

Липецкий государственный технический университет

Факультет автоматизации и информатики
Кафедра автоматизированных систем управления

Отчет по лабораторной работе № 3
«Процессы в операционной системе Linux»
по курсу «Операционная система Linux»

Студент

подпись, дата

Закиров Р.Р.

фамилия, инициалы

Группа

Руководитель

Доцент, к. пед. наук

ученая степень, ученое звание

подпись, дата

Кургасов В.В.

фамилия, инициалы

Липецк 2022 г.

Содержание

Цель работы	3
Задание кафедры	4
1. Часть I	7
2. Часть II	14
3. Часть III	18
Выводы	20
Контрольные вопросы	21

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. Индивидуальные задания (Вариант 3):

1. Сгенерировать следующую информацию о m ($m > 2$) процессах системы, имеющих значение идентификатора больше заданного n : флаг – сведения о процессе, статус, PID, PPID, приоритет, использованное время и имя программы.
2. Завершить выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершить с помощью сигнала SIGKILL, задав его имя, второй – с помощью сигнала SIGINT, 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.

1. Часть I

Задание:

1. Загрузиться не root, а пользователем.

```
Debian GNU/Linux 11 debian tty1
debian login: rolan
Password:
Linux debian 5.10.0-19-amd64 #1 SMP Debian 5.10.149-2 (2022-10-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Nov 28 22:07:24 MSK 2022 on tty1
rolan@debian:~$ _
```

Рисунок 1 – Загрузка не root, а пользователем.

2. Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.

```
rolan@debian:~$ cd /
rolan@debian:/$ ls
bin    etc      initrd.img.old  lib64      media  proc  sbin  sys  var
boot  home     lib             libx32     mnt    root  slink tmp  vmlinuz
dev    initrd.img lib32           lost+found opt     run   srv   usr  vmlinuz.old
rolan@debian:/$ cd boot
rolan@debian:/boot$ ls
config-5.10.0-18-amd64  initrd.img-5.10.0-18-amd64  System.map-5.10.0-19-amd64
config-5.10.0-19-amd64  initrd.img-5.10.0-19-amd64  vmlinuz-5.10.0-18-amd64
grub                    System.map-5.10.0-18-amd64  vmlinuz-5.10.0-19-amd64
```

Рисунок 2 – Поиск файла с образом ядра.

Исходя из рис.2 делаем вывод, что номер версии Linux - 5.10.0.

3. Посмотреть процессы `ps -f`. Прокомментировать. Для этого почитать `man ps`.

```
rolan@debian:/boot$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
rolan         540      393  0 18:17 tty1        00:00:00 -bash
rolan        1010     1009  0 22:28 tty1        00:00:00 -bash
rolan        1357     1356  0 23:21 tty1        00:00:00 -bash
rolan        1408     1357  0 23:48 tty1        00:00:00 ps -f
```

Рисунок 3 – Просмотр процессов `ps -f`.

- UID – идентификатор пользователя.
- PID – идентификатор процесса. Он принудительно назначается планировщиком при запуске процесса.
- PPID – идентификатор родительского процесса.
- C – численное значение расходования ресурсов процессора в процентах.
- STIME – это время начала процесса.
- TTY – имя управляющего терминала - терминала, с которого запущен процесс.
- TIME – это общее время использования процессорного времени процессом.
- CMD – команда, которой был запущен процесс, если программа не может прочитать аргументы процесса, он будет выведен в квадратных скобках.

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`

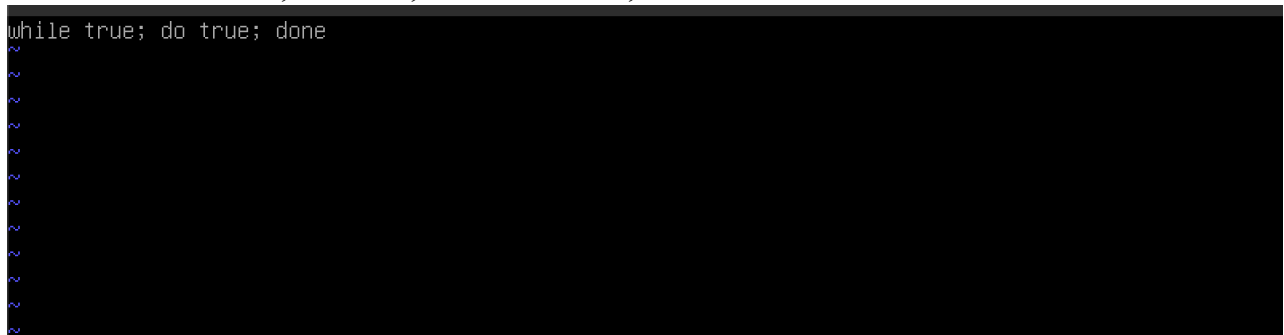
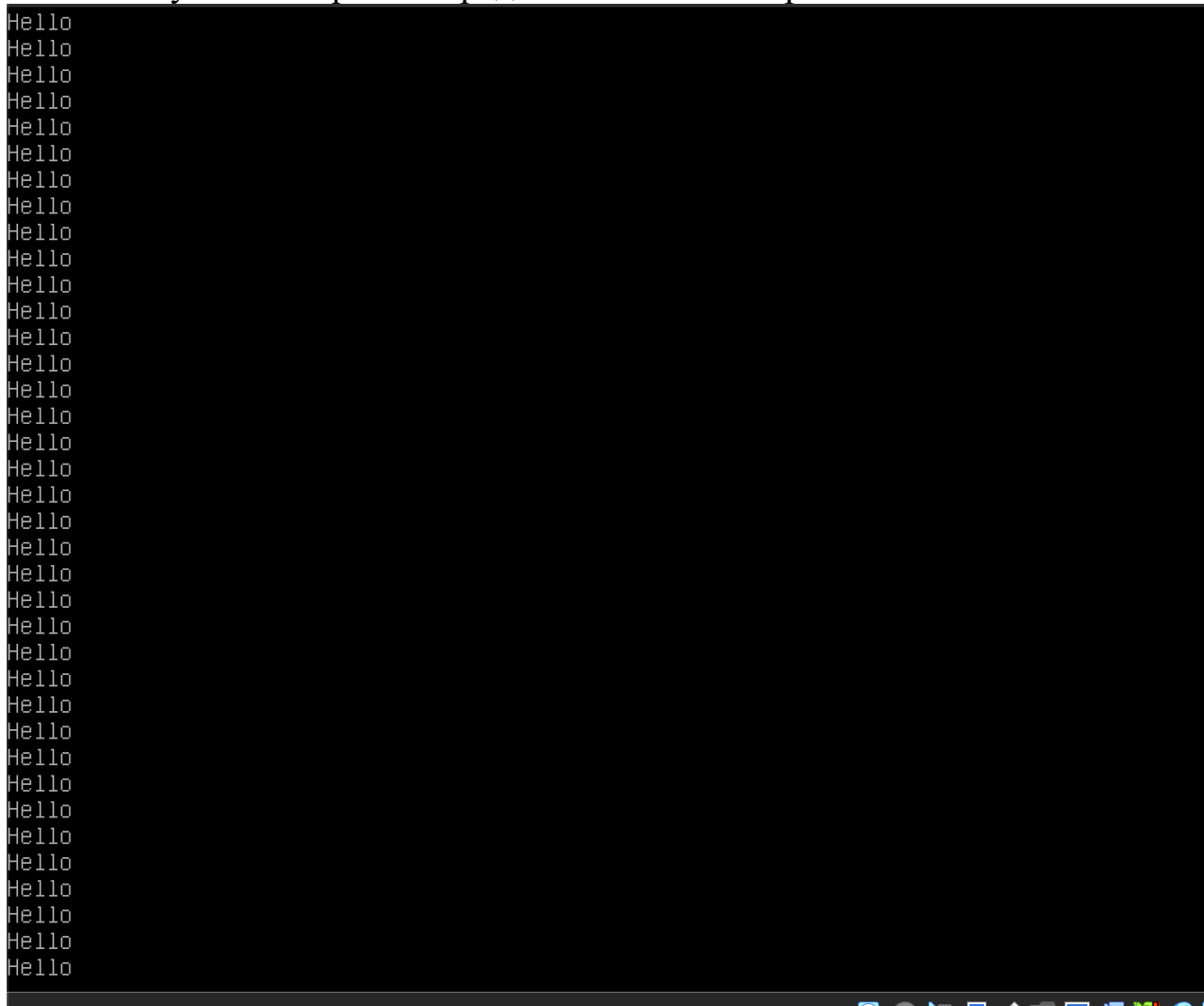


Рисунок 4 – Открытие редактора vi для файла loop.sh.

Аналогичным образом, записываем для файла loop2.sh

5. Запустить loop2 на переднем плане: `sh loop2`.

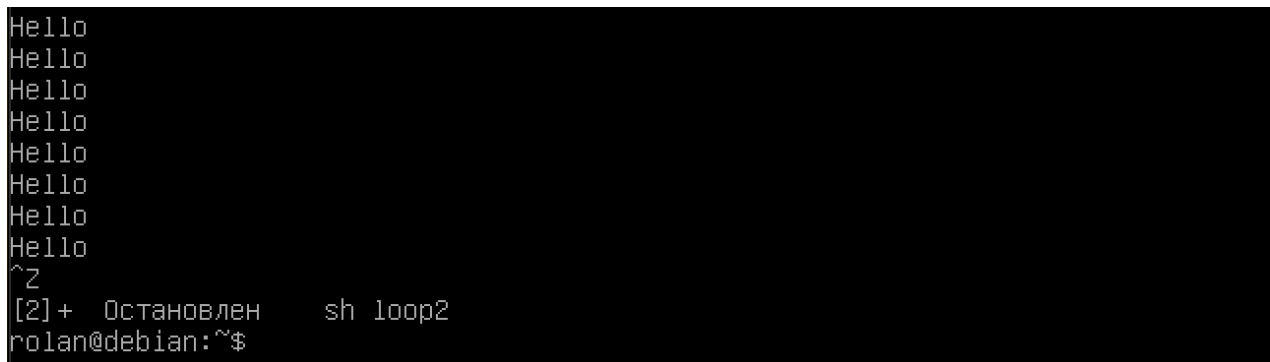
A terminal window with a black background and white text. The text consists of 25 lines, each starting with the word "Hello". The terminal is running a shell script that prints "Hello" repeatedly. The window has a standard Linux desktop environment at the bottom with various icons.

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Рисунок 5 – Запуск loop2 на переднем плане: `sh loop2.sh`.

6. Остановить, послав сигнал STOP.

С помощью сигнала STOP останавливаем (рис. 6).

A terminal window with a black background and white text. It shows the same "Hello" messages as in Figure 5, followed by a control character (^Z), a prompt for a signal ([2]+), and the message "Остановлен sh loop2". The user's prompt "rolan@debian:~\$" is visible at the bottom.

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
^Z
[2]+  Остановлен  sh loop2
rolan@debian:~$
```

Рисунок 6 – Остановка сигналом STOP.

7. Посмотреть последовательно несколько раз `ps -f`. Записать сообщение, объяснить.

```
rolan@debian:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
rolan         540      393  0  14:33 tty1        00:00:00 -bash
rolan        1010     1009  0  18:43 tty1        00:00:00 -bash
rolan        1357     1356  0  19:37 tty1        00:00:00 -bash
rolan        1413     1357  0  20:11 tty1        00:00:00 vi loop
rolan        1730     1357 40  20:46 tty1        00:00:54 sh loop2
rolan        1732     1357  0  20:48 tty1        00:00:00 ps -f
rolan@debian:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
rolan         540      393  0  14:33 tty1        00:00:00 -bash
rolan        1010     1009  0  18:43 tty1        00:00:00 -bash
rolan        1357     1356  0  19:37 tty1        00:00:00 -bash
rolan        1413     1357  0  20:11 tty1        00:00:00 vi loop
rolan        1730     1357 37  20:46 tty1        00:00:54 sh loop2
rolan        1733     1357  0  20:49 tty1        00:00:00 ps -f
rolan@debian:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
rolan         540      393  0  14:33 tty1        00:00:00 -bash
rolan        1010     1009  0  18:43 tty1        00:00:00 -bash
rolan        1357     1356  0  19:37 tty1        00:00:00 -bash
rolan        1413     1357  0  20:11 tty1        00:00:00 vi loop
rolan        1730     1357 35  20:46 tty1        00:00:54 sh loop2
rolan        1734     1357  0  20:49 tty1        00:00:00 ps -f
rolan@debian:~$ _
```

Рисунок 7 – Просмотр `ps -f` последовательно.

На рисунке изображен последовательный просмотр списка процессов с помощью команды `ps -f`. Процессы `bash` и `loop2` не изменяют свой PID, в отличие от процессов, вызываемых командами `ps -f`. Это значит, что каждый такой вызов запускает новый процесс и завершает его после вывода результата команды на экран.

8. Убить процесс loop2, послав сигнал kill -9 PID. Записать сообщение. Прокомментировать.

```
rolan@debian:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
rolan      540       393    0 14:33 tty1        00:00:00 -bash
rolan     1010      1009    0 18:43 tty1        00:00:00 -bash
rolan     1357      1356    0 19:37 tty1        00:00:00 -bash
rolan     1413      1357    0 20:11 tty1        00:00:00 vi loop
rolan     1730      1357   35 20:46 tty1        00:00:54 sh loop2
rolan     1734      1357    0 20:49 tty1        00:00:00 ps -f
rolan@debian:~$ kill -9 1730
rolan@debian:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
rolan      540       393    0 14:33 tty1        00:00:00 -bash
rolan     1010      1009    0 18:43 tty1        00:00:00 -bash
rolan     1357      1356    0 19:37 tty1        00:00:00 -bash
rolan     1413      1357    0 20:11 tty1        00:00:00 vi loop
rolan     1736      1357    0 20:55 tty1        00:00:00 ps -f
[2]+  убито          sh loop2
rolan@debian:~$
```

Рисунок 8 – Уничтожение процесса loop2.sh при помощи сигнала kill -9 PID.

На рисунке 8 представлено удаление процесса и сообщение об успешном удалении.

9. Запустить в фоне процесс loop: sh loop&. Не останавливая, посмотреть несколько раз: ps -f. Записать значение, объяснить.

```
rolan@debian:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
rolan      540       393    0 14:33 tty1        00:00:00 -bash
rolan     1010      1009    0 18:43 tty1        00:00:00 -bash
rolan     1357      1356    0 19:37 tty1        00:00:00 -bash
rolan     1413      1357    0 20:11 tty1        00:00:00 vi loop
rolan     1737      1357   93 20:57 tty1        00:00:05 sh loop
rolan     1739      1357    0 20:57 tty1        00:00:00 ps -f
rolan@debian:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
rolan      540       393    0 14:33 tty1        00:00:00 -bash
rolan     1010      1009    0 18:43 tty1        00:00:00 -bash
rolan     1357      1356    0 19:37 tty1        00:00:00 -bash
rolan     1413      1357    0 20:11 tty1        00:00:00 vi loop
rolan     1737      1357   97 20:57 tty1        00:00:12 sh loop
rolan     1740      1357    0 20:57 tty1        00:00:00 ps -f
rolan@debian:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
rolan      540       393    0 14:33 tty1        00:00:00 -bash
rolan     1010      1009    0 18:43 tty1        00:00:00 -bash
rolan     1357      1356    0 19:37 tty1        00:00:00 -bash
rolan     1413      1357    0 20:11 tty1        00:00:00 vi loop
rolan     1737      1357   96 20:57 tty1        00:00:16 sh loop
rolan     1741      1357    0 20:57 tty1        00:00:00 ps -f
rolan@debian:~$ _
```

Рисунок 9 – Запуск в фоне процесса loop: sh loop.sh&.

На основе рисунка 9 можно сделать вывод, что доля ресурсов, затраченных процессором, не уменьшается с течением времени. Из этого делаем заключение, что процесс работает.

10. Завершить процесс loop командой kill -15 PID. Записать сообщение, прокомментировать.

```
rolan@debian:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
rolan         540      393  0  14:33 tty1        00:00:00 -bash
rolan        1010     1009  0  18:43 tty1        00:00:00 -bash
rolan        1357     1356  0  19:37 tty1        00:00:00 -bash
rolan        1413     1357  0  20:11 tty1        00:00:00 vi loop
rolan        1737     1357 96  20:57 tty1        00:00:16 sh loop
rolan        1741     1357  0  20:57 tty1        00:00:00 ps -f
rolan@debian:~$ kill -15 1737
rolan@debian:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
rolan         540      393  0  14:33 tty1        00:00:00 -bash
rolan        1010     1009  0  18:43 tty1        00:00:00 -bash
rolan        1357     1356  0  19:37 tty1        00:00:00 -bash
rolan        1413     1357  0  20:11 tty1        00:00:00 vi loop
rolan        1742     1357  0  20:59 tty1        00:00:00 ps -f
[2]-  Завершено      sh loop
rolan@debian:~$ _
```

Рисунок 10 – Завершение процесса loop командой kill -15 PID.

11. Третий раз запустить в фоне. Не останавливая убить командой kill -9 PID.

```
rolan@debian:~$ sh loop&
[2] 1744
rolan@debian:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
rolan         540      393  0  14:33 tty1        00:00:00 -bash
rolan        1010     1009  0  18:43 tty1        00:00:00 -bash
rolan        1357     1356  0  19:37 tty1        00:00:00 -bash
rolan        1413     1357  0  20:11 tty1        00:00:00 vi loop
rolan        1744     1357 95  21:02 tty1        00:00:03 sh loop
rolan        1745     1357  0  21:02 tty1        00:00:00 ps -f
rolan@debian:~$ kill -9 1744
rolan@debian:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
rolan         540      393  0  14:33 tty1        00:00:00 -bash
rolan        1010     1009  0  18:43 tty1        00:00:00 -bash
rolan        1357     1356  0  19:37 tty1        00:00:00 -bash
rolan        1413     1357  0  20:11 tty1        00:00:00 vi loop
rolan        1746     1357  0  21:03 tty1        00:00:00 ps -f
[2]-  Убито          sh loop
rolan@debian:~$
```

Рисунок 11 – Запуск процесса и его уничтожение командой kill -9 PID.

12. Запустить еще один экземпляр оболочки: bash.

```
rolan@debian:~$ bash
rolan@debian:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
rolan         540       393  0  14:33 tty1        00:00:00 -bash
rolan        1010      1009  0  18:43 tty1        00:00:00 -bash
rolan        1357      1356  0  19:37 tty1        00:00:00 -bash
rolan        1413      1357  0  20:11 tty1        00:00:00 vi loop
rolan        1747      1357  0  21:04 tty1        00:00:00 bash
rolan        1749      1747  0  21:04 tty1        00:00:00 ps -f
rolan@debian:~$ _
```

Рисунок 12 – Запуск экземпляра оболочки bash.

13. Запустить несколько процессов в фоне. Останавливать их и снова запускать. Записать результаты просмотра командой ps -f.

```
rolan@debian:~$ sh loop&
[2] 1751
rolan@debian:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
rolan         540       393  0  14:33 tty1        00:00:00 -bash
rolan        1010      1009  0  18:43 tty1        00:00:00 -bash
rolan        1357      1356  0  19:37 tty1        00:00:00 -bash
rolan        1413      1357  0  20:11 tty1        00:00:00 vi loop
rolan        1747      1357  0  21:04 tty1        00:00:00 bash
rolan        1750      1747  50  21:07 tty1        00:03:34 sh loop
rolan        1751      1747  48  21:07 tty1        00:03:27 sh loop
rolan        1754      1747  0  21:14 tty1        00:00:00 ps -f
rolan@debian:~$ kill -19 1750
rolan@debian:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
rolan         540       393  0  14:33 tty1        00:00:00 -bash
rolan        1010      1009  0  18:43 tty1        00:00:00 -bash
rolan        1357      1356  0  19:37 tty1        00:00:00 -bash
rolan        1413      1357  0  20:11 tty1        00:00:00 vi loop
rolan        1747      1357  0  21:04 tty1        00:00:00 bash
rolan        1750      1747  49  21:07 tty1        00:03:39 sh loop
rolan        1751      1747  49  21:07 tty1        00:03:40 sh loop
rolan        1755      1747  0  21:14 tty1        00:00:00 ps -f

[1]+  Остановлен      sh loop
rolan@debian:~$ kill -18 1750
rolan@debian:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
rolan         540       393  0  14:33 tty1        00:00:00 -bash
rolan        1010      1009  0  18:43 tty1        00:00:00 -bash
rolan        1357      1356  0  19:37 tty1        00:00:00 -bash
rolan        1413      1357  0  20:11 tty1        00:00:00 vi loop
rolan        1747      1357  0  21:04 tty1        00:00:00 bash
rolan        1750      1747  45  21:07 tty1        00:03:41 sh loop
rolan        1751      1747  53  21:07 tty1        00:04:17 sh loop
rolan        1756      1747  0  21:15 tty1        00:00:00 ps -f
rolan@debian:~$ _
```

Рисунок 13 – Запуск нескольких процессов в фоне.

2. Часть II

Задание:

1. Запустить в консоли на выполнение три задачи, две в интерактивном режиме, одну - в фоновом.

Запустим задачи с помощью команд (рис. 13):

- sh loop.sh (интерактивный режим)
- sh loop.sh (интерактивный режим)
- sh loop.sh (фоновый режим)

```
rolan@debian:~$ sh loop
^X
^Z
[1]+  Остановлен    sh loop
rolan@debian:~$ sh loop
^Z
[2]+  Остановлен    sh loop
rolan@debian:~$ sh loop&
[3] 545
rolan@debian:~$ ps -f
UID      PID    PPID  C  STIME TTY          TIME CMD
rolan      532      384  0  21:19 tty1        00:00:00 -bash
rolan      543      532 35  21:21 tty1        00:00:21 sh loop
rolan      544      532  6  21:21 tty1        00:00:00 sh loop
rolan      545      532 99  21:22 tty1        00:00:07 sh loop
rolan      546      532  0  21:22 tty1        00:00:00 ps -f
rolan@debian:~$ _
```

Рисунок 13 – Запуск трех задач.

С помощью команды jobs получим список процессов в текущей оболочке (рис. 14):

- jobs (список процессов в текущей оболочке)

```
rolan@debian:~$ jobs -l
[1]-  543 Остановлено  sh loop
[2]+  544 Остановлено  sh loop
[3]    545 Запущен      sh loop &
rolan@debian:~$ bg 2
[2]+  sh loop &
```

Рисунок 14 – Использование команды jobs.

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

С помощью команды `bg` и номера, присвоенного задаче командной оболочкой при остановке ее исполнения, переведем задачу из интерактивного режима в фоновый (рис. 15):

- `bg %1` (возобновление задачи №1)

```
rolan@debian:~$ jobs
[1]+  Остановлен   sh loop
[2]   Запущен      sh loop &
[3]-  Запущен      sh loop &
rolan@debian:~$ fg 2
sh loop
^[[A^[[A^X
^X
^Z
[2]+  Остановлен   sh loop
rolan@debian:~$
rolan@debian:~$ fg 3
sh loop
^X^Z
[3]+  Остановлен   sh loop
rolan@debian:~$ jobs
[1]   Остановлен   sh loop
[2]-  Остановлен   sh loop
[3]+  Остановлен   sh loop
rolan@debian:~$ bg 1
[1] sh loop &
rolan@debian:~$ jobs
[1]   Запущен      sh loop &
[2]-  Остановлен   sh loop
[3]+  Остановлен   sh loop
rolan@debian:~$ fg 1
sh loop
^Z
[1]+  Остановлен   sh loop
rolan@debian:~$
```

Рисунок 15 – Перевод задачи в фоновый режим.

- `jobs` (список процессов в текущей оболочке)

3. 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.

Создадим именнованный канал для архивирования с помощью команды `mkfifo`. Посмотрим, что получилось в результате работы команды с помощью `ls -l`. Результат выполнения представлен на рис. 16.

- `mkfifo myBlog` (создание именованного канала с именем «myBlog»)
- `ls -l myBlog` (проверка создания файла)
- `gzip -9 -c < myBlog > out.gz` (передача дом. каталога)
- `zcat out.gz` (просмотр сжатых файлов)
- `tar -cvf out.tar /home > myBlog` (передача каталога с подкаталогами и файлами)
- `zcat out.gz` (просмотр сжатых файлов)

```
rolan@debian:~$ mkfifo myBlog
rolan@debian:~$ ls -l myBlog
prw-r--r-- 1 rolan rolan 0 ноя 28 21:51 myBlog
rolan@debian:~$
```

Рисунок 16 – создание именованного канала и проверка.

```
root@debian:/home/rolan# gzip -9 -c < myBlog > out.gz &
[1] 564
root@debian:/home/rolan# ls -R > myBlog
[1]+  Завершён      gzip -9 -c < myBlog > out.gz
root@debian:/home/rolan# zcat out.gz
.:
loop
loop2
myBlog
myCh
myChannel
out.gz
root@debian:/home/rolan# _
```

Рисунок 17 – Передача списка файлов домашнего каталога пользователя «rolan» и проверка.


```

root@debian:/home/rolan# mkdir MYDIR
root@debian:/home/rolan# cd MYDIR/
root@debian:/home/rolan/MYDIR# mkdir MYDIR1
root@debian:/home/rolan/MYDIR# mkdir MYDIR2
root@debian:/home/rolan/MYDIR# mkdir MYDIR3
root@debian:/home/rolan/MYDIR# touch MYFILE1
root@debian:/home/rolan/MYDIR# cd MYDIR2
root@debian:/home/rolan/MYDIR/MYDIR2# touch MYFILE2
root@debian:/home/rolan/MYDIR/MYDIR2# cd ..
root@debian:/home/rolan/MYDIR# touch MYFILE3
root@debian:/home/rolan/MYDIR# ls -l
итого 12
drwxr-xr-x 2 root root 4096 ноя 28 22:12 MYDIR1
drwxr-xr-x 2 root root 4096 ноя 28 22:13 MYDIR2
drwxr-xr-x 2 root root 4096 ноя 28 22:12 MYDIR3
-rw-r--r-- 1 root root    0 ноя 28 22:12 MYFILE1
-rw-r--r-- 1 root root    0 ноя 28 22:13 MYFILE3
root@debian:/home/rolan/MYDIR# cd ..
root@debian:/home/rolan# gzip -9 -c < myBlog > out.gz &
[1] 589
root@debian:/home/rolan# tar -cvf out.tar MYDIR > myBlog
[1]+  Завершён      gzip -9 -c < myBlog > out.gz

```

Рисунок 18 – Создание нового каталога с файлами и подкаталогами.

```

root@debian:/home/rolan# gzip -9 -c < myBlog > out.gz &
[1] 589
root@debian:/home/rolan# tar -cvf out.tar MYDIR > myBlog
[1]+  Завершён      gzip -9 -c < myBlog > out.gz

```

Рисунок 19 – Передача всего каталога с подкаталогами.

```

root@debian:/home/rolan# zcat out.gz
MYDIR/
MYDIR/MYDIR2/
MYDIR/MYDIR2/MYFILE2
MYDIR/MYDIR3/
MYDIR/MYDIR1/
MYDIR/MYFILE3
MYDIR/MYFILE1

```

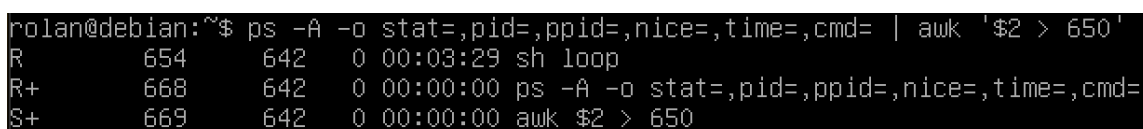
Рисунок 20 – Просмотр результатов.

3. Часть III

Вариант 3:

1. Сгенерировать следующую информацию о m ($m > 2$) процессах системы, имеющих значение идентификатора больше заданного n : флаг — сведения о процессе, статус, PID, PPID, приоритет, использованное время и имя программы.

```
ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk ' $2 > 650'
```

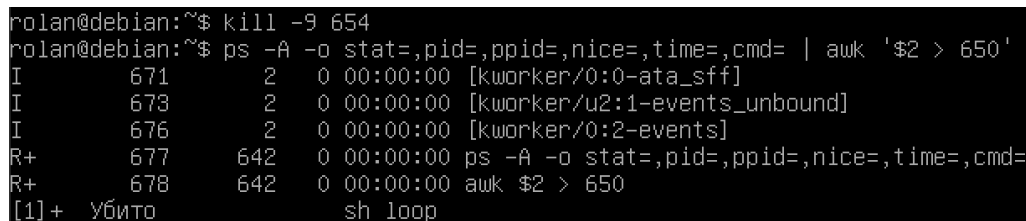


```
rolan@debian:~$ ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk '$2 > 650'
R      654      642    0 00:03:29 sh loop
R+     668      642    0 00:00:00 ps -A -o stat=,pid=,ppid=,nice=,time=,cmd=
S+     669      642    0 00:00:00 awk $2 > 650
```

Рисунок 21 – Генерация информации о процессах системы

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

```
kill -9 654
killall -2 ' sh loop'
```



```
rolan@debian:~$ kill -9 654
rolan@debian:~$ ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk '$2 > 650'
I      671        2    0 00:00:00 [kworker/0:0-ata_sff]
I      673        2    0 00:00:00 [kworker/u2:1-events_unbound]
I      676        2    0 00:00:00 [kworker/0:2-events]
R+     677      642    0 00:00:00 ps -A -o stat=,pid=,ppid=,nice=,time=,cmd=
R+     678      642    0 00:00:00 awk $2 > 650
[1]+  Убито                  sh loop
```

Рисунок 22 – Выполнения задания.

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

- `ps -ao pid,cmd,gid|grep -v 600`

С помощью опции команды `ps -a` выведем все процессы, а с помощью `-o` – отфильтруем вывод. С помощью команды `grep -v` отфильтруем строки так, чтобы в выводе были все строки за исключением, содержащих данный образец.

```
rolan@debian:~$ ps -ao pid,cmd,gid|grep -v 600
  PID CMD                      GID
   534 -bash                    1000
   577 ps -ao pid,cmd,gid       1000
rolan@debian:~$
```

Рисунок 23 – Выполнения задания.

Выводы

В ходе выполнения данной лабораторной работы мной были получены знания о понятии процесса, приобретен опыт и навыки управления процессами в операционной системе Linux.

Контрольные вопросы

1. Перечислите состояния задачи в ОС Ubuntu.

- running (выполнение) – после выделения ей процессора.
- sleeping (спячка) – при блокировке экрана.
- stopped (остановлена) – выполнение задачи прекращено, но из системы не удалена.
- dead (смерть) – может быть удалена из системы.
- active (активный) – используются при планировании выполнения процесса.
- expired (неактивный) – используются при планировании выполнения процесса.

2. Как создаются задачи в ОС Ubuntu?

Задачи создаются путем вызова функции clone.

3. Назовите классы потоков в ОС Ubuntu.

- Потоки реального времени, обслуживаемые по алгоритму FIFO.
- Потоки реального времени, обслуживаемые в порядке циклической очереди.
- Потоки разделения времени.

4. Как используется приоритет планирования при запуске задачи.

У каждого потока есть приоритет планирования. Значение по умолчанию равно 20, но оно может быть изменено при помощи системного вызова nice(value), вычитающего значение value из 20. Поскольку value должно находиться в диапазоне от -20 до +19, приоритеты всегда попадают в промежуток от 1 до 40.

5. Как можно изменить приоритет планирования для выполняющейся задачи?

с помощью команды nice.