has not
                             changed size after N (default 5) iterations
                             to see if it has been unlinked or renamed
                             (this is the usual case of rotated log files);
                             with inotify, this option is rarely useful
--pid=PID                    with -f, terminate after process ID, PID dies
-q, --quiet, --silent        never output headers giving file names
--retry                       keep trying to open a file if it is inaccessible
-s, --sleep-interval=N       with -f, sleep for approximately N seconds
                             (default 1.0) between iterations;
                             with inotify and --pid=P, check process P at
                             least once every N seconds
-v, --verbose                always output headers giving file names
-z, --zero-terminated        line delimiter is NUL, not newline
--help                       display this help and exit
--version                     output version information and exit

NUM may have a multiplier suffix:
b 512, kB 1000, K 1024, MB 1000*1000, M 1024*1024,
GB 1000*1000*1000, G 1024*1024*1024, and so on for T, P, E, Z, Y.

With --follow (-f), tail defaults to following the file descriptor, which
means that even if a tail'ed file is renamed, tail will continue to track
its end. This default behavior is not desirable when you really want to
track the actual name of the file, not the file descriptor (e.g., log
rotation). Use --follow=name in that case. That causes tail to track the
named file in a way that accommodates renaming, removal and creation.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/tail>
or available locally via: info '(coreutils) tail invocation'
elemabor@ubuntu:~/lab3$ tail -n 3 cat_file.txt
gfdgbhcjukiytfdesvuoyitfvdgbnhjkio
uyhtgfhukioygrfsdvgbhuyi8tfguuythgrfewarfgyth
fyfhhfytrdfgvhyu7tfhgbhfyt
elemabor@ubuntu:~/lab3$ _

```

Рисунок 3 – Пример выполнения команды `tail` для вывода трех последних строк

Рассмотрим команды `more` и `less`. Данные утилиты предназначена для постраничного просмотра содержимого файлов. В сравнении с `more`, команда `less` имеет больший функционал (об этом можно говорить, посмотрев на помощника по данной команды). Так, например, на рисунке 4 изображена вся вспомогательная справка по команде `more`. А на рисунке 5 изображена только малая часть вспомогательной справки по команде `less`.

Выполним соответствующие команды: `more cat_file.txt` и `less cat_file.txt`. Для более наглядного просмотра дополним текстовый файл дополнительными текстовыми строчками. Пример выполнения команды с `more` изображен на рисунке 6, а с использованием `less` на рисунке 7. Одним из преимуществ команды `less` заключается в том, что во время такого просмотра файла мы можем производить поиск по содержимому файла.

```

elemabor@ubuntu:~/lab3$ more --help

Usage:
  more [options] <file>...

A file perusal filter for CRT viewing.

Options:
  -d          display help instead of ringing bell
  -f          count logical rather than screen lines
  -l          suppress pause after form feed
  -c          do not scroll, display text and clean line ends
  -p          do not scroll, clean screen and display text
  -s          squeeze multiple blank lines into one
  -u          suppress underlining
  -<number>   the number of lines per screenful
  +<number>   display file beginning from line number
  +/<string>  display file beginning from search string match

      --help    display this help
      -V, --version  display version

For more details see more(1).
elemabor@ubuntu:~/lab3$

```

Рисунок 4 – Содержимое документации по команде more

```

                                SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, N.
Notes in parentheses indicate the behavior if N is given.
A key preceded by a caret indicates the Ctrl key; thus ^K is ctrl-K.

h H          Display this help.
q :q Q :Q ZZ  Exit.
-----

                                MOVING

e ^E j ^N CR  * Forward one line (or N lines).
y ^Y k ^K ^P  * Backward one line (or N lines).
f ^F ^V SPACE * Forward one window (or N lines).
b ^B ESC-v    * Backward one window (or N lines).
z          * Forward one window (and set window to N).
w          * Backward one window (and set window to N).
ESC-SPACE  * Forward one window, but don't stop at end-of-file.
d ^D          * Forward one half-window (and set half-window to N).
u ^U          * Backward one half-window (and set half-window to N).
ESC-) RightArrow * Right one half screen width (or N positions).
ESC-( LeftArrow  * Left one half screen width (or N positions).
ESC-} ^RightArrow Right to last column displayed.
ESC-{ ^LeftArrow  Left to first column.
F          Forward forever; like "tail -f".
ESC-F      Like F but stop when search pattern is found.
r ^R ^L      Repaint screen.
R          Repaint screen, discarding buffered input.
-----

Default "window" is the screen height.
Default "half-window" is half of the screen height.
-----

                                SEARCHING

HELP -- Press RETURN for more, or q when done_

```

Рисунок 5 – Часть содержимого документации по команде less


```

gfdgbhc.juki ytfdefsvuoyitfvdgbnh.jkio
uyhtgf.hukioytrgrfsdvgbhuyi8tfguuythgrfearfgth
fyfhf ytrdfgvhyu7tfhgbbfyt
fzdr tghyugihkoluhfrdsfhyukitfrdsfrhyjug
gfdzxhg.jkigbfhng.juki.jhnbgbf\b\uyfhgffb
hfbtyf.jyft hfgdgb
tyfhng.jugy.jfhdgfsdfftgh.juikug.jyft r
dfgh.juki ytrgdfgh.jkiuytrgfsdfgh.jklopiuytrgdfsa
fghuyiytrsedadfggh.jkiouytrewdsfgh.jkiouythrfeadse
ryuiuo.kjhytfgf edrgthythgf vdc
hjuguyuyuyyhg tfvrdcfrgthuykiu.jytyrdnnmhgkkgg.ju
gh.jnyfhg.jdhsfg.jsdhfgosdfgk.jf lshdg
gfshdubhgds.iughlosdkngoishdiugbea
gbfdshiovf.sndgiouvsbdi gh
foasieof.ius.jfohdiougohav
rouaesof.hsdoi ghosidngovidshrozshgdi bg
foisua.jiofhaoseifnvgodanbguvisdhiohasiofb
trgdsfvsdrgtretwres
dfdhg.jfygdgrdgvg
chnv hgdxf.vstgrdt.j
yfg.jnbf.vf dsfgbf nh.jyug
itrrydgdrfxvf bnh.jmutyh
gfdgbhc.juki ytfdefsvuoyitfvdgbnh.jkio
uyhtgf.hukioytrgrfsdvgbhuyi8tfguuythgrfearfgth
fyfhf ytrdfgvhyu7tfhgbbfyt
fzdr tghyugihkoluhfrdsfhyukitfrdsfrhyjug
gfdzxhg.jkigbfhng.juki.jhnbgbf\b\uyfhgffb
hfbtyf.jyft hfgdgb
tyfhng.jugy.jfhdgfsdfftgh.juikug.jyft r
dfgh.juki ytrgdfgh.jkiuytrgfsdfgh.jklopiuytrgdfsa
--More-- (9%)

```

Рисунок 6 – Пример выполнения команды more

```

1312rdfgvf dx
trgdsfvsdrgtretwres
dfdhg.jfygdgrdgvg
chnv hgdxf.vstgrdt.j
yfg.jnbf.vf dsfgbf nh.jyug
itrrydgdrfxvf bnh.jmutyh
gfdgbhc.juki ytfdefsvuoyitfvdgbnh.jkio
uyhtgf.hukioytrgrfsdvgbhuyi8tfguuythgrfearfgth
fyfhf ytrdfgvhyu7tfhgbbfyt
fzdr tghyugihkoluhfrdsfhyukitfrdsfrhyjug
gfdzxhg.jkigbfhng.juki.jhnbgbf\b\uyfhgffb
hfbtyf.jyft hfgdgb
tyfhng.jugy.jfhdgfsdfftgh.juikug.jyft r
dfgh.juki ytrgdfgh.jkiuytrgfsdfgh.jklopiuytrgdfsa
fghuyiytrsedadfggh.jkiouytrewdsfgh.jkiouythrfeadse
ryuiuo.kjhytfgf edrgthythgf vdc
hjuguyuyuyyhg tfvrdcfrgthuykiu.jytyrdnnmhgkkgg.ju
gh.jnyfhg.jdhsfg.jsdhfgosdfgk.jf lshdg
gfshdubhgds.iughlosdkngoishdiugbea
gbfdshiovf.sndgiouvsbdi gh
foasieof.ius.jfohdiougohav
rouaesof.hsdoi ghosidngovidshrozshgdi bg
foisua.jiofhaoseifnvgodanbguvisdhiohasiofb
trgdsfvsdrgtretwres
dfdhg.jfygdgrdgvg
chnv hgdxf.vstgrdt.j
yfg.jnbf.vf dsfgbf nh.jyug
itrrydgdrfxvf bnh.jmutyh
gfdgbhc.juki ytfdefsvuoyitfvdgbnh.jkio
uyhtgf.hukioytrgrfsdvgbhuyi8tfguuythgrfearfgth
fyfhf ytrdfgvhyu7tfhgbbfyt
fzdr tghyugihkoluhfrdsfhyukitfrdsfrhyjug
gfdzxhg.jkigbfhng.juki.jhnbgbf\b\uyfhgffb
hfbtyf.jyft hfgdgb
tyfhng.jugy.jfhdgfsdfftgh.juikug.jyft r
dfgh.juki ytrgdfgh.jkiuytrgfsdfgh.jklopiuytrgdfsa
cat_file.txt_

```

Рисунок 7 – Пример выполнения команды less

Последними двумя командами, которыми необходимо воспользоваться, являются: `grep` и `find`. Утилита `grep` предназначена для фильтрации выходных данных или для поиска внутри файла. Для примера использования напомним команду, которая будет искать все строчки с символом «z». Полученная команда будет выглядеть следующим образом: `grep z cat_file.txt`. Примером выполнения данной команды будет служить рисунок 8, на данном рисунке красным цветом выделены найденные элементы. В свою очередь утилита `find` может производить поиск по всем файлам. Для примера выполним следующее: выйдем из текущей директории с текстовым файлом на один каталог выше и произведем поиск данного файла. Для этого выполним команду `find -name "cat_file.txt"`. После поиска будет выведен путь относительно текущей папки. Пример выполнения данной команды изображен на рисунке 9.

```
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
fzdr tghyug ihko luhfrdsf hyukitfrdsfrhy.jug  
gfdzxhg jkigbfhng juki jhnbgb\uyf hgffb  
rouaesoi fhsdo ighos idngovidshrozshgdibg  
elemabor@ubuntu:~/lab3$ _
```

Рисунок 8 – Пример выполнения команды `grep`

```
elemabor@ubuntu:~/lab3$ ls -l
total 12
-rw-rw-r-- 1 elemabor elemabor 11981 Nov 10 12:29 cat_file.txt
elemabor@ubuntu:~/lab3$ cd ..
elemabor@ubuntu:~$ find -name "cat_file.txt"
./lab3/cat_file.txt
elemabor@ubuntu:~$ _
```

Рисунок 9 – Пример выполнения команды find

Задание 2

В данном задании необходимо разобраться с понятиями конвейер и перенаправление ввода-вывода.

Рассмотрим сначала перенаправление ввода вывода, для этого перенаправим результат выполнения команды `head -n 3 cat_file.txt` в файл `task2.txt`. Также перенаправим содержимое выполнения команды `tail -n 3 cat_file.txt` в файл `task2.txt` с записью в конец файла. Для выполнения данных операций нам необходимо выполнить следующие команды: `head -n 3 cat_file.txt > task2.txt` и `tail -n 3 cat_file.txt >> task2.txt`. Рассмотрим данные команды более подробнее. В первой команде используется оператор перенаправления “>”, он перенаправляет поток вывода в файл (файл будет перезаписан, либо создан), тем самым в файл `task2.txt` мы записываем 3 первых строки файла `cat_file.txt`. Во второй команде используется оператор перенаправления “>>”, данный оператор перенаправляет поток вывода в файл с записью данных в конец, либо создание нового файла. Для перенаправления потока ввода выполним команду `sort < task2.txt`. Пример выполнения данных команд изображен на рисунке 10.

```
elemabor@ubuntu:~/lab3$ ls -l
total 12
-rw-rw-r-- 1 elemabor elemabor 11981 Nov 10 12:29 cat_file.txt
elemabor@ubuntu:~/lab3$ head -n 3 cat_file.txt > task2.txt
elemabor@ubuntu:~/lab3$ cat task2.txt
1312rdfgvfdx
trgdsfvsdrgtretwres
dfd hg jfg ydgrdgvg
elemabor@ubuntu:~/lab3$ tail -n 3 cat_file.txt >> task2.txt
elemabor@ubuntu:~/lab3$ cat task2.txt
1312rdfgvfdx
trgdsfvsdrgtretwres
dfd hg jfg ydgrdgvg
foasieof ius jfohdiougohav
rouaesoi fhsdoi ghosidngovidshrozshydbg
foisua jiofhaoseifnvgo danbg uvisdhiohasiofb
elemabor@ubuntu:~/lab3$ ls -l
total 16
-rw-rw-r-- 1 elemabor elemabor 11981 Nov 10 12:29 cat_file.txt
-rw-rw-r-- 1 elemabor elemabor 156 Nov 10 13:16 task2.txt
elemabor@ubuntu:~/lab3$ sort < task2.txt
1312rdfgvfdx
dfd hg jfg ydgrdgvg
foasieof ius jfohdiougohav
foisua jiofhaoseifnvgo danbg uvisdhiohasiofb
rouaesoi fhsdoi ghosidngovidshrozshydbg
trgdsfvsdrgtretwres
elemabor@ubuntu:~/lab3$ _
```

Рисунок 10 – Пример перенаправления ввода/вывода

Теперь рассмотрим конвейер. Конвейер – некоторое множество процессов, для которых выполняется следующее: то, что выводит на поток стандартного вывода предыдущий процесс, попадает в поток стандартного ввода следующего процесса. Тем самым, нам не нужно хранить промежуточные значения в каком-то файле. Попробуем выполним следующую операцию: получим значения первых трех строк файла `cat_file.txt` с помощью команды `head -n 3 cat_file.txt`, потом с помощью команды `grep [r]d` найдем все подстроки, которые содержат `rd`, и отсортируем их по убыванию с помощью команды `sort -r`. В итоге у нас получится следующая команда: `head -n 3 cat_file.txt | grep [r]d | sort -r`. Выполним каждую команду по порядку и общую, для более наглядного результата. Данный пример изображен на рисунке 11.

```
elemabor@ubuntu:~/lab3$ head -n 3 cat_file.txt
1312rdfgvfdx
trgdsfvsdrgtretwres
dfdhgjfggydgrdgvg
elemabor@ubuntu:~/lab3$ head -n 3 cat_file.txt | grep [r]d
1312rdfgvfdx
dfdhgjfggydgrdgvg
elemabor@ubuntu:~/lab3$ head -n 3 cat_file.txt | grep [r]d | sort -r
dfdhgjfggydgrdgvg
1312rdfgvfdx
elemabor@ubuntu:~/lab3$ _
```

Рисунок 11 – Пример выполнения конвейера

Задание 3

Повторим назначение прав доступа. Для этого используются следующие команды: `chmod` и `chown`.

Команда `chmod` – изменяет права доступа к файлу. Для использования этой команды необходимо иметь права владельца файла или права `root`. Пример команды следующий: `chmod user mode filename`. `User` – пользователь для которого будет выполнена данная команда, `mode` – права доступа, устанавливаемые на файл (символьный или абсолютный), `filename` – имя файла для которого необходимо изменить права доступа. Для демонстрации работы данной команды загрузимся под ранее созданным пользователем `user`, перейдем в каталог с необходимым файлом созданным пользователем `elemabor` (изображено на рисунке 12) и попробуем изменить его. В результате такой попытке мы получим сообщение изображенное на рисунке 13.

```
Ubuntu 18.04.5 LTS ubuntu tty2
ubuntu login: user
Password:
Last login: Wed Nov 11 11:40:38 PST 2020 on tty2
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-118-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch
New release '20.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

$ cd ..
$ cd elemabor
$ cd lab3
$ ls -l
total 20
-rw-rw-r-- 1 elemabor elemabor 11981 Nov 10 12:29 cat_file.txt
-rw-rw-r-- 1 elemabor elemabor 156 Nov 10 13:16 task2.txt
-rw-rw-r-- 1 elemabor elemabor 27 Nov 11 11:48 task3.txt
$
```

Рисунок 12 – Авторизация под пользователем `user` и переход в необходимый каталог


```
"task3.txt" 2L, 52C written
$ cat task3.txt
sample text for task3_lab3
added new text from user
$
```

Рисунок 15 – Пример записи данных в файл от пользователя user

Теперь рассмотрим команду `chown`. Команда `chown` позволяет сменить владельца файла. Для использования данной команды необходимо иметь права владельца текущего файла или права `root` пользователя. Синтаксис данной команды следующий: *`chown username:groupname filename`*. Username – имя пользователя (новый владелец файла), groupname – имя группы (нового владельца файла), filename – имя файла владельца которого необходимо изменить. Для примера данной команды теперь изменим владельца файла `task3.txt` на пользователя `user`. Это можно сделать следующей командой: `sudo chown user task3.txt`. Пример выполнения данной команды изображен на рисунке 16.

```
elemabor@ubuntu:~/lab3$ ls -l
total 20
-rw-rw-r-- 1 elemabor elemabor 11981 Nov 10 12:29 cat_file.txt
-rw-rw-r-- 1 elemabor elemabor 156 Nov 10 13:16 task2.txt
-rw-rw-rw- 1 elemabor elemabor 52 Nov 11 12:22 task3.txt
elemabor@ubuntu:~/lab3$ sudo chown user task3.txt
[sudo] password for elemabor:
elemabor@ubuntu:~/lab3$ ls -l
total 20
-rw-rw-r-- 1 elemabor elemabor 11981 Nov 10 12:29 cat_file.txt
-rw-rw-r-- 1 elemabor elemabor 156 Nov 10 13:16 task2.txt
-rw-rw-rw- 1 user elemabor 52 Nov 11 12:22 task3.txt
elemabor@ubuntu:~/lab3$ _
```

Рисунок 16 – Изменение владельца файла `task3.txt` на пользователя `user`

Задание 4

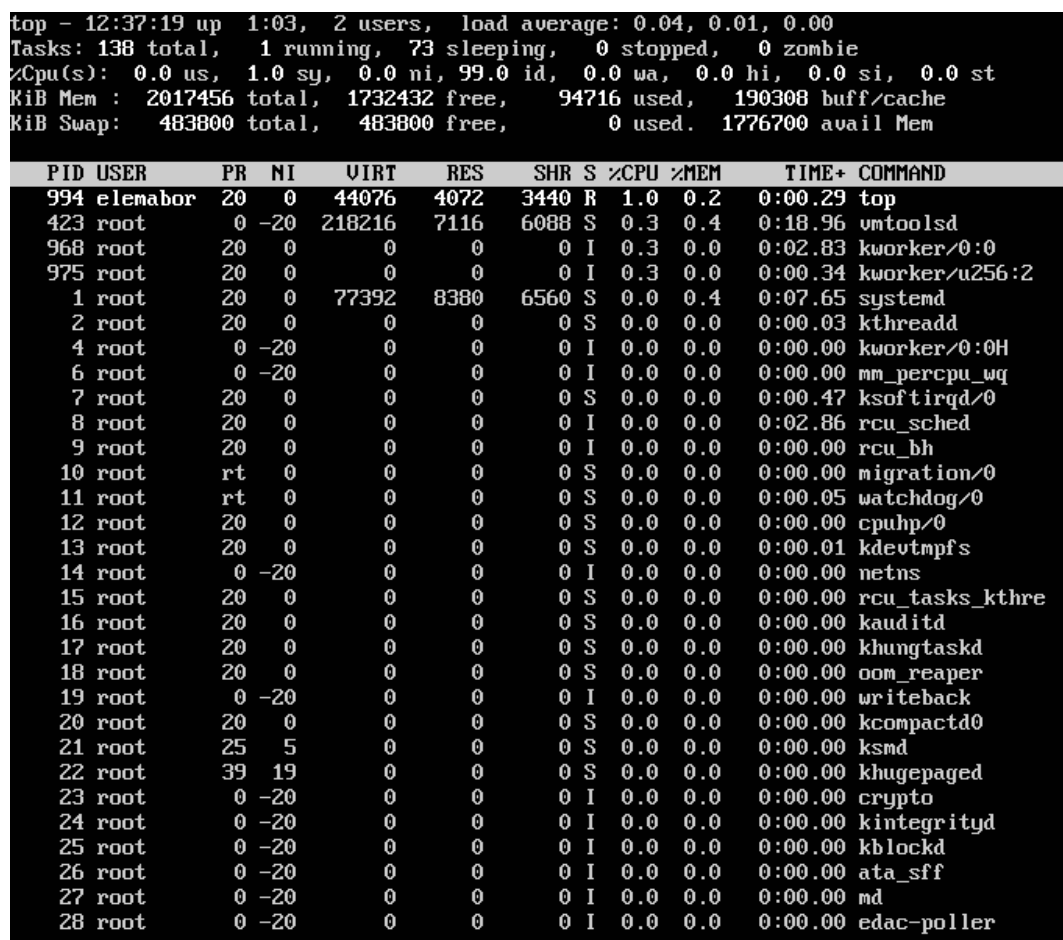
Ознакомимся с информацией по теме процессы, посмотрим и опробуем примеры наиболее распространенных команд.

Дадим определение процессу. Процесс – понятие совокупности программного кода и данных, загруженных в память ЭВМ. Процесс – это не запущенная программа или команда, так как приложение может создавать несколько процессов одновременно. Код процесса не обязательно должен выполнять в текущий момент времени, так как процесс может находиться в спящем состоянии.

Рассмотрим примеры наиболее распространенных команд на практике.

1) Отображение всех процессов

В данном примере с помощью команды *top* мы сможем увидеть информацию о задачах, памяти, нагрузки на процессор и т.д. Пример выполнения данной команды изображен на рисунке 16.



```
top - 12:37:19 up 1:03, 2 users, load average: 0.04, 0.01, 0.00
Tasks: 138 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 1.0 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1732432 free, 94716 used, 190308 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used, 1776700 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
994	elemabor	20	0	44076	4072	3440	R	1.0	0.2	0:00.29	top
423	root	0	-20	218216	7116	6088	S	0.3	0.4	0:18.96	umtoolsd
968	root	20	0	0	0	0	I	0.3	0.0	0:02.83	kworker/0:0
975	root	20	0	0	0	0	I	0.3	0.0	0:00.34	kworker/u256:2
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.65	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.47	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:02.86	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.05	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
21	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
22	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	crypto
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller

Рисунок 16 – Пример использования команды *top*

2) Сортировка информации полученной командой *top* по одному из полей

Для сортировки информации полученной командой *top* по какому-либо из полей необходимо выполнить команду *top* и нажать комбинацию клавиш *SHIFT+F*. После этого перед пользователем будет список полей по которым возможно выполнить сортировку. Выполним сортировку по PID, для этого выберем соответствующий элемент и выберем его нажатием кнопки «s». Пример данного окна изображен на рисунке 17. После выбора поля выйдем из данного меню нажатием кнопки «q». Отсортированные значение приведены на рисунке 18.

```
Fields Management for window 1:Def, whose current sort field is PID
Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
'd' or <Space> toggles display, 's' sets sort. Use 'q' or <Esc> to end!

* PID      = Process Id
* USER     = Effective User Name
* PR       = Priority
* NI       = Nice Value
* VIRT     = Virtual Image (KiB)
* RES     = Resident Size (KiB)
* SHR     = Shared Memory (KiB)
* S       = Process Status
* %CPU    = CPU Usage
* %MEM    = Memory Usage (RES)
* TIME+   = CPU Time, hundredths
* COMMAND = Command Name/Line
PPID     = Parent Process pid
UID      = Effective User Id
RUID     = Real User Id
RUSER    = Real User Name
SUID     = Saved User Id
SUSER    = Saved User Name
GID      = Group Id
GROUP    = Group Name
PGRP     = Process Group Id
TTY      = Controlling Tty
TPGID    = Tty Process Grp Id
SID      = Session Id
nTH      = Number of Threads
P        = Last Used Cpu (SMP)
TIME     = CPU Time
SWAP     = Swapped Size (KiB)
CODE     = Code Size (KiB)
DATA     = Data+Stack (KiB)
nMaj     = Major Page Faults
nMin     = Minor Page Faults
nDRT     = Dirty Pages Count

WCHAN    = Sleeping in Function
Flags    = Task Flags <sched.h>
CGROUPS  = Control Groups
SUPGIDS  = Supp Groups IDs
SUPGRPS  = Supp Groups Names
TGID     = Thread Group Id
OOMa     = OOMEM Adjustment
OOMs     = OOMEM Score current
ENVIRON  = Environment vars
vMj      = Major Faults delta
vMn      = Minor Faults delta
USED     = Res+Swap Size (KiB)
nsIPC    = IPC namespace Inode
nsMNT    = MNT namespace Inode
nsNET    = NET namespace Inode
nsPID    = PID namespace Inode
nsUSER   = USER namespace Inode
nsUTS    = UTS namespace Inode
LXC      = LXC container name
RSan     = RES Anonymous (KiB)
RSfd     = RES File-based (KiB)
RSlk     = RES Locked (KiB)
RSSh     = RES Shared (KiB)
CGNAME   = Control Group name
```

Рисунок 17 – Выбор поля для сортировки

```
top - 12:44:50 up 1:11, 2 users, load average: 0.00, 0.02, 0.00
Tasks: 149 total, 1 running, 84 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.3 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731808 free, 94708 used, 190940 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1776404 avail Mem
```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
995	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:1
994	elemabor	20	0	44188	4200	3504	R	0.1	0.2	0:06.28	top
992	root	20	0	0	0	0	I	0.0	0.0	0:00.17	kworker/u256:1
975	root	20	0	0	0	0	I	0.1	0.0	0:00.57	kworker/u256:2
968	root	20	0	0	0	0	I	0.0	0.0	0:03.51	kworker/0:0
930	user	20	0	4628	1728	1612	S	0.0	0.1	0:00.04	sh
926	user	20	0	111372	2024	28	S	0.0	0.1	0:00.00	(sd-pam)
925	user	20	0	76560	6972	6060	S	0.0	0.3	0:00.08	systemd
889	root	20	0	78764	3744	3176	S	0.0	0.2	0:00.10	login
863	elemabor	20	0	22484	5180	3676	S	0.0	0.3	0:00.86	bash
859	elemabor	20	0	111372	2008	28	S	0.0	0.1	0:00.00	(sd-pam)
858	elemabor	20	0	76560	7084	6172	S	0.0	0.4	0:00.10	systemd
821	root	20	0	78760	3776	3212	S	0.0	0.2	0:00.03	login
529	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/u257:1
485	root	0	-20	218216	7116	6088	S	0.0	0.4	0:00.00	gmain
481	root	20	0	170384	17160	9436	S	0.0	0.9	0:00.00	gmain
448	syslog	20	0	263036	4684	3604	S	0.0	0.2	0:00.04	rs:main Q:Reg
447	syslog	20	0	263036	4684	3604	S	0.0	0.2	0:00.01	in:imklog
446	syslog	20	0	263036	4684	3604	S	0.0	0.2	0:00.02	in:inmuxsock
444	syslog	20	0	263036	4684	3604	S	0.0	0.2	0:00.04	rsyslogd
443	root	20	0	70632	6136	5348	S	0.0	0.3	0:00.60	systemd-logind
442	root	20	0	170384	17160	9436	S	0.0	0.9	0:00.62	networkd-dispat
441	root	20	0	287548	6892	6000	S	0.0	0.3	0:00.00	gdbus
436	message+	20	0	49928	4264	3696	S	0.0	0.2	0:00.30	dbus-daemon
435	root	20	0	287548	6892	6000	S	0.0	0.3	0:00.36	gmain
431	root	20	0	31320	3316	3036	S	0.0	0.2	0:00.04	cron
428	root	20	0	287548	6892	6000	S	0.0	0.3	0:00.20	accounts-daemon
427	systemd+	20	0	141956	3380	2844	S	0.0	0.2	0:00.01	sd-resolve
426	root	0	-20	218216	7116	6088	S	0.0	0.4	0:00.55	HangDetector
423	root	0	-20	218216	7116	6088	S	0.5	0.4	0:20.67	umtolsd

Рисунок 18 – Значения отсортированные по PID

3) Вывод информации всех процессов какого-либо пользователя

Для вывода процессов какого-либо конкретного пользователя необходимо выполнить следующую команду: *top -u username*. Username – имя конкретного пользователя. Для примера выполним данную команду для пользователя elemabor. Пример выполнения такой команды изображен на рисунке 19.

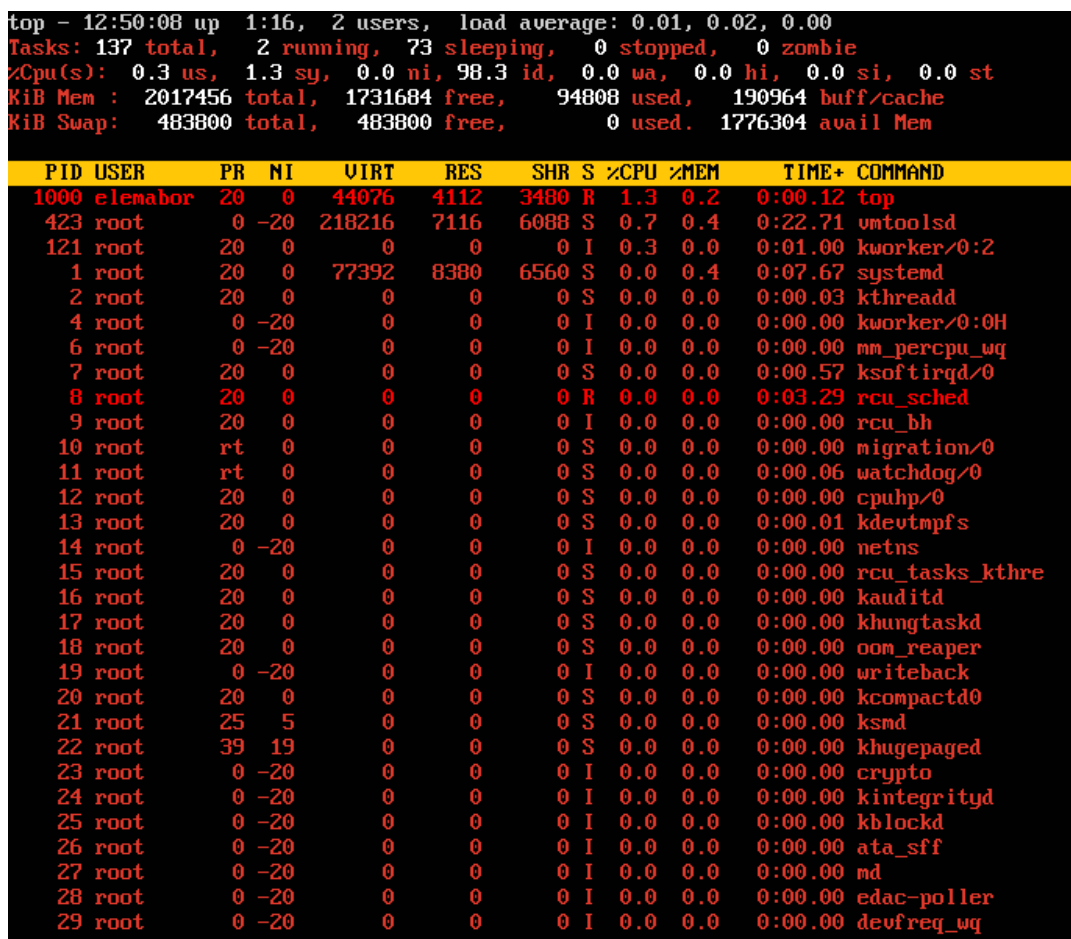
```
top - 12:47:08 up 1:13, 2 users, load average: 0.02, 0.02, 0.00
Tasks: 138 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 1.0 sy, 0.0 ni, 98.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731808 free, 94708 used, 190940 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1776404 avail Mem
```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
998	elemabor	20	0	44080	4044	3408	R	12.5	0.2	0:00.14	top
858	elemabor	20	0	76560	7084	6172	S	0.0	0.4	0:00.10	systemd
859	elemabor	20	0	111372	2008	28	S	0.0	0.1	0:00.00	(sd-pam)
863	elemabor	20	0	22484	5180	3676	S	0.0	0.3	0:00.90	bash

Рисунок 19 – Пример получения информации о процессах пользователя elemabor

4) Выделение выполняющихся процессов

Для выделения процессов выполняющихся в данный момент необходимо выполнить команду *top* и нажать на клавишу «z». Тем самым будут выделены красным цветом выполняющиеся процессы. Пример выполнения такой команды изображен на рисунке 20.



```
top - 12:50:08 up 1:16, 2 users, load average: 0.01, 0.02, 0.00
Tasks: 137 total, 2 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 1.3 sy, 0.0 ni, 98.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731684 free, 94808 used, 190964 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used, 1776304 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1000	elemabor	20	0	44076	4112	3480	R	1.3	0.2	0:00.12	top
423	root	0	-20	218216	7116	6088	S	0.7	0.4	0:22.71	umtoolsd
121	root	20	0	0	0	0	I	0.3	0.0	0:01.00	kworker/0:2
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.67	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.57	ksoftirqd/0
8	root	20	0	0	0	0	R	0.0	0.0	0:03.29	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.06	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
21	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
22	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	crypto
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq

Рисунок 20 – Пример использования команды *top* для выделения выполняющихся процессов

5) Отображение абсолютных путей для каждого процесса

Для отображения абсолютных путей для каждого из процессов необходимо выполнить команду *top* и нажать на клавишу «с». Тем самым значение в столбце **COMMAND** изменится на абсолютный путь для каждого из процессов. Пример выполнения данной команды изображен на рисунке 21.

```
top - 12:52:39 up 1:18, 2 users, load average: 0.01, 0.02, 0.00
Tasks: 138 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 0.7 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731808 free, 94684 used, 190964 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1776428 avail Mem
```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1002	elemabor	20	0	44104	4048	3416	R	1.3	0.2	0:00.41	top
121	root	20	0	0	0	0	I	0.3	0.0	0:01.30	[kworker/0:2]
423	root	0	-20	218216	7116	6088	S	0.3	0.4	0:23.50	/usr/bin/vmtoolsd
1001	root	20	0	0	0	0	I	0.3	0.0	0:00.07	[kworker/u256:2]
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.67	/sbin/init noprompt
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	[kthreadd]
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[kworker/0:0H]
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[mm_percpu_wq]
7	root	20	0	0	0	0	S	0.0	0.0	0:00.59	[ksoftirqd/0]
8	root	20	0	0	0	0	I	0.0	0.0	0:03.38	[rcu_sched]
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	[rcu_bh]
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	[migration/0]
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.07	[watchdog/0]
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[cpuhp/0]
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	[kdevtmpfs]
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[netns]
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[rcu_tasks_kthre]
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kauditd]
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[khungtaskd]
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[oom_reaper]
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[writeback]
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kcompactd0]
21	root	25	5	0	0	0	S	0.0	0.0	0:00.00	[ksmd]
22	root	39	19	0	0	0	S	0.0	0.0	0:00.00	[khugepaged]
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[crypto]
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[kintegrityd]
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[kblockd]
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[ata_sff]
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[md]
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[edac-poller]

Рисунок 21 – Отображения абсолютных путей каждого из процессов

б) Установка периода обновления списка процессов

Для установки периода обновления списка процессов необходимо выполнить команду *top* и после её выполнения нажать на клавишу «d». После этого нас попросят установить период обновления. Установим период обновления раз в 15 секунд. Пример выполнения команды во время установки периода изображен на рисунке 22, после истечения 15 секунд на рисунке 23.

```

top - 12:55:51 up 1:22, 2 users, load average: 0.00, 0.02, 0.00
Tasks: 137 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 1.0 sy, 0.0 ni, 98.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731684 free, 94788 used, 190984 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used, 1776320 avail Mem
Change delay from 3.0 to 15

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1004	elemabor	20	0	44076	4000	3368	R	16.7	0.2	0:00.05	top
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.67	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.62	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:03.51	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.07	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
21	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
22	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	crypto
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq
30	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	watchdogd
34	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kswapd0

Рисунок 22 – Пример установки значения периода обновления на 15 секунд

```
top - 12:56:30 up 1:22, 2 users, load average: 0.00, 0.01, 0.00
Tasks: 137 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.3 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731684 free, 94788 used, 190984 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1776320 avail Mem
```

PID	USER	PR	NI	UIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
423	root	0	-20	218216	7116	6088	S	0.5	0.4	0:24.58	umtoolsd
121	root	20	0	0	0	0	I	0.3	0.0	0:01.84	kworker/0:2
1004	elemabor	20	0	44076	4000	3368	R	0.1	0.2	0:00.10	top
8	root	20	0	0	0	0	I	0.1	0.0	0:03.54	rcu_sched
992	root	20	0	0	0	0	I	0.1	0.0	0:00.45	kworker/u256:1
1001	root	20	0	0	0	0	I	0.1	0.0	0:00.18	kworker/u256:2
428	root	20	0	287548	6892	6000	S	0.0	0.3	0:00.64	accounts-daemon
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.67	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.62	ksoftirqd/0
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.07	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
21	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
22	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	crypto
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff

Рисунок 23 – Обновления информации о процессах через 15 секунд

7) Завершение процессов с помощью аргумента «k»

Для завершения процесса находящегося в списке предоставленной команды *top* необходимо нажать на клавишу «k» и ввести необходимый PID процесса. Для примера выполнения данной команды выполним завершение процесса *top*, его PID = 1006. Пример выполнения данной команды изображен на рисунке 24. После этого необходимо выбрать сигнал для завершения данного процесса, если оставить без изменения, то будет выбран сигнал 15. Пример выполнения данной команды изображен на рисунке 25.

```

top - 12:59:32 up 1:25, 2 users, load average: 0.07, 0.02, 0.00
Tasks: 138 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 1.0 sy, 0.0 ni, 98.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731684 free, 94780 used, 190992 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1776328 avail Mem
PID to signal/kill [default pid = 1006] 1006

```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1006	elemabor	20	0	44076	4056	3424	R	11.1	0.2	0:00.05	top
121	root	20	0	0	0	0	I	5.6	0.0	0:02.12	kworker/0:2
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.68	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.63	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:03.56	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.08	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0

Рисунок 24 – Выбор процесса для завершения

```

Tasks: 138 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 1.0 sy, 0.0 ni, 98.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731684 free, 94780 used, 190992 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1776328 avail Mem
Send pid 1006 signal [15/sigterm]

```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1006	elemabor	20	0	44076	4056	3424	R	11.1	0.2	0:00.05	top
121	root	20	0	0	0	0	I	5.6	0.0	0:02.12	kworker/0:2
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.68	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.63	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:03.56	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.08	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
21	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
22	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	crypto
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq
30	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	watchdogd

elemabor@ubuntu:~/lab3\$

Рисунок 25 – Завершение процесса top с сигналом завершения 15

8) Сортировка процессов по нагрузке на процессор

Для сортировки списка процессов по нагрузке на процессор необходимо выполнить команду *top* для отображения всего списка процессов и выполнить комбинацию клавиш *SHIFT+P*. После этого список процессов будет отсортирован по нагрузке на процессор. Пример выполнения данной команды изображен на рисунке 26.

```
top - 13:05:03 up 1:31, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 138 total, 1 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.4 us, 12.0 sy, 0.0 ni, 82.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1731652 free, 94780 used, 191024 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used, 1776328 avail Mem
```

PID	USER	PR	NI	UIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1010	elemabor	20	0	44076	3992	3360	R	17.3	0.2	0:00.33	top
423	root	0	-20	218216	7116	6088	S	3.7	0.4	0:27.09	mntoolsd
121	root	20	0	0	0	0	I	1.2	0.0	0:03.01	kworker/0:2
1	root	20	0	77392	8380	6560	S	0.0	0.4	0:07.69	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.63	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:03.76	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.08	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
21	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
22	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	crypto
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq

Рисунок 26 – Пример сортировки списка процессов по нагрузке на процессор

9) Изменение приоритета процесса

Для изменения приоритета процесса необходимо получить общий список процессов с помощью команды *top* и найти столбец *NI*, данный столбец показывает приоритет процесса. Для изменения приоритета конкретного процесса необходимо нажать клавишу «*r*», выбрать *PID* процесса у которого необходимо изменить приоритет и ввести новое значение приоритета. Для примера выберем процесс команды *top* (изначальный приоритет данной команды 0) и установим ей новый приоритет равный 10. Результат команды

top до изменения приоритета изображен на рисунке 27. Список процессов после изменения приоритета на рисунке 28.

```
top - 13:10:30 up 0 min, 1 user, load average: 1.81, 0.51, 0.17
Tasks: 164 total, 1 running, 69 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 1.3 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1823880 free, 90900 used, 102676 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1799640 avail Mem
```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
619	elemabor	20	0	44076	4072	3404	R	1.0	0.2	0:00.13	top
26	root	20	0	0	0	0	I	0.7	0.0	0:00.25	kworker/0:1
8	root	20	0	0	0	0	I	0.3	0.0	0:00.54	rcu_sched
416	root	0	-20	144416	6568	5684	S	0.3	0.3	0:00.30	umtoolsd
1	root	20	0	77388	8412	6664	S	0.0	0.4	0:05.57	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
5	root	20	0	0	0	0	I	0.0	0.0	0:00.01	kworker/u256:0
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.30	ksoftirqd/0
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0

Рисунок 27 – Список процессов до изменения приоритета

```
top - 13:11:17 up 1 min, 1 user, load average: 0.85, 0.43, 0.16
Tasks: 164 total, 2 running, 69 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.5 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017456 total, 1823440 free, 91176 used, 102840 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 1799280 avail Mem
```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
619	elemabor	30	10	44076	4072	3404	R	0.6	0.2	0:00.86	top
26	root	20	0	0	0	0	R	0.3	0.0	0:00.39	kworker/0:1
416	root	0	-20	144416	6568	5684	S	0.3	0.3	0:00.52	umtoolsd
250	root	20	0	0	0	0	I	0.2	0.0	0:00.08	kworker/u256:27
1	root	20	0	77388	8412	6664	S	0.0	0.4	0:05.59	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
5	root	20	0	0	0	0	I	0.0	0.0	0:00.01	kworker/u256:0
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.35	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:00.56	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0

Рисунок 28 – Список процессов после изменения приоритета процесса top (PID 619)

10) Сохранения результата выполнения команды *top* в файл

Для сохранения результата выполнения команды *top* в файл необходимо перенаправить поток вывода из консоли на файл. Для этого выполним следующую команду: *top -n 1 -b > top-output.txt*. После этого первые 20 строчек с помощью команды: *head -n 20 top-output.txt*. Пример выполнения данных команд изображен на рисунке 29.

```
elemabor@ubuntu:~/lab3$ top -n 1 -b > top-output.txt
elemabor@ubuntu:~/lab3$ head -n 20 top-output.txt
top - 13:18:00 up 8 min,  1 user,  load average: 0.00, 0.11, 0.09
Tasks: 134 total,  1 running, 69 sleeping,  0 stopped,  0 zombie
%Cpu(s):  2.3 us,  4.8 sy,  0.2 ni, 92.4 id,  0.1 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem : 2017456 total, 1823380 free,  90560 used,  103516 buff/cache
KiB Swap: 483800 total, 483800 free,  0 used. 1799564 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S %CPU  %MEM    TIME+  COMMAND
 661 elemabor  20   0  44076   3872  3304 R 13.0   0.2   0:00.09 top
    1 root       20   0  77388   8416  6668 S  0.0   0.4   0:05.60 systemd
    2 root       20   0         0     0     0 S  0.0   0.0   0:00.01 kthreadd
    4 root       0 -20         0     0     0 I  0.0   0.0   0:00.00 kworker/0:0H
    5 root       20   0         0     0     0 I  0.0   0.0   0:00.01 kworker/u256:0
    6 root       0 -20         0     0     0 I  0.0   0.0   0:00.00 mm_percpu_wq
    7 root       20   0         0     0     0 S  0.0   0.0   0:00.40 ksoftirqd/0
    8 root       20   0         0     0     0 I  0.0   0.0   0:00.82 rcu_sched
    9 root       20   0         0     0     0 I  0.0   0.0   0:00.00 rcu_bh
   10 root      rt    0         0     0     0 S  0.0   0.0   0:00.00 migration/0
   11 root      rt    0         0     0     0 S  0.0   0.0   0:00.00 watchdog/0
   12 root       20   0         0     0     0 S  0.0   0.0   0:00.00 cpuhp/0
   13 root       20   0         0     0     0 S  0.0   0.0   0:00.00 kdevtmpfs
elemabor@ubuntu:~/lab3$ _
```

Рисунок 29 – Перенаправление потока вывода в файл с последующим его выводом

11) Получение справки по командам процесса *top*

Для получения справки по командам процесса *top* необходимо запустить сам процесс *top* и нажать на клавишу «h». После этого откроется новое окно с описанием всех возможных клавиш и их описанием. Пример данного окна изображен на рисунке 30.

```
Help for Interactive Commands - procps-ng 3.3.12
Window 1:Def: Cumulative mode Off. System: Delay 3.0 secs; Secure mode Off.

Z,B,E,e Global: 'Z' colors; 'B' bold; 'E'/'e' summary/task memory scale
l,t,m Toggle Summary: 'l' load avg; 't' task/cpu stats; 'm' memory info
0,1,2,3,I Toggle: '0' zeros; '1/2/3' cpus or numa node views; 'I' Irix mode
f,F,X Fields: 'f'/'F' add/remove/order/sort; 'X' increase fixed-width

L,&,<,> . Locate: 'L'/'&' find/again; Move sort column: '<'/'>' left/right
R,H,U,J . Toggle: 'R' Sort; 'H' Threads; 'U' Forest view; 'J' Num justify
c,i,S,j . Toggle: 'c' Cmd name/line; 'i' Idle; 'S' Time; 'j' Str justify
x,y . Toggle highlights: 'x' sort field; 'y' running tasks
z,b . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
u,U,o,o . Filter by: 'u'/'U' effective/any user; 'o'/'O' other criteria
n,#,^0 . Set: 'n'/'#' max tasks displayed; Show: Ctrl+'O' other filter(s)
C,... . Toggle scroll coordinates msg for: up,down,left,right,home,end

k,r Manipulate tasks: 'k' kill; 'r' renice
d or s Set update interval
W,Y Write configuration file 'W'; Inspect other output 'Y'
q Quit
( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
Type 'q' or <Esc> to continue █
```

Рисунок 30 – Пример окна справки процесса top

12) Завершение команды top после определенного числа повторений

Команда top обновляется каждый раз, когда происходит нажатие на клавишу «q». Для того, чтобы завершить процесс top после определенного числа повторений необходимо выполнить следующую команду: *top -n 5*. Тем самым, процесс top завершится после 5 повторений.

Следующим шагом изучим возможность запуска процессов в supervisor. Сначала, дадим определение supervisor`у. Supervisor – это система клиент/сервер, при помощи которой пользователь (администратор) может контролировать подключенные процессы в системах типа UNIX. Инструмент создает процессы в виде подпроцессов от своего имени, поэтому имеет полный контроль над ними.

Произведем установку и настройку supervisor. Для установки данного пакета необходимо выполнить следующую команду: *sudo apt get install supervisor*. Пример установки данного пакета изображен на рисунке 32.

```

elemabor@ubuntu:~$ sudo apt-get install supervisor
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-meld3 python-minimal
  python-pkg-resources python2.7 python2.7-minimal
Suggested packages:
  python-doc python-tk python-setuptools python2.7-doc binutils binfmt-support supervisor-doc
The following NEW packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-meld3 python-minimal
  python-pkg-resources python2.7 python2.7-minimal supervisor
0 upgraded, 10 newly installed, 0 to remove and 21 not upgraded.
Need to get 4,380 kB of archives.
After this operation, 19.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y_

```

Рисунок 31 – Установка Supervisor

Следующим шагом нам необходимо написать скрипт, который будет необходимо выполнять. Напишем простой скрипт на языке Python, который будет раз в 5 секунд выводить случайное число и текущее время с датой. Содержание созданного скрипта изображено на рисунке 32.

```

from time import sleep
from datetime import datetime
from random import randrange

while True:
    sleep(5)
    print(f"Rand int: {randrange(100, 999)} in {datetime.now()}", flush=True)

```

Рисунок 32 – Скрипт на языке Python

Далее supervisor необходимо сконфигурировать и добавить программы/процессы, которыми он будет управлять. Файл конфигурации находится в `/etc/supervisor/supervisord.conf`. Дополним его содержимым изображенным на рисунке 33.

Рассмотрим параметры дополненные в данном файле:

- `[program:worker]` – название процесса/воркера, к которому будут относиться все последующие секции.
- `command=python3 -u /home/elembor/lab3/task4.py` – команда на запуск файла, и путь к необходимому файлу.
- `stdout_logfile=/home/elembor/lab3/task4_out.log` – вывод консоли в файл.
- `autostart=true` – запуск воркера вместе с запуском supervisor.
- `autorestart=true` – перезапуск воркера, если тот по какой-то причине

упал.

- user=elemabor – запуск процесса под определенным пользователем.
- stopsignal=KILL – сигнал остановки процесса. По умолчанию – TERM.
- numprocs=1 – количество экземпляров заданного воркера.

```
; supervisor config file

[unix_http_server]
file=/var/run/supervisor.sock ; (the path to the socket file)
chmod=0700 ; socket file mode (default 0700)

[supervisord]
logfile=/var/log/supervisor/supervisord.log ; (main log file;default $CWD/supervisord.log)
pidfile=/var/run/supervisord.pid ; (supervisord pidfile;default supervisord.pid)
childlogdir=/var/log/supervisor ; ('AUTO' child log dir, default $TEMP)

[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[supervisorctl]
serverurl=unix:///var/run/supervisor.sock ; use a unix:// URL for a unix socket

[include]
files = /etc/supervisor/conf.d/*.conf

[program:worker]
command=python -u _/home/elemabor/lab3/task4.py
stdout_logfile=/home/elemabor/lab3/task4_out.log
autostart=true
autorestart=true
user=elemabor
stopsignal=KILL
numprocs=1
~
~
~
~
~
~
~
"/etc/supervisor/supervisord.conf" 28L, 888C 22,18 All
```

Рисунок 33 – Создание воркера управления процессом task4.py

После добавления новых процессов/воркеров перезагрузим supervisor с помощью следующей команды: */etc/init.d/supervisor restart*. После перезапуска и некоторой работы мы увидим, что в указанной папке появился файл task4_out.log. Это изображено на рисунке 34.

```
elemabor@ubuntu:~/lab3$ rm -rf task4_out.log
elemabor@ubuntu:~/lab3$ ls
cat_file.txt task2.txt task3.txt task4.py top-output.txt
elemabor@ubuntu:~/lab3$ sudo /etc/init.d/supervisor restart
[ ok ] Restarting supervisor (via systemctl): supervisor.service.
elemabor@ubuntu:~/lab3$ ls
cat_file.txt task2.txt task3.txt task4_out.log task4.py top-output.txt
elemabor@ubuntu:~/lab3$ _
```

Рисунок 34 – Создание лог файла нового процесса

Задание 5

Изучение возможности автоматического запуска программ по расписанию.

Для автоматического запуска программ по расписанию используется `cron`. `Cron` – это программа-демон, предназначенная для выполнения заданий в определенное время, или через определенные временные промежутки. Для редактирования заданий используется утилита `crontab`.

Создадим новый файл `cron` на запуск процесса `yes` каждую минуту. Посмотрим содержимое расписания с помощью следующей команды: `crontab -l`. Добавим созданное расписание с помощью команды `crontab cron_file`. Еще раз посмотрим на содержимое списка расписания. И в заключении посмотрим на успешный запуск процесса в указанное время через планировщика с помощью команды `ps -fC yes`. Полученный результат изображен на рисунке 35.

```
elemabor@ubuntu:~/lab3$ crontab -l
no crontab for elemabor
elemabor@ubuntu:~/lab3$ crontab cron_file
elemabor@ubuntu:~/lab3$ crontab -l
SHELL=/bin/bash
MAILTO=elemabor
* * * * * yes > /dev/null
elemabor@ubuntu:~/lab3$ ps -fC yes
UID          PID    PPID  C STIME TTY          TIME CMD
elemabor@ubuntu:~/lab3$ ps -fC yes
UID          PID    PPID  C STIME TTY          TIME CMD
elemabor   10198   10197  94 14:25 ?        00:00:03 yes
elemabor@ubuntu:~/lab3$ ps -fC yes
UID          PID    PPID  C STIME TTY          TIME CMD
elemabor   10198   10197  99 14:25 ?        00:00:10 yes
elemabor@ubuntu:~/lab3$ _
```

Рисунок 35 – Успешный запуск процесса в указанное время

Вывод

В ходе выполнения данной лабораторной работы я повторил использование команд `cat`, `head`, `tail`, `more`, `less`, `grep` и `find` на практике. Разобрался с понятиями конвейер и перенаправление ввода-вывода. Повторил назначение прав доступа к файлам и каталогам с помощью команд `chmod` (изменение прав доступа к файлу) и `chown` (изменение владельца файла). Ознакомился с информацией по теме процессы, опробовал 12 примеров наиболее распространенных команд и изучил возможность запуска процессов в `supervisor` на примере написанного скрипта на Python. Также на практике была изучена возможность запуска программ по расписанию с помощью программ `cron` и `crontab`.