

**Липецкий государственный технический университет**

**Факультет автоматизации и информатики**

**Кафедра автоматизированных систем управления**

**Лабораторная работа № 2**

**По дисциплине «OS Linux»**

**Процессы в операционной системе Linux**

Студент

Бахмутский М.В.

Группа АС-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

## Цель работы

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

## Ход работы

### 1. Часть 1

На рисунке 1 представлена загрузка обычным пользователем.

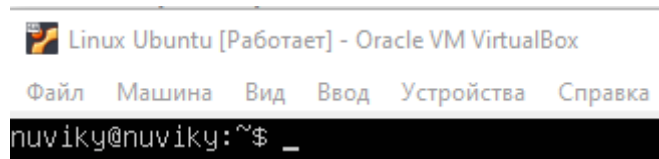


Рисунок 1 – Загрузка обычным пользователем

На рисунке 2 представлен файл с образом ядра

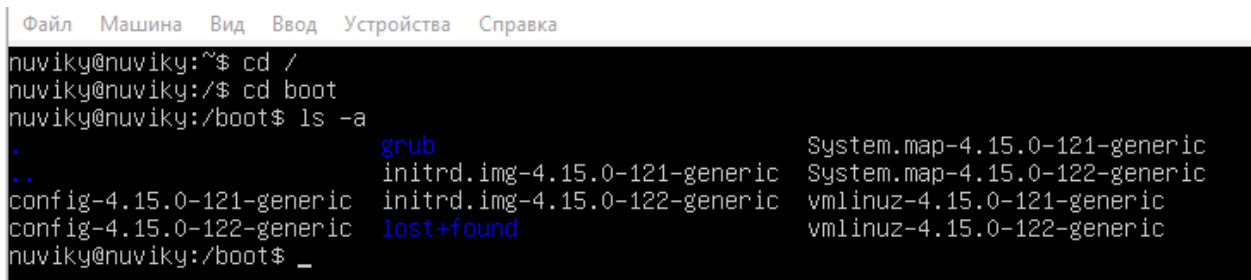


Рисунок 2 – Файл ядра Linux

Из рисунка 2 видно, что версия ядра 4.15.0-121.

На рисунке 3 показан вывод всех запущенных процессов в Linux в данной оболочке с помощью `ps -f`.

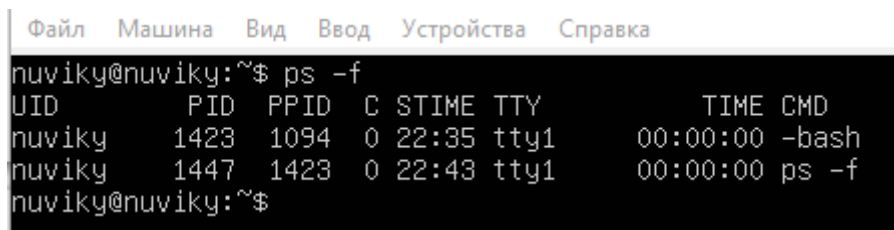


Рисунок 3 – Запущенные процессы в текущей оболочке

На рисунке 3 видно два запущенных процесса, с помощью файла справки проанализируем полученную информацию:

1. UID – пользователь, от имени которого запущен процесс;
2. PID – идентификатор процесса;
3. PPID – идентификатор родительского процесса;
4. C – процент времени CPU, используемого процессом;
5. STIME – время запуска процесса;
6. TTY – терминал, из которого запущен процесс;

7. TIME – общее время процессора, затраченное на выполнение процессора;

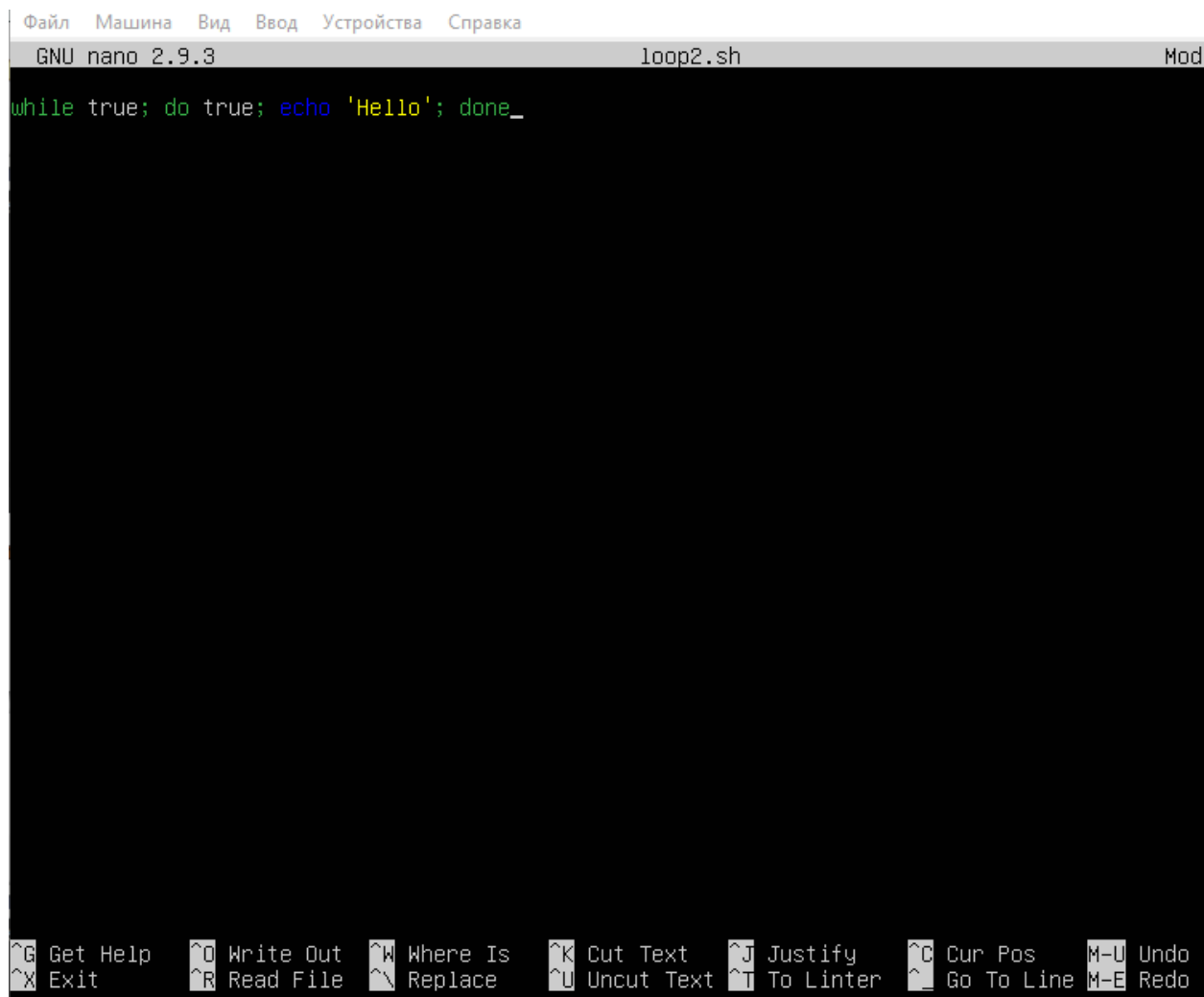
8. CMD – команда запуска процессора;

Создадим два сценария loop и loop2 с помощью текстового редактора nano, процесс создания показан на рисунках 4 и 5.



The image shows a terminal window with the nano text editor open. The title bar at the top indicates 'GNU nano 2.9.3' and the filename 'loop.sh'. The editor's main area is black with green text showing the command 'while true; do true; done' on the first line. The bottom status bar displays various keyboard shortcuts for editing and navigation, including '^G Get Help', '^O Write Out', '^W Where Is', '^K Cut Text', '^J Justify', '^C Cur Pos', 'M-U Undo', '^X Exit', '^R Read File', '^\_ Replace', '^U Uncut Text', '^T To Linter', '^\_ Go To Line', and 'M-E Redo'. A '[ New File ]' button is also visible above the status bar.

Рисунок 4 – Создание сценария loop



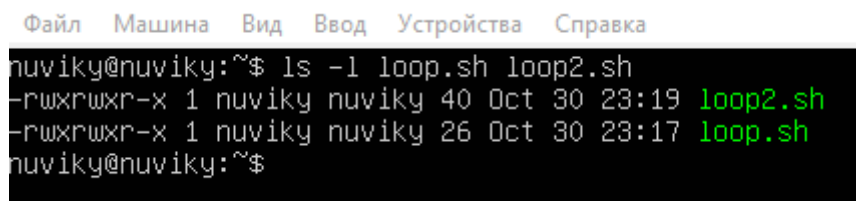
```
Файл  Машина  Вид  Ввод  Устройства  Справка
GNU nano 2.9.3                                loop2.sh                                Mod

while true; do true; echo 'Hello'; done_

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^_ Replace    ^U Uncut Text ^I To Linter  ^_ Go To Line M-E Redo
```

Рисунок 5 – Создание сценария loop2

Результат создания сценариев показан на рисунке 6.



```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviky@nuviky:~$ ls -l loop.sh loop2.sh
-rwxrwxr-x 1 nuviky nuviky 40 Oct 30 23:19 loop2.sh
-rwxrwxr-x 1 nuviky nuviky 26 Oct 30 23:17 loop.sh
nuviky@nuviky:~$
```

Рисунок 6 – Сценарии loop и loop2

Запустим процесс loop2 с помощью команды `sh loop2.sh`, результат показан на рисунке 7.



```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1493    1423  7  23:23 tty1        00:00:15 sh loop2.sh
nuviku       1510    1423  0  23:26 tty1        00:00:00 ps -f
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1493    1423  6  23:23 tty1        00:00:15 sh loop2.sh
nuviku       1511    1423  0  23:26 tty1        00:00:00 ps -f
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1493    1423  6  23:23 tty1        00:00:15 sh loop2.sh
nuviku       1512    1423  0  23:27 tty1        00:00:00 ps -f
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1493    1423  5  23:23 tty1        00:00:15 sh loop2.sh
nuviku       1513    1423  0  23:27 tty1        00:00:00 ps -f
nuviku@nuviku:~$ _
```

Рисунок 9 – Последовательный запуск команды ps -f

На рисунке 9 видно как у процесса loop2 меняется значения параметра C (процент времени CPU, используемого процессом). Процесс остается в выводе ps -f и у него постепенно уменьшается параметр C, потому что мы останавливали процесс а не завершили.

Завершим процесс loop2 с помощью команды kill -9 1493. (рисунок 10)

```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1493    1423  4  23:23 tty1        00:00:15 sh loop2.sh
nuviku       1515    1423  0  23:29 tty1        00:00:00 ps -f
nuviku@nuviku:~$ kill -9 1493
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1516    1423  0  23:29 tty1        00:00:00 ps -f
[1]+  Killed                  sh loop2.sh
nuviku@nuviku:~$
```

Рисунок 10 – Завершение процесса loop2

Из рисунка 10 видно, что процесс был завершен. С помощью сигнала KILL мы завершаем процесс без проверки на корректное завершение.

Запустим процесс loop с помощью команды sh loop.sh&. (рисунок 11)

```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviky@nuviky:~$ sh loop.sh&
[1] 1557
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C   STIME TTY          TIME CMD
nuviky       1423    1094  0   22:35 tty1        00:00:00 -bash
nuviky       1557    1423  90   23:34 tty1        00:00:04 sh loop.sh
nuviky       1558    1423  0   23:34 tty1        00:00:00 ps -f
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C   STIME TTY          TIME CMD
nuviky       1423    1094  0   22:35 tty1        00:00:00 -bash
nuviky       1557    1423  97   23:34 tty1        00:00:07 sh loop.sh
nuviky       1559    1423  0   23:34 tty1        00:00:00 ps -f
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C   STIME TTY          TIME CMD
nuviky       1423    1094  0   22:35 tty1        00:00:00 -bash
nuviky       1557    1423  97   23:34 tty1        00:00:10 sh loop.sh
nuviky       1560    1423  0   23:34 tty1        00:00:00 ps -f
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C   STIME TTY          TIME CMD
nuviky       1423    1094  0   22:35 tty1        00:00:00 -bash
nuviky       1557    1423  96   23:34 tty1        00:00:14 sh loop.sh
nuviky       1561    1423  0   23:34 tty1        00:00:00 ps -f
nuviky@nuviky:~$ _
```

Рисунок 11 – Запуск процесса loop

На рисунке 11 видно, что процесс запущен и занимает большой процент процессорного времени.

Завершим процесс с помощью команды kill -15 1557. (рисунок 12)

```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C   STIME TTY          TIME CMD
nuviky       1423    1094  0   22:35 tty1        00:00:00 -bash
nuviky       1557    1423  99   23:34 tty1        00:02:48 sh loop.sh
nuviky       1563    1423  0   23:36 tty1        00:00:00 ps -f
nuviky@nuviky:~$ kill -15 1557
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C   STIME TTY          TIME CMD
nuviky       1423    1094  0   22:35 tty1        00:00:00 -bash
nuviky       1564    1423  0   23:37 tty1        00:00:00 ps -f
[1]+  Terminated                  sh loop.sh
nuviky@nuviky:~$
```

Рисунок 12 – Завершения процесса loop с помощью сигнала TERM

На рисунке 12 видно, что процесс завершен после корректного завершения выполнения процесса.

Создадим процесс loop в фоне и закроем его с помощью команды kill -9 1569. (рисунок 13)



```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ sh loop.sh&
[1] 1569
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1569    1423  91  23:39 tty1        00:00:03 sh loop.sh
nuviku       1570    1423  0  23:39 tty1        00:00:00 ps -f
nuviku@nuviku:~$ kill -9 1569
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1572    1423  0  23:39 tty1        00:00:00 ps -f
[1]+  Killed                  sh loop.sh
nuviku@nuviku:~$ _
```

Рисунок 13 – Завершения процесса loop с помощью сигнала KILL

Из рисунка 13 видно, что процесс loop был закрыт, не дожидаясь корректного завершения процесса.

Запустим еще один экземпляр bash с помощью команды bash. (рисунок 14)

```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1574    1423  0  23:41 tty1        00:00:00 ps -f
nuviku@nuviku:~$ bash
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1575    1423  0  23:41 tty1        00:00:00 bash
nuviku       1583    1575  0  23:41 tty1        00:00:00 ps -f
nuviku@nuviku:~$
```

Рисунок 14 – Запуск еще одного процесса bash

Запуск нескольких процессов loop показан на рисунке 15.

```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ sh loop.sh&
[1] 1589
nuviku@nuviku:~$ sh loop.sh&
[2] 1590
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1423    1094  0  22:35 tty1        00:00:00 -bash
nuviku       1575    1423  0  23:41 tty1        00:00:00 bash
nuviku       1589    1575  99  23:44 tty1        00:00:42 sh loop.sh
nuviku       1590    1575  99  23:45 tty1        00:00:17 sh loop.sh
nuviku       1591    1575  0  23:45 tty1        00:00:00 ps -f
nuviku@nuviku:~$
```

Рисунок 15 – Запуск нескольких процессов в фоновом режиме

Остановка процессов loop с помощью команды kill -19 1462 1463 показана на рисунке 16.

```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1438    1121   0  23:47 tty1        00:00:00 -bash
nuviku       1452    1438   0  23:47 tty1        00:00:00 bash
nuviku       1462    1452  99  23:48 tty1        00:00:21 sh loop.sh
nuviku       1463    1452  99  23:48 tty1        00:00:16 sh loop.sh
nuviku       1466    1452   0  23:48 tty1        00:00:00 ps -f
nuviku@nuviku:~$ kill -19 1462 1463

[1]-  Stopped                  sh loop.sh

[2]+  Stopped                  sh loop.sh
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1438    1121   0  23:47 tty1        00:00:00 -bash
nuviku       1452    1438   0  23:47 tty1        00:00:00 bash
nuviku       1462    1452  86  23:48 tty1        00:00:41 sh loop.sh
nuviku       1463    1452  84  23:48 tty1        00:00:36 sh loop.sh
nuviku       1467    1452   0  23:49 tty1        00:00:00 ps -f
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1438    1121   0  23:47 tty1        00:00:00 -bash
nuviku       1452    1438   0  23:47 tty1        00:00:00 bash
nuviku       1462    1452  63  23:48 tty1        00:00:41 sh loop.sh
nuviku       1463    1452  60  23:48 tty1        00:00:36 sh loop.sh
nuviku       1468    1452   0  23:49 tty1        00:00:00 ps -f
nuviku@nuviku:~$
```

Рисунок 16 – Остановка процессов

Из рисунка 16 видно, что процессы приостанавливаются т.к. уменьшается процент затрачиваемого процессорного времени.

Возобновим процессы с помощью команды kill -18 1462 1463. (рисунок 17)

```

Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ kill -18 1462 1463
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1438    1121   0  23:47 tty1        00:00:00 -bash
nuviku       1452    1438   0  23:47 tty1        00:00:00 bash
nuviku       1462    1452  22  23:48 tty1        00:00:47 sh loop.sh
nuviku       1463    1452  20  23:48 tty1        00:00:42 sh loop.sh
nuviku       1470    1452   0  23:51 tty1        00:00:00 ps -f
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1438    1121   0  23:47 tty1        00:00:00 -bash
nuviku       1452    1438   0  23:47 tty1        00:00:00 bash
nuviku       1462    1452  23  23:48 tty1        00:00:50 sh loop.sh
nuviku       1463    1452  21  23:48 tty1        00:00:45 sh loop.sh
nuviku       1471    1452   0  23:51 tty1        00:00:00 ps -f
nuviku@nuviku:~$
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1438    1121   0  23:47 tty1        00:00:00 -bash
nuviku       1452    1438   0  23:47 tty1        00:00:00 bash
nuviku       1462    1452  27  23:48 tty1        00:01:01 sh loop.sh
nuviku       1463    1452  25  23:48 tty1        00:00:56 sh loop.sh
nuviku       1472    1452   0  23:51 tty1        00:00:00 ps -f
nuviku@nuviku:~$ _

```

Рисунок 17 – Возобновление процессов

Из рисунка 17 видно, что процессы возобновляются, о чем свидетельствуем увеличение процента затрачиваемого процессорного времени.

## 2. Часть 2

Запустим два процесса loop.sh в интерактивном режиме и один в фоновом. (рисунок 18)

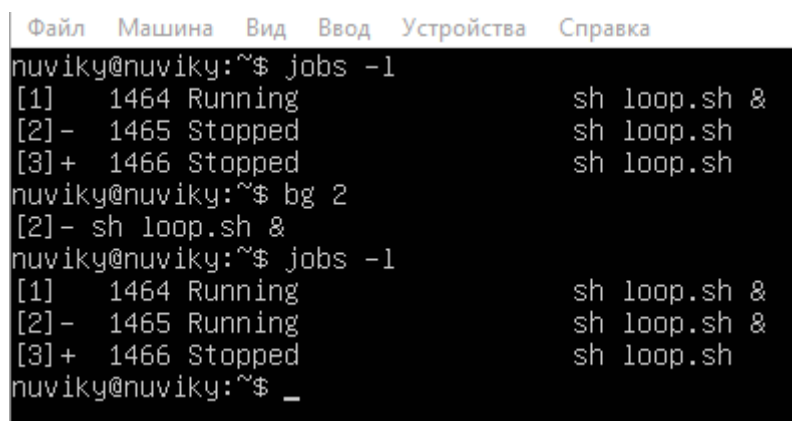
```

Файл  Машина  Вид  Ввод  Устройства  Справка
nuviku@nuviku:~$ sh loop.sh&
[1] 1464
nuviku@nuviku:~$ sh loop.sh; sh loop.sh
^Z
[2]+  Stopped                  sh loop.sh
^Z
[3]+  Stopped                  sh loop.sh
nuviku@nuviku:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
nuviku       1449    1139   0  00:01 tty1        00:00:00 -bash
nuviku       1464    1449  99  00:02 tty1        00:00:33 sh loop.sh
nuviku       1465    1449  13  00:02 tty1        00:00:02 sh loop.sh
nuviku       1466    1449  48  00:02 tty1        00:00:07 sh loop.sh
nuviku       1467    1449   0  00:02 tty1        00:00:00 ps -f
nuviku@nuviku:~$

```

Рисунок 18 – Запуск процессов

Переведем один из процессов в фоновый режим с помощью команды `bg` 2, результат показан на рисунке 19.



```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviky@nuviky:~$ jobs -l
[1]  1464 Running                sh loop.sh &
[2]- 1465 Stopped                sh loop.sh
[3]+ 1466 Stopped                sh loop.sh
nuviky@nuviky:~$ bg 2
[2]- sh loop.sh &
nuviky@nuviky:~$ jobs -l
[1]  1464 Running                sh loop.sh &
[2]- 1465 Running                sh loop.sh &
[3]+ 1466 Stopped                sh loop.sh
nuviky@nuviky:~$ _
```

Рисунок 19 – Перевод процесса в фоновый режим

Переведем процесс 1 в интерактивный режим с помощью команды `fg %1`, затем переведем процесс 3 в фоновый режим, результат показан на рисунке 20.

```
Файл  Машина  Вид  Ввод  Устройства  Справка
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
nuviky       1449    1139   0 00:01 tty1        00:00:00 -bash
nuviky       1464    1449   99 00:02 tty1        00:10:19 sh loop.sh
nuviky       1465    1449   49 00:02 tty1        00:04:57 sh loop.sh
nuviky       1466    1449   1 00:02 tty1        00:00:07 sh loop.sh
nuviky       1473    1449   0 00:12 tty1        00:00:00 ps -f
nuviky@nuviky:~$ jobs -l
[1] 1464 Running                  sh loop.sh &
[2]- 1465 Running                  sh loop.sh &
[3]+ 1466 Stopped                  sh loop.sh
nuviky@nuviky:~$ fg %1
sh loop.sh
^Z
[1]+  Stopped                      sh loop.sh
nuviky@nuviky:~$ jobs -l
[1]+ 1464 Stopped                  sh loop.sh
[2] 1465 Running                  sh loop.sh &
[3]- 1466 Stopped                  sh loop.sh
nuviky@nuviky:~$ bg %3
[3]- sh loop.sh &
nuviky@nuviky:~$ jobs -l
[1]+ 1464 Stopped                  sh loop.sh
[2] 1465 Running                  sh loop.sh &
[3]- 1466 Running                  sh loop.sh &
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
nuviky       1449    1139   0 00:01 tty1        00:00:00 -bash
nuviky       1464    1449   93 00:02 tty1        00:11:25 sh loop.sh
nuviky       1465    1449   57 00:02 tty1        00:06:48 sh loop.sh
nuviky       1466    1449   3 00:02 tty1        00:00:24 sh loop.sh
nuviky       1475    1449   0 00:14 tty1        00:00:00 ps -f
nuviky@nuviky:~$
```

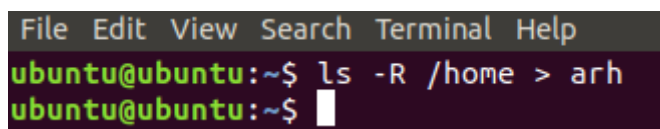
Рисунок 20 – Исследование переводов режимов работы процесса

Создадим именованный канал с помощью команды `mkfifo arh`, после организуем связь между выводами и вводами различных процессов. С помощью данного канала сделаем так, чтобы утилита для архивирования `gzip` получала ввод из канала и выводила результат в файл `test`, результат показан на рисунке 21.

```
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ mkfifo arh
ubuntu@ubuntu:~$ gzip -9 -c < arh > test
```

Рисунок 21- Создание именованного канала для архивирования

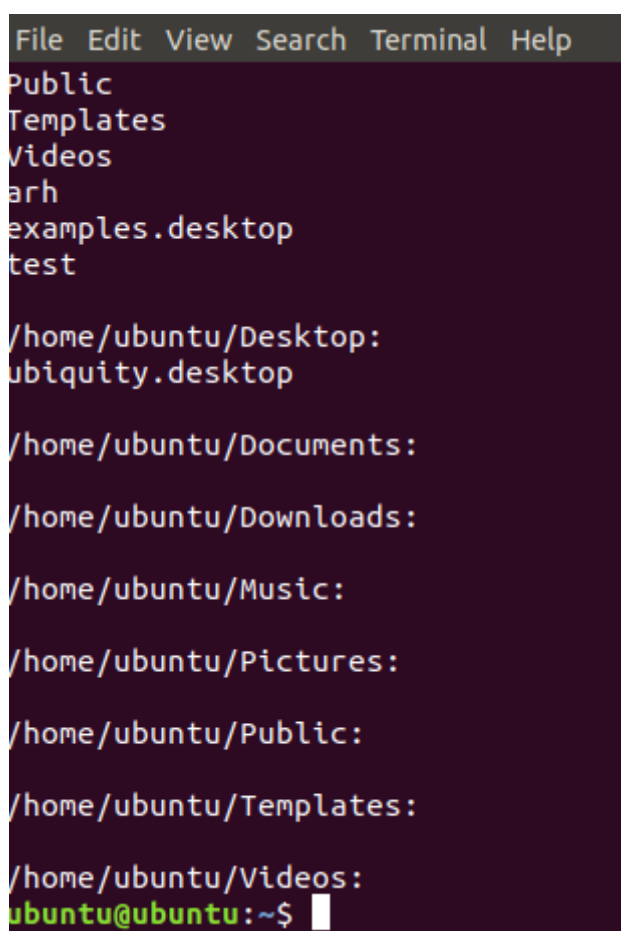
Для передачи в канал списка содержимого каталога и его подкаталогов откроем новый терминал и напишем команду `ls -R /home > arh`, результат показан на рисунке 22.



```
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ ls -R /home > arh
ubuntu@ubuntu:~$
```

Рисунок 22 – Передача в канал для архивирования список содержимого домашнего каталога

Для просмотра содержимого файла `test` откроем первый терминал и введем команду `zcat test`, результат показан на рисунке 23.



```
File Edit View Search Terminal Help
Public
Templates
Videos
arh
examples.desktop
test

/home/ubuntu/Desktop:
ubiquity.desktop

/home/ubuntu/Documents:

/home/ubuntu/Downloads:

/home/ubuntu/Music:

/home/ubuntu/Pictures:

/home/ubuntu/Public:

/home/ubuntu/Templates:

/home/ubuntu/Videos:
ubuntu@ubuntu:~$
```

Рисунок 23 – Содержание файла `test`

Подготовим тестовый каталог, поместив в него файлы и подкаталог. Каталоги создаем с помощью команды `mkdir`, а файлы с помощью `touch`, процесс создания показан на рисунке 24.

```
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ mkdir test_kat
ubuntu@ubuntu:~$ cd test_kat
ubuntu@ubuntu:~/test_kat$ mkdir test_subkat
ubuntu@ubuntu:~/test_kat$ touch test1
ubuntu@ubuntu:~/test_kat$ touch test2
ubuntu@ubuntu:~/test_kat$ touch test3
ubuntu@ubuntu:~/test_kat$ cd test_subkat
ubuntu@ubuntu:~/test_kat/test_subkat$ touch test 4
ubuntu@ubuntu:~/test_kat/test_subkat$ touch test 5
ubuntu@ubuntu:~/test_kat/test_subkat$ touch test 6
ubuntu@ubuntu:~/test_kat/test_subkat$
```

Рисунок 24 – Создание тестового каталога

Создадим новое соединение и передадим тестовый каталог в него с помощью команды `gzip -9 -c < arh > test2`, создание показано на рисунке 25.

```
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ gzip -9 -c < arh > test2

```

Рисунок 25 – Создание нового соединения

Для передачи списка содержимого тестового каталога откроем второй терминал и напишем команду `ls test_kat > arh`.

```
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ ls test_kat > arh
ubuntu@ubuntu:~$
```

Рисунок 26 – Передача списка содержимого тестового каталога

Для просмотра содержимого файла `test2` перейдем в первый каталог и используем команду `zcat test2`, результат показан на рисунке 27.

```
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ zcat test2
test1
test2
test3
test_subkat
ubuntu@ubuntu:~$
```

Рисунок 27 – Содержимое файла `test2`

### 3 Часть 3 (вариант 1)

Для просмотра информации о всех процессах системы воспользуемся командой `ps -eF`, результат показан на рисунке 28.

Файл	Машина	Вид	Ввод	Устройства	Справка				
root	423	2	0	0	0	1	02:34	?	00:00:00 [ib-comp-unb-wq]
root	424	2	0	0	0	5	02:34	?	00:00:00 [ib_mcast]
root	425	2	0	0	0	3	02:34	?	00:00:00 [ib_n1_sa_wq]
root	429	2	0	0	0	5	02:34	?	00:00:00 [kworker/5:2]
root	430	1	0	26476	1932	3	02:34	?	00:00:00 /sbin/lvmtool -f
root	433	2	0	0	0	5	02:34	?	00:00:00 [rdma_cm]
root	434	1	0	11674	5564	4	02:34	?	00:00:01 /lib/systemd/systemd-udev
root	456	2	0	0	0	1	02:34	?	00:00:00 [kworker/1:2]
root	461	2	0	0	0	4	02:34	?	00:00:00 [kworker/4:2]
root	463	2	0	0	0	2	02:34	?	00:00:00 [iprt-VBoxQueue]
root	487	2	0	0	0	2	02:34	?	00:00:00 [kworker/2:1H]
root	587	2	0	0	0	2	02:34	?	00:00:00 [jbd2/sda2-8]
root	588	2	0	0	0	1	02:34	?	00:00:00 [ext4-rsv-conver]
root	613	2	0	0	0	3	02:34	?	00:00:00 [kworker/3:2]
systemd+	630	1	0	35489	3400	0	02:34	?	00:00:00 /lib/systemd/systemd-timesyncd
systemd+	794	1	0	20020	5320	2	02:34	?	00:00:00 /lib/systemd/systemd-networkd
systemd+	810	1	0	17665	5512	5	02:34	?	00:00:00 /lib/systemd/systemd-resolved
root	904	1	0	27604	2044	0	02:34	?	00:00:00 /usr/sbin/iqbalance --foreground
daemon	909	1	0	7083	2500	3	02:34	?	00:00:00 /usr/sbin/atd -f
root	919	1	0	7507	3308	2	02:34	?	00:00:00 /usr/sbin/cron -f
root	930	1	0	71564	6920	0	02:34	?	00:00:00 /usr/lib/accounts-service/accounts-
root	933	1	0	23885	1488	3	02:34	?	00:00:00 /usr/bin/lxcfs /var/lib/lxcfs/
root	934	1	0	17653	6016	4	02:34	?	00:00:00 /lib/systemd/systemd-logind
syslog	936	1	0	65759	4452	3	02:34	?	00:00:00 /usr/sbin/rsyslogd -n
root	997	1	0	42276	17052	0	02:34	?	00:00:00 /usr/bin/python3 /usr/bin/networkd
message+	1004	1	0	12512	4716	5	02:34	?	00:00:00 /usr/bin/dbus-daemon --system --ad
root	1072	1	0	46486	20056	2	02:34	?	00:00:00 /usr/bin/python3 /usr/share/unatte
root	1104	1	0	19691	3708	5	02:34	tty1	00:00:00 /bin/login -p --
root	1105	1	0	72221	6624	5	02:34	?	00:00:00 /usr/lib/policykit-1/polkitd --no-
nuviky	1441	1	0	19173	7188	1	02:34	?	00:00:00 /lib/systemd/systemd --user
nuviky	1450	1441	0	27959	2448	0	02:34	?	00:00:00 (sd-pam)
nuviky	1468	1104	0	5433	5344	0	02:34	tty1	00:00:00 -bash
nuviky	1507	1468	99	1157	852	1	02:37	tty1	00:00:51 sh loop.sh
nuviky	1508	1468	99	1157	780	3	02:37	tty1	00:00:50 sh loop.sh
nuviky	1509	1468	99	1157	864	2	02:38	tty1	00:00:33 sh loop.sh
nuviky	1511	1468	0	9593	3592	4	02:38	tty1	00:00:00 ps -ef

Рисунок 28 – Информация о всех процессах системы

Для завершения процесса с помощью сигнала SIGTERM воспользуемся командой kill -15 1508, а для завершения процесса с помощью сигнала SIGKILL воспользуемся командой kill -9 1507, результат показан на рисунке 29.



```

nuviky@nuviky:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
nuviky       1468    1104   0 02:34 tty1        00:00:00 -bash
nuviky       1507    1468   99 02:37 tty1        00:01:20 sh loop.sh
nuviky       1508    1468   99 02:37 tty1        00:01:18 sh loop.sh
nuviky       1509    1468   98 02:38 tty1        00:01:02 sh loop.sh
nuviky       1513    1468   0 02:39 tty1        00:00:00 ps -f
nuviky@nuviky:~$ kill -9 1507
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
nuviky       1468    1104   0 02:34 tty1        00:00:00 -bash
nuviky       1508    1468   99 02:37 tty1        00:02:05 sh loop.sh
nuviky       1509    1468   99 02:38 tty1        00:01:49 sh loop.sh
nuviky       1514    1468   0 02:39 tty1        00:00:00 ps -f
[1] Killed sh loop.sh
nuviky@nuviky:~$ kill -15 1508
nuviky@nuviky:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
nuviky       1468    1104   0 02:34 tty1        00:00:00 -bash
nuviky       1509    1468   99 02:38 tty1        00:02:10 sh loop.sh
nuviky       1517    1468   0 02:40 tty1        00:00:00 ps -f
[2]- Terminated sh loop.sh
nuviky@nuviky:~$ _

```

Рисунок 29 – Завершение процессов с помощью сигнала SIGTERM и SIG-KILL

Для определения идентификаторов процессов, владельцем которых не является root, можно узнать с помощью команды `ps -fu nuviky`, результат показан на рисунке 30.

```

nuviky@nuviky:~$ ps -fu nuviky
UID          PID    PPID  C STIME TTY          TIME CMD
nuviky       1441      1   0 02:34 ?          00:00:00 /lib/systemd/systemd --user
nuviky       1450    1441   0 02:34 ?          00:00:00 (sd-pam)
nuviky       1468    1104   0 02:34 tty1        00:00:00 -bash
nuviky       1509    1468   99 02:38 tty1        00:03:09 sh loop.sh
nuviky       1519    1468   0 02:41 tty1        00:00:00 ps -fu nuviky
nuviky@nuviky:~$

```

Рисунок 30 – Процессы, владельцы которых является пользователь nuviky

## 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.