

**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

Мастылина А.А.

Группа АИ-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

## Оглавление

Цель работы .....	3
Задание кафедры.....	4
Ход работы .....	5
Вывод .....	17

## 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(1).
3. Ознакомиться с информацией из рекомендованных источников(2) и других про конвейеризации(3).
4. Повторить назначение прав доступа (4). Команды chmod, chown
5. Ознакомиться с информацией(5) по теме процессы, посмотреть и опробовать примеры наиболее распространенных(6) команд, изучить возможность запуска процессов в supervisor (7).
6. Изучить возможность автоматического запуска программ по расписанию

## Ход работы

Повторим команды `cat` и `head`. Создадим с помощью команды `cat` файл `one` и выведем начальные строки файла `one` (по умолчанию 10). Результаты работы представлены на рисунке 1.

Команда `head` выводит начальные строки из одного или нескольких документов.

### \$ head опции файл

Чаще всего к команде `head` применяются такие опции:

- c — позволяет задавать количество текста не в строках, а в байтах.
- n — показывает заданное количество строк вместо 10, которые выводятся по умолчанию.
- q — выводит только текст, не добавляя к нему название файла.
- v — перед текстом выводит название файла.
- z — символы перехода на новую строку заменяет символами завершения строк.

С помощью команды `cat` можно очень просто посмотреть содержимое небольшого файла, склеить несколько файлов и многое другое.

### \$ cat опции файл1 файл2 ...

Опции команды `cat`:

- b - нумеровать только непустые строки;
- E - показывать символ \$ в конце каждой строки;
- n - нумеровать все строки;
- s - удалять пустые повторяющиеся строки;
- T - отображать табуляции в виде ^I;
- h - отобразить справку;
- v - версия утилиты.

```
root@annaserver:/home/anna# cat > one
q
w
e
r
^C
root@annaserver:/home/anna# head -v one
==> one <==
q
w
e
r
root@annaserver:/home/anna# _
```

Рисунок 1 – Использование команды cat и head

Повторим команду tail. Выведем последние 10 строк из файла two. Результаты работы представлены на рисунке 2. Команда tail по умолчанию выводит десять последних строк из файла.

Опции команды tail:

- с - выводить указанное количество байт с конца файла;
- f - обновлять информацию по мере появления новых строк в файле;
- n - выводить указанное количество строк из конца файла;
- pid - используется с опцией -f, позволяет завершить работу утилиты, когда завершится указанный процесс;
- q - не выводить имена файлов;
- retry - повторять попытки открыть файл, если он недоступен;
- v - выводить подробную информацию о файле;

```
root@annaserver:/home/anna# cat > two
q
w
e
r
t
y
u
i
o
p
a
s
d
f
g
h
j
k
l
root@annaserver:/home/anna# tail two
y
u
i
o
p
a
s
d
f
g
h
j
k
l
root@annaserver:/home/anna#
```

Рисунок 2 – Использование команды cat и tail

Далее откроем файл two с помощью команд more и less. Результаты работы представлены на рисунке 3.

more предназначена для постраничного просмотра файлов.

\$ more опции файл

Опции команды more

-d — вывод информации в конце страницы о клавишах, используемых для продолжения работы, завершения её или получения инструкций;

-l — игнорирование в тексте символа разрыва страницы;

-f — подсчёт числа логических строк вместо экранных;

-p — очистка экрана терминала для того, чтобы пользователю не пришлось пользоваться прокруткой перед выводом следующей порции текста;

-c — устранение потребности в прокрутке (как и -p) — отображение текста, начиная с верха экрана, и стирание при этом предыдущего вывода построчно;

-s — замена нескольких пустых строк, расположенных подряд, одной пустой строкой;

-u — удаление подчёркивания;

-n — отображение n-го количества строк;

+n — отображение текста, начиная со строки с номером n;

+/-строка — поиск в файле указанной строки и начало вывода текста именно с неё;

--help — вызов справки;

-v (--version) — вывод на экран текущей версии утилиты.

Утилита less предназначена для постраничного вывода содержимого текстовых файлов значительного объема. Она имеет больше функций чем команда more.

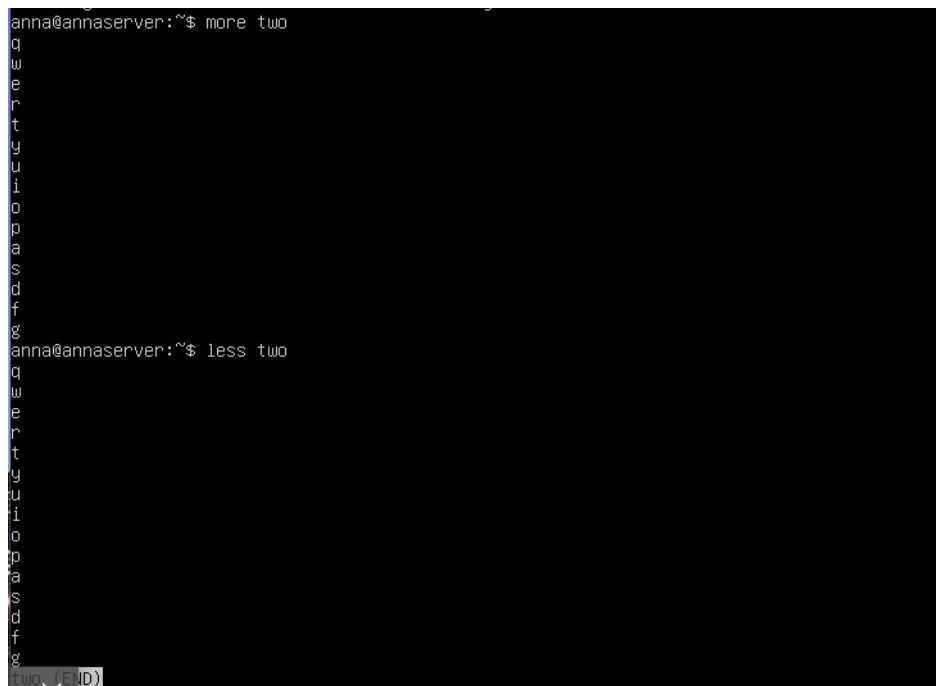


Рисунок 3 – Использование команд more и less

С помощью команды grep найдём в файле two элемент q, с помощью команды find найдём файл two. Результат работы представлен на рисунке 4.

Grep - это утилита командной строки Linux, который даёт пользователям возможность вести поиск строки. С его помощью можно даже искать конкретные слова в файле.

grep [опции] шаблон [имя файла...]

Опции команды grep

-b - показывать номер блока перед строкой;

-c - подсчитать количество вхождений шаблона;

-h - не выводить имя файла в результатах поиска внутри файлов Linux;

-i - не учитывать регистр;



- l - отобразить только имена файлов, в которых найден шаблон;
- n - показывать номер строки в файле;
- s - не показывать сообщения об ошибках;
- v - инвертировать поиск, выдавать все строки кроме тех, что содержат шаблон;

- w - искать шаблон как слово, окружённое пробелами;

- e - использовать регулярные выражения при поиске;

- An - показать вхождение и n строк до него;

- Bn - показать вхождение и n строк после него;

- Cn - показать n строк до и после вхождения;

Команда `find` позволяет искать файлы не только по названию, но и по дате добавления, содержимому и регулярным выражениям.

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

Некоторые параметры команды `find`.

- 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

- type f - искать только файлы

- type d - поиск папки в Linux



```
anna@annaserver:~$ grep q two
q
anna@annaserver:~$ find two
two
anna@annaserver:~$ _
```

Рисунок 4 – Использование команд `grep` и `find`

Следующим заданием нужно разобраться с понятиями перенаправления ввода-вывода. Результат работы представлен на рисунке 5.

В данном примере мы перенаправляем содержимое файла t.txt в u.txt, перезаписываем u.txt. Следующим шагом перенаправим содержимое файла t.txt в файл u.txt, информация добавится в конец файла u.txt. Далее перенаправим содержимое файла u.txt на стандартный ввод программы.

Операторы перенаправления:

> - перенаправляет стандартный поток в файл (другой поток). При этом если файл существует, то он перезаписывается, если не существует – создается.

>> - перенаправляет стандартный поток в файл. При этом если файл существует, то информация добавляется в конец, если не существует – файл создается.

< - перенаправляет содержимое указанного файла на стандартный ввод программы.

>& - перенаправляет стандартные потоки вывода и ошибок друг в друга.




```
root@annaserver:/home/anna# cat > u.txt
^C
root@annaserver:/home/anna# cat > t.txt
q
r
p
^C
root@annaserver:/home/anna# head t.txt > u.txt
root@annaserver:/home/anna# cat u.txt
q
r
p
root@annaserver:/home/anna# head t.txt >> u.txt
root@annaserver:/home/anna# cat u.txt
q
r
p
q
r
p
root@annaserver:/home/anna# cat < u.txt
q
r
p
q
r
p
root@annaserver:/home/anna#
```

Рисунок 5 – Перенаправление ввода-вывода

Разберёмся с понятием конвейер.

Конвейеры — это возможность нескольких программ работать совместно, когда выход одной программы непосредственно идет на вход другой без использования промежуточных временных файлов.

В данном случае мы хотим направить stdout команды ls на stdin команды sort. Результат работы представлен на рисунке 6.



```
root@annaserver:/home/anna# ls | sort -r
u.txt
txtone
two
text1
t.txt
ress
res
one
new
loop2.sh
loop.sh
channel
chan
```

Рисунок 6 – Использование состыкованных команд (конвейер)

Далее повторим назначение прав доступа. Команды chmod, chown. Результаты работы представлены на рисунках 7 и 8.

\$ chown пользователь опции /путь/к/файлу.

Вот основные опции, которые могут понадобиться:

- c, --changes - подробный вывод всех выполняемых изменений;
- f, --silent, --quiet - минимум информации, скрыть сообщения об ошибках;
- dereference - изменять права для файла к которому ведет символическая ссылка вместо самой ссылки (поведение по умолчанию);
- h, --no-dereference - изменять права символических ссылок и не трогать файлы, к которым они ведут;
- from - изменять пользователя только для тех файлов, владельцем которых является указанный пользователь и группа;
- R, --recursive - рекурсивная обработка всех подкаталогов;
- H - если передана символическая ссылка на директорию - перейти по ней;
- L - переходить по всем символическим ссылкам на директории;

-P - не переходить по символическим ссылкам на директории (по умолчанию).

\$ chmod опции права /путь/к/файлу.

Есть три основных вида прав:

r - чтение;

w - запись;

x - выполнение;

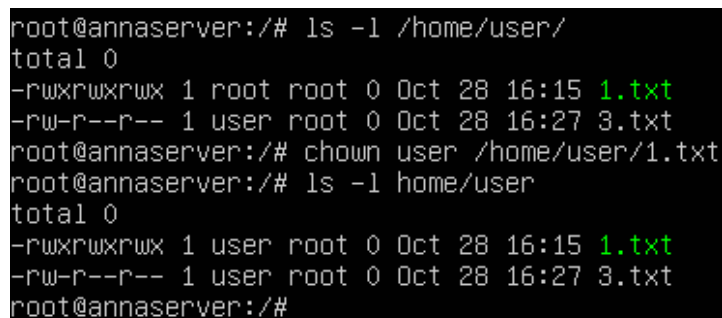
s - выполнение от имени суперпользователя (дополнительный);

Также есть три категории пользователей, для которых вы можете установить эти права на файл linux:

u - владелец файла;

g - группа файла;

o - все остальные пользователи;



```
root@annaserver:/# ls -l /home/user/  
total 0  
-rwxrwxrwx 1 root root 0 Oct 28 16:15 1.txt  
-rw-r--r-- 1 user root 0 Oct 28 16:27 3.txt  
root@annaserver:/# chown user /home/user/1.txt  
root@annaserver:/# ls -l /home/user  
total 0  
-rwxrwxrwx 1 user root 0 Oct 28 16:15 1.txt  
-rw-r--r-- 1 user root 0 Oct 28 16:27 3.txt  
root@annaserver:/#
```

Рисунок 7 – Использование команды chown

Изменим права доступа на файл 1.txt в директории пользователя user с помощью команды chmod. Результат работы представлен на рисунке 8. Здесь мы установили права на чтение, запись и исполнение файла для всех категорий пользователя: 777.

```
root@annaserver:~# chmod 777 /home/user/1.txt
root@annaserver:~# ls-l
ls-l: command not found
root@annaserver:~# ls -l
total 4
drwxr-xr-x 3 root root 4096 Oct 23 07:58 snap
root@annaserver:~#
```

### Рисунок 8 – Использование команды chmod

Посмотрим и опробуем примеры наиболее распространенных команд.

Например команда `top`. Результат работы представлен на рисунках 9-12.

Она показывает, что процессор и память используются, а также другую информацию, такую как запущенные процессы.

q или Esc - выход из `top`;

A - выбор цветовой схемы;

d или s - изменить интервал обновления информации;

H - выводить потоки процессов;

k - послать сигнал завершения процессу;

W - записать текущие настройки программы в конфигурационный файл;

Y - посмотреть дополнительные сведения о процессе, открытые файлы, порты, логи и т.д.;

Z - изменить цветовую схему;

l - скрыть или вывести информацию о средней нагрузке на систему;

m - выключить или переключить режим отображения информации о памяти;

x - выделять жирным колонку, по которой выполняется сортировка;

y - выделять жирным процессы, которые выполняются в данный момент;

z - отображает запущенный процесс в цвете, который помогает легко идентифицировать запущенный процесс;

c - переключение режима вывода команды, доступен полный путь и только команда;

F - настройка полей с информацией о процессах;

- o - фильтрация процессов по произвольному условию;
- u - фильтрация процессов по имени пользователя;
- V - отображение процессов в виде дерева;
- i - переключение режима отображения процессов, которые сейчас не используют ресурсы процессора;
- n - максимальное количество процессов, для отображения в программе;
- L - поиск по слову;
- < > - перемещение поля сортировки вправо и влево;

```

top - 14:23:32 up 1:45, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 92 total, 1 running, 91 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.3 sy, 0.0 ni, 99.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 981.3 total, 550.3 free, 131.6 used, 299.3 buff/cache
MiB Swap: 1766.0 total, 1766.0 free, 0.0 used, 701.8 avail Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
996	anna	20	0	7916	3620	3092	R	0.3	0.4	0:00.10	top
1	root	20	0	101992	11524	8444	S	0.0	1.1	0:02.10	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblockd
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.12	ksoftirqd/0
11	root	20	0	0	0	0	I	0.0	0.0	0:02.35	rcu_sched
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.05	migration/0
13	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
16	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
23	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
24	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
70	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
71	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
72	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
73	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
74	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
75	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
76	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
77	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq

Рисунок 9 – Использование команды top

```

top - 14:38:32 up 2:00, 1 user, load average: 0.08, 0.03, 0.01
Tasks: 91 total, 1 running, 90 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 981.3 total, 549.8 free, 130.6 used, 300.9 buff/cache
MiB Swap: 1766.0 total, 1766.0 free, 0.0 used, 702.8 avail Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1020	anna	20	0	7916	3748	3220	R	0.7	0.4	0:00.03	top
1	root	20	0	101992	11524	8444	S	0.0	1.1	0:02.11	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblockd
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.12	ksoftirqd/0
11	root	20	0	0	0	0	I	0.0	0.0	0:02.58	rcu_sched
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.06	migration/0
13	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
16	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
23	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
24	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
70	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
71	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
72	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
73	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
74	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
75	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
76	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
77	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq

Рисунок 10 – Использование команды top, пример нажатия клавиши z  
Изменим кнопкой d интервал обновления экрана с 3.0 секунд до 2.0 секунд

```

top - 14:45:05 up 2:06, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 91 total, 1 running, 90 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
MiB Mem : 981.3 total, 549.8 free, 130.6 used, 300.9 buff/cache
MiB Swap: 1766.0 total, 1766.0 free, 0.0 used, 702.8 avail Mem
Change delay from 2.0 to 2.0

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1004	root	20	0	0	0	0	I	0.5	0.0	0:00.16	[kworker/u2:1-event
1005	root	20	0	0	0	0	I	0.5	0.0	0:01.08	[kworker/0:2-event
1	root	20	0	101992	11524	8444	S	0.0	1.1	0:02.12	/sbin/init maybe-u
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kthreadd]
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[rcu_gp]
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[rcu_par_gp]
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[kworker/0:0H-kblo
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[mm_percpu_wq]
10	root	20	0	0	0	0	S	0.0	0.0	0:00.14	[ksoftirqd/0]
11	root	20	0	0	0	0	I	0.0	0.0	0:02.61	[rcu_sched]
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.06	[migration/0]
13	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	[idle_inject/0]
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[cpuhp/0]
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kdevtmpfs]
16	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[netns]
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[rcu_tasks_kthre]
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kauditd]
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[khungtaskd]
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[oom_reaper]
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[writeback]
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kcompactd0]
23	root	25	5	0	0	0	S	0.0	0.0	0:00.00	[ksmd]
24	root	39	19	0	0	0	S	0.0	0.0	0:00.00	[khugepaged]
70	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[kintegrityd]
71	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[kblockd]
72	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[blkcg_punt_bio]
73	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[tpm_dev_wq]
74	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[ata_sff]
75	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[md]
76	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	[edac-poller]

Рисунок 11 – Использование команды top, пример нажатия клавиши d

При нажатии на клавишу **h** получим справку по верхней команде

```
Help for Interactive Commands - procps-ng UNKNOWN
Window 1:Def: Cumulative mode Off. System: Delay 2.0 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,J,C . Toggle: 'R' Sort; 'H' Threads; 'J' Num justify; 'C' Coordinates
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,q,0 . Filter by: 'u'/'U' effective/any user; 'o'/'O' other criteria
n,#,0 . Set: 'n'/'#' max tasks displayed; Show: Ctrl+'O' other filter(s)
V,v . Toggle: 'V' forest view; 'v' hide/show forest view children

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 12 – Использование команды **top**, пример нажатия клавиши **h**  
Изучим возможность запуска процессов в supervisor.

Supervisor — это система клиент/сервер, при помощи которой пользователь (администратор) может контролировать подключенные процессы в системах типа UNIX. Инструмент создает процессы в виде под-процессов от своего имени, поэтому имеет полный контроль над ними.

Изучим возможность автоматического запуска программ по расписанию  
**cron** – программа-демон, предназначенная для выполнения заданий в определенное время, или через определенные промежутки времени. Для редактирования заданий используется утилита **crontab**.

Время запуска представляется в таком виде:

минута час день\_месяца месяц день\_недели команда



## Вывод

В ходе выполнения лабораторной работы я ознакомилась на практике с понятием процесса в операционной системе, приобрела опыт и навыки управления процессами в операционной системе Linux, были закреплены умения по командам `cat`, `head`, `tail`, `more`, `less`, `grep`, `find`, `chown`, `chmod`, изучено понятие конвейера и основные принципы перенаправления ввода-вывода, утилиты `top`, `cron` и `supervisor`.