

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине «Операционная система Linux»

Процессы в операционной системе Linux

Студент

Осипов А.А.

Группа АИ-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Оглавление

Цель работы	3
Задание кафедры.....	4
Ход работы	6
Вывод	18

Цель работы

Ознакомиться на практике с понятием процесса в операционной системе. Приобрести опыт и навыки управления процессами в операционной системе Linux.

Задание кафедры

Часть I

- 1) Загрузиться не root, а пользователем.
- 2) Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.
- 3) Посмотреть процессы `ps -f`. Прокомментировать. Для этого почитать `man ps`.
- 4) Написать с помощью редактора `vi` два сценария `loop` и `loop2`. Текст сценариев:

Loop:

```
while true; do true; done
```


Loop2:

```
while true; do true; echo 'Hello'; done
```
- 5) Запустить `loop2` на переднем плане: `sh loop2`.
- 6) Остановить, послав сигнал `STOP`.
- 7) Посмотреть последовательно несколько раз `ps -f`. Записать сообщение, объяснить.
- 8) Убить процесс `loop2`, послав сигнал `kill -9 PID`. Записать сообщение. Прокомментировать.
- 9) Запустить в фоне процесс `loop`: `sh loop&`. Не останавливая, посмотреть несколько раз: `ps -f`. Записать значение, объяснить.
- 10) Завершить процесс `loop` командой `kill -15 PID`. Записать сообщение, прокомментировать.
- 11) Третий раз запустить в фоне. Не останавливая убить командой `kill -9 PID`.
- 12) Запустить еще один экземпляр оболочки: `bash`.
- 13) Запустить несколько процессов в фоне. Останавливать их и снова запускать. Записать результаты просмотра командой `ps -f`.

Часть II

1. Запустить в консоли на выполнение три задачи, две в интерактивном

режиме, одну - в фоновом.

2. Перевести одну из задач, выполняющихся в интерактивном режиме, в фоновый режим.

3. Провести эксперименты по переводу задач из фонового режима в интерактивный и наоборот.

4. Создать именованный канал для архивирования и осуществить передачу в канал

- списка файлов домашнего каталога вместе с подкаталогами (ключ -R);
- одного каталога вместе с файлами и подкаталогами.

5. В отчете предоставьте все шаги ваших действий. То есть следует привести следующее: текст задания, а следом за ним снимок экрана консоли с результатами выполнения задания. Кроме того, перед скриншотом следует привести текстовую запись использованных команд.

Часть III Индивидуальное задание

Вариант 2

1. Получить следующую информацию о процессах текущего пользователя: идентификатор и имя владельца процесса, статус и приоритет процесса.

2. Завершить выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершить с помощью сигнала SIGINT, задав его имя, второй — с помощью сигнала SIGQUIT, задав его номер.

3. Определить идентификаторы и имена процессов, идентификатор группы которых не равен идентификатору группы текущего пользователя.

4. В отчете предоставьте все шаги ваших действий. То есть следует привести следующее: текст задания, а следом за ним снимок экрана консоли с результатами выполнения задания. Кроме того, перед скриншотом следует привести текстовую запись использованных команд. Кратко поясните результаты выполнения всех команд.

Ход работы

Часть I

Запустим виртуальную машину Linux Ubuntu и загрузимся не root, а пользователем. Найдём файл с обзором ядра и выясним по имени файла номер версии Linux. Результаты работы представлены на рисунке 1.

```
artem@artemserver:~$ cd /boot
artem@artemserver:/boot$ ls
config-5.4.0-42-generic  initrd.img-5.4.0-42-generic  System.map-5.4.0-42-generic  vmlinuz.old
grub                    initrd.img.old               vmlinuz
initrd.img              lost+found                   vmlinuz-5.4.0-42-generic
artem@artemserver:/boot$
```

Рисунок 1 – Загрузка пользователем, версия ядра

Исходя из рисунка 1 ядро имеет версию 5.4.0

Далее на рисунке 2 посмотрим процессы `ps -f`.

```
artem@artemserver:/boot$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem        12163    719   0   08:57 tty1        00:00:00 -bash
artem        12193   12163   0   09:01 tty1        00:00:00 ps -f
artem@artemserver:/boot$ _
```

Рисунок 2 – Просмотр процессов `ps -f`

Поясним просмотр процессов.

UID - имя пользователя, от имени которого работает процесс;

PID - идентификатор пользователя;

PPID - идентификатор родительского процесса пользователя;

C - расходование ресурсов процессора, в процентах;

STIME - время, когда процесс был запущен;

TTY - если процесс привязан к терминалу, то здесь будет выведен его номер;

TIME - общее время выполнения процесса (user + system);

CMD - команда, которой был запущен процесс, если программа не может прочитать аргументы процесса, он будет выведен в квадратных скобках;

Напишем с помощью редактора `vi` два сценария `loop` и `loop2`. Текст сценариев: (`cd /home`). Результаты работы приведены на рисунках 3, 4 и 5.

Loop:

```
while true; do true; done
```

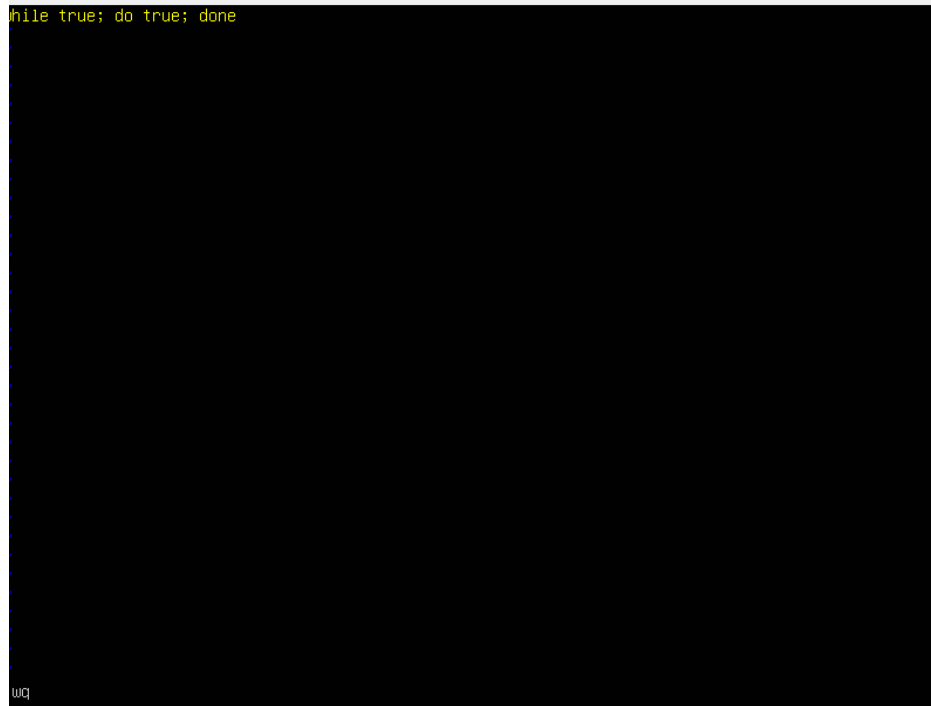


Рисунок – 3 Написание сценария loop



Рисунок 4 – Написание сценария loop2

Loop2:

```
while true; do true; echo 'Hello'; done
```


Далее запустим loop2 на переднем плане: `sh loop2.sh` и остановим, послав сигнал `STOP`. Результат работы представлен на рисунке 7.

[illegible]

Рисунок 7 – Запуск loop 2 на переднем плане loop2.sh и его остановка
сигналом стоп

Следующим заданием посмотрим последовательно несколько раз $\text{ps} -f$.
Результат работы представлен на рисунке 8.

```

artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem        12163      719  0 08:57 tty1          00:00:00 -bash
artem        12205      12163  1 09:08 tty1          00:00:02 sh loop2.sh
artem        12212      12163  41 09:11 tty1          00:00:09 sh loop2.sh
artem        12214      12163  0 09:11 tty1          00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem        12163      719  0 08:57 tty1          00:00:00 -bash
artem        12205      12163  1 09:08 tty1          00:00:02 sh loop2.sh
artem        12212      12163  30 09:11 tty1          00:00:09 sh loop2.sh
artem        12215      12163  0 09:11 tty1          00:00:00 ps -f
artem@artemserver:~$

```

Рисунок 8 – Просмотр листинга процессов

Из рисунка 8 следует, что расходование ресурсов процессора, в процентах, затраченный на процесс `loop2`, уменьшается, что говорит о приостановке процессора.

Далее нужно убить процесс loop2, послав сигнал kill -9 PID. Результат работы представлен на рисунке 9.

```
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem        12212     12163  5   09:11 tty1      00:00:09 sh loop2.sh
artem        12219     12163  0   09:13 tty1      00:00:00 ps -f
artem@artemserver:~$ kill -9 12212
[2]+  Killed                  sh loop2.sh
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem        12220     12163  0   09:14 tty1      00:00:00 ps -f
artem@artemserver:~$
```

Рисунок 9 – Уничтожение процесса loop2.sh

Далее нужно запустить в фоне процесс loop: sh loop&. Не останавливая, посмотреть несколько раз: ps -f. Результат работы представлен на рисунке 10.

```
artem@artemserver:~$ sh loop.sh&
[1] 12222
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem        12222     12163  99   09:15 tty1      00:00:06 sh loop.sh
artem        12223     12163  0   09:15 tty1      00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem        12222     12163  99   09:15 tty1      00:00:09 sh loop.sh
artem        12224     12163  0   09:15 tty1      00:00:00 ps -f
artem@artemserver:~$
```

Рисунок 10 – Запуск процесса в фоне и просмотр ps -f

По рисунку 10 заметно, что расходование ресурсов процессора, в процентах, затраченный на процесс loop.sh, не уменьшается, что говорит о том что процесс запущен.

Далее завершим процесс loop командой kill -15 PID. Результат работы представлен на рисунке 11.

```
artem@artemserver:~$ kill -15 12222
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem        12226     12163  0   09:16 tty1      00:00:00 ps -f
[1]+  Terminated              sh loop.sh
artem@artemserver:~$ _
```

Рисунок 11 – Заверение процесса loop.sh

А теперь снова запустим в фоне процесс loop.sh и не останавливая убьём командой kill -9 PID. Результат работы представлен на рисунке 12.

```
artem@artemserver:~$ sh loop.sh&
[1] 12236
artem@artemserver:~$ sh loop.sh&
[2] 12237
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0  08:57 tty1        00:00:00 -bash
artem        12236      12163 66  09:18 tty1        00:00:05 sh loop.sh
artem        12237      12163 48  09:18 tty1        00:00:02 sh loop.sh
artem        12238      12163  0  09:18 tty1        00:00:00 ps -f
artem@artemserver:~$ kill -9 12236 12237
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0  08:57 tty1        00:00:00 -bash
artem        12239      12163  0  09:19 tty1        00:00:00 ps -f
[1]-  Killed                  sh loop.sh
[2]+  Killed                  sh loop.sh
artem@artemserver:~$ _
```

Рисунок 12 – Запуск процесса loop.sh в фоне и его убийство командой kill -9

Запустим еще один экземпляр оболочки с помощью команды bash. Результат работы представлен на рисунке 13.

```
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0  08:57 tty1        00:00:00 -bash
artem        12241      12163  0  09:21 tty1        00:00:00 ps -f
artem@artemserver:~$ bash
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0  08:57 tty1        00:00:00 -bash
artem        12242      12163  0  09:21 tty1        00:00:00 bash
artem        12248      12242  0  09:21 tty1        00:00:00 ps -f
artem@artemserver:~$
```

Рисунок 13 – Запуск еще одного экземпляра

Запустим несколько процессов в фоне. Будем останавливать их и снова запускать. Результат работы представлен на рисунке 14.

```
artem@artemserver:~$ sh loop.sh&
[2] 44917
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem       33222     12163  15   09:29 tty1      00:00:57 sh loop.sh
artem       44917     12163  99   09:34 tty1      00:00:03 sh loop.sh
artem       44918     12163   0   09:35 tty1      00:00:00 ps -f
artem@artemserver:~$ kill -19 33222 44917
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem       33222     12163  14   09:29 tty1      00:00:57 sh loop.sh
artem       44917     12163  78   09:34 tty1      00:00:26 sh loop.sh
artem       44919     12163   0   09:35 tty1      00:00:00 ps -f

[2]+  Stopped                  sh loop.sh
artem@artemserver:~$ sh loop.sh&
[3] 44920
artem@artemserver:~$ sh loop.sh&
[4] 44921
artem@artemserver:~$ kill -19 44920 44921
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem        12163       719  0   08:57 tty1      00:00:00 -bash
artem       33222     12163  12   09:29 tty1      00:00:57 sh loop.sh
artem       44917     12163  29   09:34 tty1      00:00:26 sh loop.sh
artem       44920     12163  49   09:35 tty1      00:00:17 sh loop.sh
artem       44921     12163  46   09:35 tty1      00:00:14 sh loop.sh
artem       44922     12163   0   09:36 tty1      00:00:00 ps -f

[3]+  Stopped                  sh loop.sh
[4]-  Stopped                  sh loop.sh
artem@artemserver:~$
```

Рисунок 14 – Запуск процессов в фоне и их остановка

Часть II

Нужно запустить в консоли на выполнение три задачи, две в интерактивном режиме, одну - в фоновом (`sh loop.sh`, `sh loop.sh`, `sh loop.sh&`). Далее посмотрим все процессы с помощью командой `jobs`. Результат работы представлен на рисунке 15.

```
artem@artemserver:~$ sh loop.sh
^Z
[1]+  Stopped                  sh loop.sh
artem@artemserver:~$ sh loop.sh
^Z
[2]+  Stopped                  sh loop.sh
artem@artemserver:~$ sh loop.sh&
[3] 896
artem@artemserver:~$ jobs
[1]-  Stopped                  sh loop.sh
[2]+  Stopped                  sh loop.sh
[3]   Running                  sh loop.sh &
artem@artemserver:~$
```

Рисунок 15 – Запуск на выполнение трёх задач

Следующим заданием нужно перевести одну из задач, выполняющихся в интерактивном режиме, в фоновый режим (bg). Результат работы представлен на рисунке 16.

```
artem@artemserver:~$ jobs
[1]-  Stopped                  sh loop.sh
[2]+  Stopped                  sh loop.sh
[3]   Running                  sh loop.sh &
artem@artemserver:~$ bg %
[2]+  sh loop.sh &
artem@artemserver:~$ jobs
[1]+  Stopped                  sh loop.sh
[2]   Running                  sh loop.sh &
[3]-  Running                  sh loop.sh &
artem@artemserver:~$ _
```

Рисунок 16 – Перевод задачи выполняющейся в интерактивном режиме, в фоновый режим

Далее проведём эксперименты по переводу задач из фонового режима в интерактивный и наоборот. Результат работы представлен на рисунке 17.

```
artem@artemserver:~$ jobs
[1]+  Stopped                  sh loop.sh
[2]   Running                  sh loop.sh &
[3]-  Running                  sh loop.sh &
artem@artemserver:~$ fg %2
sh loop.sh
^Z
[2]+  Stopped                  sh loop.sh
artem@artemserver:~$ jobs
[1]-  Stopped                  sh loop.sh
[2]+  Stopped                  sh loop.sh
[3]   Running                  sh loop.sh &
artem@artemserver:~$ bg %2
[2]+  sh loop.sh &
artem@artemserver:~$ jobs
[1]+  Stopped                  sh loop.sh
[2]   Running                  sh loop.sh &
[3]-  Running                  sh loop.sh &
artem@artemserver:~$ _
```

Рисунок 17 – Эксперименты по переводу задач

Нужно создать именованный канал для архивирования и осуществить передачу в канал списка файлов домашнего каталога вместе с подкаталогами (ключ -R), одного каталога вместе с файлами и подкаталогами. Результат работы представлен на рисунках 18, 19, 20, 21 и 22.

```
artem@artemserver:~$ ls -l
total 12
-rw-rw-r-- 1 artem artem 27 Nov  5 11:00 loop
-rwxrwxr-x 1 artem artem 40 Nov  6 09:03 loop2.sh
-rwxrwxr-x 1 artem artem 26 Nov  6 09:02 loop.sh
artem@artemserver:~$ mkfifo channel
artem@artemserver:~$ ls -l
total 12
prw-rw-r-- 1 artem artem  0 Nov  6 09:49 channel
-rw-rw-r-- 1 artem artem 27 Nov  5 11:00 loop
-rwxrwxr-x 1 artem artem 40 Nov  6 09:03 loop2.sh
-rwxrwxr-x 1 artem artem 26 Nov  6 09:02 loop.sh
artem@artemserver:~$
```

Рисунок 18 – Создание именованного канала

```
artem@artemserver:~$ ls -l
total 12
prw-rw-r-- 1 artem artem  0 Nov  6 09:49 channel
-rw-rw-r-- 1 artem artem 27 Nov  5 11:00 loop
-rwxrwxr-x 1 artem artem 40 Nov  6 09:03 loop2.sh
-rwxrwxr-x 1 artem artem 26 Nov  6 09:02 loop.sh
artem@artemserver:~$ ls -R > channel&
[4] 921
artem@artemserver:~$ _
```

Рисунок 19 – Передача в канал листинг домашнего каталога

```
artem@artemserver:~$ mkdir new
artem@artemserver:~$ cd new
artem@artemserver:~/new$ touch 1.txt
artem@artemserver:~/new$ touch 2.txt
artem@artemserver:~/new$ mkdir new1
artem@artemserver:~/new$ cd new1
artem@artemserver:~/new/new1$ touch 3.txt
artem@artemserver:~/new/new1$ cd ..
artem@artemserver:~/new$ ls -l
total 4
-rw-rw-r-- 1 artem artem  0 Nov  6 10:01 1.txt
-rw-rw-r-- 1 artem artem  0 Nov  6 10:01 2.txt
drwxrwxr-x 2 artem artem 4096 Nov  6 10:02 new1
artem@artemserver:~/new$ _
```

Рисунок 20 – Создание нового каталога

```

artem@artemserver:~$ gzip -9 -c < channel > res
[4]- Done          ls --color=auto -R > channel
artem@artemserver:~$ zcat res
.:
channel
loop
loop2.sh
loop.sh
new

./new:
1.txt
2.txt
new1

./new/new1:
3.txt
artem@artemserver:~$ _

```

Рисунок 21 – Листинг домашнего каталога

```

artem@artemserver:~$ ls -l new/ > channel&
[4] 935
artem@artemserver:~$ gzip -9 -c < channel > res
[4]- Done          ls --color=auto -l new/ > channel
artem@artemserver:~$ zcat res
total 4
-rw-rw-r-- 1 artem artem    0 Nov  6 10:01 1.txt
-rw-rw-r-- 1 artem artem    0 Nov  6 10:01 2.txt
drwxrwxr-x 2 artem artem 4096 Nov  6 10:02 new1
artem@artemserver:~$

```

Рисунок 22 – Занесение в канал листинг созданного каталога и архивация содержимого каталога

Часть 3

Нужно получить следующую информацию о процессах текущего пользователя: идентификатор и имя владельца процесса, статус и приоритет процесса.

Для этого используем команду `top` - позволяет выводить информацию о системе, а также список процессов динамически обновляя информацию о потребляемых ими ресурсах.

- v - вывести версию программы;
- b - режим только для вывода данных, программа не воспринимает интерактивных команд и выполняется пока не будет завершена вручную;
- c - отображать полный путь к исполняемым файлам команд;
- d - интервал обновления информации;
- H - включает вывод потоков процессов;

- i - не отображать процессы, которые не используют ресурсы процессора;
- n - количество циклов обновления данных, после которых надо закрыть программу;
- o - поле, по которому надо выполнять сортировку;
- O - вывести все доступные поля для сортировки;
- p - отслеживать только указанные по PID процессы, можно указать несколько PID;
- u - выводить только процессы, запущенные от имени указанного пользователя.

```
artem@artemserver:~$ top -u artem
```

Рисунок 23 – использование команды top

```
Fields Management for window 1:Def, whose current sort field is %CPU
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
NU        = Last Used NUMA node
```

Рисунок 24 – настройка параметров вывода


```

top - 16:14:29 up 18 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 93 total, 1 running, 92 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, 572.1 free, 130.5 used, 278.6 buff/cache
MiB Swap: 1781.0 total, 1781.0 free, 0.0 used, 703.6 avail Mem

  PID USER      PR  SHR
  876 artem    20  8244
  877 artem    20    4
  882 artem    20  3528
  895 artem    20  3128

```

Рисунок 25 - получение информации о процессах текущего пользователя

Далее нужно завершить выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершить с помощью команды `kill -2` (сигнал `SIGINT`), второй – с помощью сигнала `SIGQUIT`, задав его номер.

```

artem@artemserver:~$ sh loop.sh&
[1] 949
artem@artemserver:~$ sh loop.sh&
[2] 950
artem@artemserver:~$ kill -2 949
artem@artemserver:~$ kill -3 950
[1]- Interrupt                  sh loop.sh
artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem        882     639  0 15:56 tty1        00:00:00 -bash
artem        952     882  0 17:16 tty1        00:00:00 ps -f
[2]+ Quit                      (core dumped) sh loop.sh
artem@artemserver:~$ _

```

Рисунок 26 – остановка процессов

Определим идентификаторы и имена процессов, идентификатор группы которых не равен идентификатору группы текущего пользователя.

```

artem@artemserver:~$ sudo ps -a -o pid,cmd,gid | grep -v 1000
  PID CMD                                GID
  975 sudo ps -a -o pid,cmd,gid          0
  977 ps -a -o pid,cmd,gid                0
artem@artemserver:~$ _

```

Рисунок 27 - Идентификаторы и имена процессов

Вывод

В ходе выполнения лабораторной работы я ознакомился на практике с понятием процесса в операционной системе, приобрел опыт и навыки управления процессами в операционной системе Linux.