

Липецкий государственный технический университет

Кафедра автоматизированных систем управления

Полное и сокращенное название кафедры

Лабораторная работа № 2

по

«Операционной системе Linux»

Наименование дисциплины

Понятие о процессах в ОС Linux

Наименование темы

Студент

Группа АИ-18-1

Грунау Г.Ю.

фамилия, инициалы

Руководитель
подпись

Кургасов В.В.

фамилия, инициалы

Учетная степень, учетное звание

Липецк 2020 г.

Содержание

Цель работы	3
Задание	4
Ход работы	6
1. Часть I.	6
2. Часть II.	12
3. Часть III.	16
Вывод	21

Цель работы

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

Задание

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.

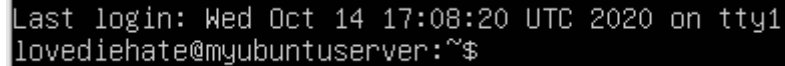
Вариант 2.

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

1. Часть I.

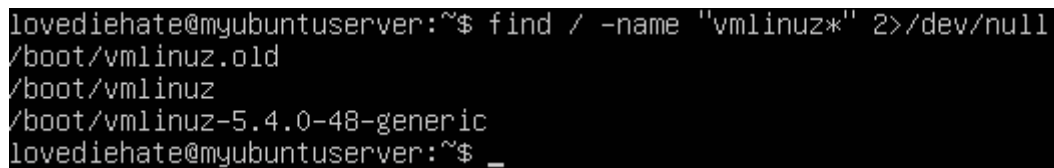
1. Загрузка пользователем, а не root



```
Last login: Wed Oct 14 17:08:20 UTC 2020 on tty1
lovediehate@myubuntuserver:~$
```

Рисунок 1 – Загрузка пользователем

2. Поиск файла с образом ядра

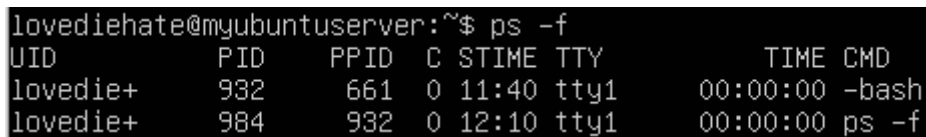


```
lovediehate@myubuntuserver:~$ find / -name "vmlinuz*" 2>/dev/null
/boot/vmlinuz.old
/boot/vmlinuz
/boot/vmlinuz-5.4.0-48-generic
lovediehate@myubuntuserver:~$ _
```

Рисунок 2 – Поиск образа ядра

На Рисунок 2 видно версию ядра – 5.4.0-48-generic.

3. Просмотр процессов ps -f



```
lovediehate@myubuntuserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
lovedie+     932     661  0 11:40 tty1        00:00:00 -bash
lovedie+     984     932  0 12:10 tty1        00:00:00 ps -f
```

Рисунок 3 – Информация о процессах

На Рисунке 3 изображено использование команды ps -f. Параметр -f означает показ полного списка процессов.

- UID – идентификатор юзера, при параметре -f показывается входное его имя.
- PID – идентификатор процесса.
- PPID – идентификатор родительского процесса.
- C – доля выделенного планировщиком времени процессора.
- STIME – время или дата запуска процесса.
- TTY – управляющий терминал.
- TIME – истраченное процессом время процессора.
- CMD – имя программы и её аргументы.

4. Написание двух сценариев с помощью редактора vi

```
"loop" [New] 1L, 26C written
lovedie+@myubuntuserver:~/lr2$ cat loop
while true; do true; done
```

Рисунок 4 – Первый сценарий loop

```
"loop2" [New] 1L, 40C written
lovedie+@myubuntuserver:~/lr2$ cat loop2
while true; do true; echo 'Hello'; done
```

Рисунок 5 – Второй сценарий loop2

5. Запуск loop2 на переднем плане

```
Hello
Hello
Hello
Hello
Hello
^Z
[7]+  Stopped                  sh loop2
```

Рисунок 6 – Запуск и остановка процесса

6. Остановка процесса сигналом STOP

На Рисунке 6 изображена остановка активного процесса сигналом STOP нажатием клавиш Ctrl+Z.

7. Последовательный запуск ps -f

```
lovedie+@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
lovedie+    1385    1265  0 12:49 tty1        00:00:00 -bash
lovedie+    1409    1385  0 12:51 tty1        00:00:00 sh loop2
lovedie+    1411    1385  0 12:53 tty1        00:00:00 ps -f
lovedie+@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
lovedie+    1385    1265  0 12:49 tty1        00:00:00 -bash
lovedie+    1409    1385  0 12:51 tty1        00:00:00 sh loop2
lovedie+    1412    1385  0 12:54 tty1        00:00:00 ps -f
lovedie+@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
lovedie+    1385    1265  0 12:49 tty1        00:00:00 -bash
lovedie+    1409    1385  0 12:51 tty1        00:00:00 sh loop2
lovedie+    1413    1385  0 12:54 tty1        00:00:00 ps -f
lovedie+@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
lovedie+    1385    1265  0 12:49 tty1        00:00:00 -bash
lovedie+    1409    1385  0 12:51 tty1        00:00:00 sh loop2
lovedie+    1414    1385  0 12:54 tty1        00:00:00 ps -f
```

Рисунок 7 – Поочередный запуск ps -f

На Рисунке 7 видно, что при последовательном запуске программы идентификатор процесса и время запуска меняется. Это может говорить о том, что каждый раз запускается новый процесс, которому присваивается новый свободный идентификатор. При этом, родительский идентификатор не меняется: все процессы вызываются из одного интерпретатора –bash.

8. Убийство процесс loop2

```
lovedie+@myubuntuserver:~/lr2$ kill -9 1409  
[1]+  Killed                  sh loop2
```

Рисунок 8 – Убийство

```
lovedie+@myubuntuserver:~/lr2$ ps -f  
UID          PID     PPID  C  STIME TTY          TIME CMD  
lovedie+    1385     1265  0  12:49 tty1          00:00:00 -bash  
lovedie+    1422     1385  0  13:20 tty1          00:00:00 ps -f
```

Рисунок 9 – Просмотр процессов

На Рисунке 8 с помощью команды `kill -9 1409` был послан сигнал KILL процессу 1409. Этот сигнал означает немедленное принудительное завершение процесса без возможности перехвата этого сигнала.

На Рисунке 9 видим, что процесса с PID 1409 больше нет.

9. Запуск на фоне loop

```
lovediehate@myubuntuserver:~/lr2$ sh loop&
[1] 1429
lovediehate@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
lovedie+    1385    1265  0  12:49 tty1        00:00:00 -bash
lovedie+    1429    1385 52  13:37 tty1        00:00:01 sh loop
lovedie+    1430    1385  0  13:37 tty1        00:00:00 ps -f
lovediehate@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
lovedie+    1385    1265  0  12:49 tty1        00:00:00 -bash
lovedie+    1429    1385 60  13:37 tty1        00:00:01 sh loop
lovedie+    1431    1385  0  13:37 tty1        00:00:00 ps -f
lovediehate@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
lovedie+    1385    1265  0  12:49 tty1        00:00:00 -bash
lovedie+    1429    1385 53  13:37 tty1        00:00:02 sh loop
lovedie+    1432    1385  0  13:37 tty1        00:00:00 ps -f
lovediehate@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
lovedie+    1385    1265  0  12:49 tty1        00:00:00 -bash
lovedie+    1429    1385 50  13:37 tty1        00:00:12 sh loop
lovedie+    1433    1385  0  13:37 tty1        00:00:00 ps -f
```

Рисунок 10 – Запуск фонового процесса

На Рисунке 10 заметно изменение значений в колонке C и TIME. Изменение в колонке C означает, что меняется % приоритета CPU, используемого процессом. TIME показывает, как долго запущен процесс. Последний вызов ps -f был через 12 секунд после запуска процесса 1429.

10. Завершение процесса командой kill -15

```
lovediehate@myubuntuserver:~/lr2$ kill -15 1445
lovediehate@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
lovedie+    1385    1265  0  12:49 tty1        00:00:00 -bash
lovedie+    1447    1385  0  14:07 tty1        00:00:00 ps -f
[1]+  Terminated                  sh loop
```

Рисунок 11 – Завершение процесса

На Рисунке 11 видно, что процесс завершён. Также в нижней строке мы видим Terminated sh loop, что может говорить нам о корректном завершении процесса.

11. Запуск в фоновом режиме и убийство процесса

```
lovediegate@myubuntuserver:~/lr2$ sh loop&
[1] 1452
lovediegate@myubuntuserver:~/lr2$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
lovedie+    1385     1265  0  12:49 tty1        00:00:00 -bash
lovedie+    1452     1385 49  14:20 tty1        00:00:11 sh loop
lovedie+    1454     1385  0  14:21 tty1        00:00:00 ps -f
lovediegate@myubuntuserver:~/lr2$ kill -9 1452
lovediegate@myubuntuserver:~/lr2$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
lovedie+    1385     1265  0  12:49 tty1        00:00:00 -bash
lovedie+    1455     1385  0  14:21 tty1        00:00:00 ps -f
[1]+  Killed                  sh loop
```

Рисунок 12 – Запуск и убийство

На Рисунке 12 видно, как файл loop запускается фоном и убивается командой kill -9. При первом просмотре моментальной съёмки статуса процессов выводится сообщение о том, что процесс успешно убит (см. последнюю строку на Рисунке 12).

12. Запуск оболочки bash

```
lovediegate@myubuntuserver:~/lr2$ kill -19 940
lovediegate@myubuntuserver:~/lr2$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
lovedie+    908     649  0  14:34 tty1        00:00:00 -bash
lovedie+    920     908  0  14:46 tty1        00:00:00 bash
lovedie+    940     920 41  14:48 tty1        00:04:08 sh loop
lovedie+    942     920 30  14:49 tty1        00:02:59 sh loop
lovedie+    951     920  0  14:58 tty1        00:00:00 ps -f

[1]+  Stopped                  sh loop
lovediegate@myubuntuserver:~/lr2$ kill -19 942
lovediegate@myubuntuserver:~/lr2$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
lovedie+    908     649  0  14:34 tty1        00:00:00 -bash
lovedie+    920     908  0  14:46 tty1        00:00:00 bash
lovedie+    940     920 40  14:48 tty1        00:04:08 sh loop
lovedie+    942     920 31  14:49 tty1        00:03:04 sh loop
lovedie+    952     920  0  14:59 tty1        00:00:00 ps -f

[2]+  Stopped                  sh loop
```

Рисунок 13 – Снимок процессов

На Рисунке 13 видно, что запущена оболочка bash с PID 920. Она будет родительским процессом последующих вызываемых процессов.

13. Запуск и остановка нескольких процессов

На Рисунке 13 видно, что запущено два процесса `sh loop`, и при остановке этих процессов выводится следующее сообщение: `[1]+ Stopped sh loop` – остановлен первый процесс `sh loop`.

```
lovediehate@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
lovedie+     908      649  0  14:34 tty1        00:00:00 -bash
lovedie+     920      908  0  14:46 tty1        00:00:00 bash
lovedie+     940      920  22  14:48 tty1        00:04:08 sh loop
lovedie+     942      920  16  14:49 tty1        00:03:04 sh loop
lovedie+     954      920  0  15:07 tty1        00:00:00 ps -f
```

Рисунок 14 – Снимок процессов

На Рисунке 14 изображен снимок процессов через 8 минут после предыдущего. Можно увидеть, что время работы остановленных процессов не изменилось.

```
lovediehate@myubuntuserver:~/lr2$ kill -18 940
lovediehate@myubuntuserver:~/lr2$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
lovedie+     908      649  0  14:34 tty1        00:00:00 -bash
lovedie+     920      908  0  14:46 tty1        00:00:00 bash
lovedie+     940      920  18  14:48 tty1        00:04:12 sh loop
lovedie+     942      920  13  14:49 tty1        00:03:04 sh loop
lovedie+     957      920  0  15:11 tty1        00:00:00 ps -f
```

Рисунок 15 – Продолжение процесса

На Рисунке 15 заметно, что при продолжении процесса (команда `kill -s CONT [PID]` или `kill -15 [PID]`) работа возобновляется.

2. Часть II.

1. Запуск в консоли на выполнение трёх задач, двух в интерактивном режиме, одной - в фоновом

Использованные команды:

- jobs – просмотр текущих задач

```
lovediehate@myubuntuserver:~/lr2$ jobs
[1]-  Stopped                  sh loop
[2]   Running                  sh loop &
[3]+  Stopped                  sh loop
```

Рисунок 16 – Задачи

На Рисунке 16 показано, что запущено 3 задачи: 2 интерактивные и 1 фоновая. Интерактивные задачи пришлось приостановить.

2. Перевод интерактивной задачи в фоновую

Использованные команды:

- bg 3 – перевод задачи в фон

```
lovediehate@myubuntuserver:~/lr2$ jobs
[1]-  Stopped                  sh loop
[2]   Running                  sh loop &
[3]+  Stopped                  sh loop
lovediehate@myubuntuserver:~/lr2$ bg 3
[3]+  sh loop &
lovediehate@myubuntuserver:~/lr2$ jobs
[1]+  Stopped                  sh loop
[2]   Running                  sh loop &
[3]-  Running                  sh loop &
```

Рисунок 17 – Перевод задачи

На Рисунке 17 изображено использование команды bg для перевода задачи 3 в фоновый режим. На скриншоте видно, что задача 3 стала фоновой (появился амперсанд и задача запустилась).

3. Перевод из задачи из фонового режима в активный

Использованные команды:

- `fg 2` – перевод задачи на передний план

```
lovediehatemyubuntuserver:~/lr2$ jobs
[1]+  Stopped                  sh loop
[2]   Running                  sh loop &
[3]-  Running                  sh loop &
lovediehatemyubuntuserver:~/lr2$ fg 2
sh loop
^Z
[2]+  Stopped                  sh loop
lovediehatemyubuntuserver:~/lr2$ jobs
[1]-  Stopped                  sh loop
[2]+  Stopped                  sh loop
[3]   Running                  sh loop &
```

Рисунок 18 – Перевод задачи

На Рисунке 18 с помощью команды `fg` задача 2 переведена в интерактивные. Для того, чтобы сделать снимок задач, пришлось приостановить действие процесса (^Z). Далее видно, что задача [2] переведена на передний план и приостановлена.

```
lovediehatemyubuntuserver:~/lr2$ bg 1
[1]-  sh loop &
lovediehatemyubuntuserver:~/lr2$ bg 2
[2]+  sh loop &
lovediehatemyubuntuserver:~/lr2$ jobs
[1]   Running                  sh loop &
[2]-  Running                  sh loop &
[3]+  Running                  sh loop &
```

Рисунок 19 – Перевод всех задач на фон

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.

Использованные команды:

- `mkfifo fifo` (создание канала)
- `ls -l` (просмотр файлов в текущем каталоге)

```
lovediehate@myubuntuserver:~$ mkfifo fifo
lovediehate@myubuntuserver:~$ ls -l
total 4
prw-rw-r-- 1 lovediehate lovediehate  0 Oct 30 18:48 fifo
drwxrwxr-x 3 lovediehate lovediehate 4096 Oct 29 20:14 lr2
```

Рисунок 20 – Создание именованного канала

Буква `r` в начале говорит о том, что файл имеет тип именованный канал.

```
lovediehate@myubuntuserver:~$ ls -R > fifo &
[3] 992
lovediehate@myubuntuserver:~$ ls lr2/ > fifo &
[4] 993
```

Рисунок 21 – Внесение списка и каталога в канал

Использованные команды:

- `gzip -l -c < fifo > my_arch` – настройка канала `fifo`

```
lovediehate@myubuntuserver:~$ jobs
[1]  Stopped                  ls --color=auto -l new/ > somename
[2]  Stopped                  ls --color=auto -R > A
[3]- Stopped                  "your command" | less
[4]+ Stopped                  "your_command" | less
[5]  Running                  ls --color=auto -R > fifo &
[6]  Running                  ls --color=auto lr2/ > fifo &
lovediehate@myubuntuserver:~$ gzip -l -c < fifo > my_arch
[5]  Done                     ls --color=auto -R > fifo
[6]  Done                     ls --color=auto lr2/ > fifo
lovediehate@myubuntuserver:~$ _
```

Рисунок 22 – Архивирование

На Рисунке 23 видно, что задачи 5 и 6 завершены успешно. Канал `fifo` был настроен на сжатие того, что туда попадёт и вывод в файл `my_arch`. Задачи 5 и 6 передали данные в канал для архивации.

Использованные команды:

- `zcat my_arch` – просмотр содержимого сжатого архива

```
lovediehat@myubuntuserver:~$ zcat my_arch
dir
fif
filen.tar.gz
filename
foo.in
foo.out
foo.pipe
loop
loop2
my-file-pipe
out
.:
arch
fifo
lr2

./lr2:
dir
fif
filen.tar.gz
filename
foo.in
foo.out
foo.pipe
loop
loop2
my-file-pipe
out
```

Рисунок 22 – Просмотр содержимого сжатого файла

Как можно увидеть на рисунке 25 – рекурсивный список файлов домашнего каталога и отдельный каталог успешно передались в `fifo`.

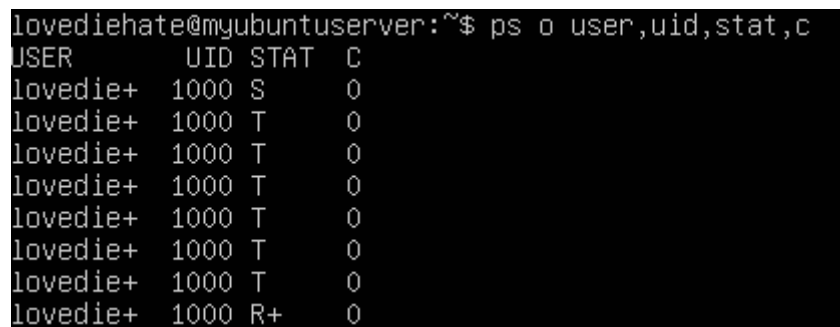
3. Часть III.

Вариант 2.

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

Использована команда:

- `ps o user,uid,stat,c`



```
lovediehate@myubuntuserver:~$ ps o user,uid,stat,c
USER      UID  STAT  C
lovedie+  1000  S      0
lovedie+  1000  T      0
lovedie+  1000  T      0
lovedie+  1000  T      0
lovedie+  1000  T      0
lovedie+  1000  T      0
lovedie+  1000  T      0
lovedie+  1000  T      0
lovedie+  1000  T      0
lovedie+  1000  R+     0
```

Рисунок 23 – Информация о процессах

На рисунке 23 выведена нужная информация по процессам: **USER** – имя владельца процесса, **UID** – ид владельца, **STAT** – статус процесса и **C** – приоритет процесса в планировщике.

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.

Использована команда:

- `kill -2 -f` (отправка процессу сигнала прерывания по имени)

```
lovedie@myubuntuserver:~/lr2$ kill -2 -f 'sh loop'
lovedie@myubuntuserver:~/lr2$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
lovedie+      925       658  0  18:12 tty1        00:00:00 -bash
lovedie+      942       925  0  18:14 tty1        00:00:00 -bash
lovedie+      946       925  0  18:16 tty1        00:00:00 -bash
lovedie+     1008       925  0  19:07 tty1        00:00:00 less
lovedie+     1020       925  0  19:07 tty1        00:00:00 less
lovedie+     1183       925  0  19:50 tty1        00:00:00 top
lovedie+     1185       925  0  19:51 tty1        00:00:00 top
lovedie+     1307       925  0  20:55 tty1        00:00:00 ps -f
[7]  Interrupt                  sh loop
```

Рисунок 24 – Прерывание процесса

На Рисунке 24 заметно, что процесс `sh loop` завершил работу [Interrupt]. Параметр `-f` значит искать по полному названию файла, т.к. название с пробелом.

Используемые команды:

- kill -3 (отправка процессу сигнала 3 [QUIT])

```
lovedie+@myubuntuserver:~/lr2$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
lovedie+      925       658  0  18:12 tty1        00:00:00 -bash
lovedie+      942       925  0  18:14 tty1        00:00:00 -bash
lovedie+      946       925  0  18:16 tty1        00:00:00 -bash
lovedie+     1008       925  0  19:07 tty1        00:00:00 less
lovedie+     1020       925  0  19:07 tty1        00:00:00 less
lovedie+     1183       925  0  19:50 tty1        00:00:00 top
lovedie+     1185       925  0  19:51 tty1        00:00:00 top
lovedie+     1308       925  94  20:57 tty1        00:01:19 sh loop
lovedie+     1312       925  50  20:58 tty1        00:00:05 sh loop
lovedie+     1314       925  0  20:58 tty1        00:00:00 ps -f
lovedie+@myubuntuserver:~/lr2$ kill -3 1312
lovedie+@myubuntuserver:~/lr2$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
lovedie+      925       658  0  18:12 tty1        00:00:00 -bash
lovedie+      942       925  0  18:14 tty1        00:00:00 -bash
lovedie+      946       925  0  18:16 tty1        00:00:00 -bash
lovedie+     1008       925  0  19:07 tty1        00:00:00 less
lovedie+     1020       925  0  19:07 tty1        00:00:00 less
lovedie+     1183       925  0  19:50 tty1        00:00:00 top
lovedie+     1185       925  0  19:51 tty1        00:00:00 top
lovedie+     1308       925  72  20:57 tty1        00:02:07 sh loop
lovedie+     1316       925  0  21:00 tty1        00:00:00 ps -f
[8]  Quit                               (core dumped) sh loop
```

Рисунок 25 – Прерывание процесса 2

На последней строчке консоли, представленной на рисунке 25, видно, что процесс завершён с сообщением Quit, как мы и хотели.

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

Использована команда:

- `id` – выводит идентификатор текущего пользователя

```
lovediehate@myubuntuserver:~/lr2$ id  
uid=1000(lovediehate) gid=1000(lovediehate) groups=1000(lovediehate)
```

Рисунок 26 – Информация о пользователе

На Рисунке 26 с помощью утилиты `id` узнаётся, какой идентификатор у текущего пользователя. В дальнейшем мы используем это значение для отрицания в поиске.

Использована команда:

- `ps -afx --format="comm pid uid uname gid group" | grep -v 1000`
(вывод информации по процессу в заданном формате, исключая из результата строки, в которых попадает значение группы текущего пользователя)

```
lovediehate@myubuntuserver:~/lr2$ ps -afx --format="comm pid uid uname gid group" | grep -v 1000
```

Рисунок 27 – Строка ввода команды

COMMAND	PID	UID	USER	GID	GROUP
kthreadd	2	0	root	0	root
_ rcu_gp	3	0	root	0	root
_ rcu_par_gp	4	0	root	0	root
_ kworker/0:0	6	0	root	0	root
_ mm_percpu_w	9	0	root	0	root
_ ksoftirqd/0	10	0	root	0	root
_ rcu_sched	11	0	root	0	root
_ migration/0	12	0	root	0	root
_ idle_inject	13	0	root	0	root
_ cpuhp/0	14	0	root	0	root
_ kdevtmpfs	15	0	root	0	root
_ netns	16	0	root	0	root
_ rcu_tasks_k	17	0	root	0	root
_ kauditd	18	0	root	0	root
_ khungtaskd	19	0	root	0	root
_ oom_reaper	20	0	root	0	root
_ writeback	21	0	root	0	root
_ kcompactd0	22	0	root	0	root
_ ksmd	23	0	root	0	root
_ khugepaged	24	0	root	0	root
_ kintegrityd	70	0	root	0	root
_ kblockd	71	0	root	0	root
_ blkcg_punt_	72	0	root	0	root
_ tpm_dev_wq	73	0	root	0	root
_ ata_sff	74	0	root	0	root
_ md	75	0	root	0	root
_ edac-poller	76	0	root	0	root
_ devfreq_wq	77	0	root	0	root
_ watchdogd	78	0	root	0	root
_ kswapd0	83	0	root	0	root
_ ecryptfs-kt	84	0	root	0	root
_ kthrotld	86	0	root	0	root
_ acpi_therma	87	0	root	0	root
_ scsi_eh_0	88	0	root	0	root
_ scsi_tmf_0	89	0	root	0	root

Рисунок 28 – Результат

На рисунке 27 выведены процессы, GID которых отличается от GID текущего пользователя.

Вывод

Я ознакомился с понятием процесса в ОС Linux и приобрел базовые навыки работы с ним.