

**Липецкий государственный технический университет**

**Факультет автоматизации и информатики**

**Кафедра автоматизированных систем управления**

**ЛАБОРАТНАЯ РАБОТА №2**

**по дисциплине «OS Linux»**

**на тему «Процессы в операционной системе Linux»**

Студент

Сухоруких А.О.

Группа АС-18

Руководитель

Кургасов В.В.

к.т.н.

Липецк 2020 г.

## Оглавление

Цель работы.....	4
Ход работы .....	5
1. Задание 1 .....	5
1.1 Загрузиться не root, а пользователем .....	5
1.2 Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.....	6
1.3 Посмотреть процессы ps -f.....	7
1.4 Написать с помощью редактора vi два сценария loop и loop2.....	8
1.5 Запустить loop2 на переднем плане.....	9
1.6 Остановить, пошлав сигнал STOP .....	9
1.7 Посмотреть последовательно несколько раз ps -f.....	10
1.8 Убить процесс loop2, пошлав сигнал kill -9 PID .....	11
1.9 Запустить в фоне процесс loop. Не останавливая, осмотреть несколько раз: ps -f.....	12
1.10 Завершить процесс loop командой kill -15 PID.....	13
1.11 Третий раз запустить в фоне. Не останавливая убить командой kill -9 PID .....	13
1.12 Запустить несколько процессов в фоне. Останавливать их и снова запускать. Записать результаты просмотра командой ps -f .....	14
2. Задание 2.....	15
2.1 Запустить в консоли на выполнение три задачи, две в интерактивном режиме, одну - в фоновом.....	16
2.2 Перевести одну из задач, выполняющихся в интерактивном режиме, в фоновый режим .....	16
2.3 Провести эксперименты по переводу задач из фонового режима в интерактивный и наоборот .....	17
2.4 Создать именованный канал для архивирования и осуществить передачу в канал .....	18
3. Задание 3 .....	22

3.1 Отобразить информацию о трех заданных процессах в реальном режиме, одному из процессов переназначить приоритет, не выходя из команды.....	22
3.2 Послать сигнал на безусловное завершение (по имени и по номеру сигнала) процессу по его имени, установить подтверждение завершения.....	23
3.3 Выведите статистику работы системы с момента последней загрузки. ...	23
Вывод .....	26

## Цель работы

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

## Ход работы

### 1. Задание 1

- 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`.

#### 1.1 Загрузиться не root, а пользователем

После запуска ОС, система предложит нам произвести вход под определенным пользователем. Вводим логин пользователя и пароль. Результат выполнения представлен на рисунке 1.

```
Ubuntu 20.04.1 LTS artemserver tty1
artemserver login: [ 68.431727] cloud-init[812]: Cloud-init v. 20.3-2-g371b392c-0ubuntu1~20.04.1 r
unning 'modules:config' at Thu, 29 Oct 2020 16:48:06 +0000. Up 67.88 seconds.
[ 69.533214] cloud-init[819]: Cloud-init v. 20.3-2-g371b392c-0ubuntu1~20.04.1 running 'modules:fin
al' at Thu, 29 Oct 2020 16:48:07 +0000. Up 69.15 seconds.
[ 69.533455] cloud-init[819]: Cloud-init v. 20.3-2-g371b392c-0ubuntu1~20.04.1 finished at Thu, 29
Oct 2020 16:48:08 +0000. DataSource DataSourceNone. Up 69.51 seconds
[ 69.533603] cloud-init[819]: 2020-10-29 16:48:08,215 - cc_final_message.py[WARNING]: Used fallback
k datasource
artem
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

 * Introducing self-healing high availability clustering for MicroK8s!
   Super simple, hardened and opinionated Kubernetes for production.

   https://microk8s.io/high-availability

39 updates can be installed immediately.
9 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Wed Oct 28 14:36:42 UTC 2020 on tty1
artem@artemserver:~$
```

Рисунок 1 – Вход под определенным пользователем

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

Для этого перейдем в корневой каталог, затем в папку boot. Там находим файл vmlinuz-5.4.0. Из названия файла видно, что версия ядра – 5.4.0. Результат поиска номера версии представлен на рисунке 2.

```
artem@artemserver:~$ cd /
artem@artemserver:/$ ls -l
total 68
drwxr-xr-x 1 root root 7 Jul 31 16:28 bin -> usr/bin
drwxr-xr-x 4 root root 4096 Oct 28 14:36 boot
drwxr-xr-x 2 root root 4096 Oct 3 06:29 cdrom
drwxr-xr-x 19 root root 4080 Oct 29 16:47 dev
drwxr-xr-x 94 root root 4096 Oct 28 14:35 etc
drwxr-xr-x 4 root root 4096 Oct 16 16:55 home
drwxr-xr-x 1 root root 7 Jul 31 16:28 lib -> usr/lib
drwxr-xr-x 1 root root 9 Jul 31 16:28 lib32 -> usr/lib32
drwxr-xr-x 1 root root 9 Jul 31 16:28 lib64 -> usr/lib64
drwxr-xr-x 1 root root 10 Jul 31 16:28 libx32 -> usr/libx32
drwx----- 2 root root 16384 Oct 3 06:28 lost+found
drwxr-xr-x 2 root root 4096 Jul 31 16:28 media
drwxr-xr-x 2 root root 4096 Jul 31 16:28 mnt
drwxr-xr-x 2 root root 4096 Jul 31 16:28 opt
dr-xr-xr-x 104 root root 0 Oct 29 16:47 proc
drwx----- 4 root root 4096 Oct 17 07:30 root
drwxr-xr-x 25 root root 760 Oct 29 16:48 run
drwxr-xr-x 1 root root 8 Jul 31 16:28/sbin -> usr/sbin
drwxr-xr-x 6 root root 4096 Oct 3 06:40 snap
drwxr-xr-x 2 root root 4096 Jul 31 16:28 srv
dr-xr-xr-x 13 root root 0 Oct 29 16:47 sys
drwxr-xrwt 11 root root 4096 Oct 29 16:48 tmp
drwxr-xr-x 14 root root 4096 Jul 31 16:29 usr
drwxr-xr-x 13 root root 4096 Jul 31 16:30 var
artem@artemserver:/$ cd boot
artem@artemserver:/boot$ ls -la
.
..
config-5.4.0-52-generic vmlinuz
grub vmlinuz-5.4.0-48-generic
System.map-5.4.0-48-generic initrd.img vmlinuz-5.4.0-51-generic
System.map-5.4.0-51-generic initrd.img-5.4.0-48-generic vmlinuz-5.4.0-52-generic
System.map-5.4.0-52-generic initrd.img-5.4.0-51-generic vmlinuz.old
config-5.4.0-48-generic initrd.img.old
config-5.4.0-51-generic lost+found
artem@artemserver:/boot$
```

Рисунок 2 – Поиск версии по имени файла

### 1.3 Посмотреть процессы ps -f

Для выполнения данного пункта выполним команду ps -f. Результат команды представлен на рисунке 3.

```
pidlist.  
q pidlist  
    Select by process ID (quick mode).  Identical to -q and --quick-pid.  
  
-q pidlist  
    Select by PID (quick mode).  This selects the processes whose process ID numbers  
    appear in pidlist.  With this option ps reads the necessary info only for the pids  
    listed in the pidlist and doesn't apply additional filtering rules.  The order of  
    pids is unsorted and preserved.  No additional selection options, sorting and  
    forest type listings are allowed in this mode.  Identical to q and --quick-pid.  
  
--quick-pid pidlist  
    Select by process ID (quick mode).  Identical to -q and q.  
  
-s sesslist  
    Select by session ID.  This selects the processes with a session ID specified in  
    sesslist.  
  
--sid sesslist  
    Select by session ID.  Identical to -s.  
  
t ttylist  
    Select by tty.  Nearly identical to -t and --tty, but can also be used with an  
    empty ttylist to indicate the terminal associated with ps.  Using the T option is  
    considered cleaner than using t with an empty ttylist.  
  
-t ttylist  
    Select by tty.  This selects the processes associated with the terminals given in  
    ttylist.  Terminals (ttys, or screens for text output) can be specified in several  
    forms: /dev/ttyS1, ttyS1, S1.  A plain "-" may be used to select processes not  
    attached to any terminal.  
artem@artemserver:/boot$ ps -f  
UID      PID     PPID  C  STIME TTY          TIME CMD  
artem    951      675  0  16:48 tty1        00:00:00 -bash  
artem    1043     951  1  17:02 tty1        00:00:00 ps -f  
artem@artemserver:/boot$
```

Рисунок 3 – Результат выполнения команды ps -f

При выполнении команды ps -f будет выведена следующая информация о процессах:

UID - пользователь, от имени которого запущен процесс;

PID - идентификатор процесса;

PPID - идентификатор родительского процесса;

C - процент времени CPU, используемого процессом;

STIME - время запуска процесса;

TTY - терминал, из которого запущен процесс;

TIME - общее время процессора, затраченное на выполнение процессора;

CMD - команда запуска процессора;







```
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  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
^Z  
[1]+ Stopped sh loop2  
artem@artemserver:~$
```

### Рисунок 7 – Сигнал STOP

### 1.7 Посмотреть последовательно несколько раз ps -f

Выполнив команды `ps -f` 5 раз мы видим, что `UID`, `PID`, `PPID`, `STIME`, `TIME` и `CMD` не поменялись, а параметр `C` каждый раз менялся. Мы видим, что процент времени `CPU` используемого процессором уменьшается. Результат выполнения представлен на рисунке 8.

```

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
^Z
[1]+  Stopped                  sh loop2
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1086      951  55  17:20 tty1        00:01:13 sh loop2
artem        1088      951  0  17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1086      951  52  17:20 tty1        00:01:13 sh loop2
artem        1089      951  0  17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1086      951  49  17:20 tty1        00:01:13 sh loop2
artem        1090      951  0  17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1086      951  47  17:20 tty1        00:01:13 sh loop2
artem        1091      951  0  17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1086      951  46  17:20 tty1        00:01:13 sh loop2
artem        1092      951  0  17:22 tty1        00:00:00 ps -f
artem@artemserver:~$

```

Рисунок 8 – Выполнение команды ps -f 5 раз

### 1.8 Убить процесс loop2, послав сигнал kill -9 PID

Команда kill -9 1086 выполняет уничтожение процесса с PID – 1086, что соответствует loop2. Сигнал SIGKILL же выполняет уничтожение процесса всегда, так как его нельзя перехватить или проигнорировать. Результат команды kill -9 1086 представлен на рисунке 9.

```

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
^Z
[1]+  Stopped                  sh loop2
artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem         951      675  0 16:49 tty1        00:00:00 -bash
artem        1086      951 55 17:20 tty1        00:01:13 sh loop2
artem        1088      951  0 17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem         951      675  0 16:49 tty1        00:00:00 -bash
artem        1086      951 52 17:20 tty1        00:01:13 sh loop2
artem        1089      951  0 17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem         951      675  0 16:49 tty1        00:00:00 -bash
artem        1086      951 49 17:20 tty1        00:01:13 sh loop2
artem        1090      951  0 17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem         951      675  0 16:49 tty1        00:00:00 -bash
artem        1086      951 47 17:20 tty1        00:01:13 sh loop2
artem        1091      951  0 17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
artem         951      675  0 16:49 tty1        00:00:00 -bash
artem        1086      951 46 17:20 tty1        00:01:13 sh loop2
artem        1092      951  0 17:22 tty1        00:00:00 ps -f
artem@artemserver:~$ kill -9 1086
[1]+  Killed                  sh loop2
artem@artemserver:~$

```

Рисунок 9 – Результат команды kill -9 1086

1.9 Запустить в фоне процесс loop. Не останавливая, осмотреть несколько раз: ps -f

Мы видим, что процент времени CPU, используемого процессом изменяется и вырос до 99%, также мы видим, что общее время процессора, затраченное на выполнение процесса, увеличивается. Результат выполнения показан на рисунке 10.

```

artem      1086      951 46 17:20 tty1      00:01:13 sh loop2
artem      1092      951 0 17:22 tty1      00:00:00 ps -f
artem@artemserver:~$ kill -9 1086
[1]+  Killed                  sh loop2
artem@artemserver:~$ sh loop&
[1] 1096
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem         951       675  0  16:49 tty1      00:00:00 -bash
artem        1096      951 99  17:43 tty1      00:02:42 sh loop
artem        1098      951  0  17:46 tty1      00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem         951       675  0  16:49 tty1      00:00:00 -bash
artem        1096      951 99  17:43 tty1      00:02:43 sh loop
artem        1099      951  0  17:46 tty1      00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem         951       675  0  16:49 tty1      00:00:00 -bash
artem        1096      951 98  17:43 tty1      00:02:45 sh loop
artem        1100      951  0  17:46 tty1      00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem         951       675  0  16:49 tty1      00:00:00 -bash
artem        1096      951 98  17:43 tty1      00:02:46 sh loop
artem        1101      951  0  17:46 tty1      00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem         951       675  0  16:49 tty1      00:00:00 -bash
artem        1096      951 98  17:43 tty1      00:02:47 sh loop
artem        1102      951  0  17:46 tty1      00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem         951       675  0  16:49 tty1      00:00:00 -bash
artem        1096      951 99  17:43 tty1      00:03:13 sh loop
artem        1103      951  0  17:47 tty1      00:00:00 ps -f
artem@artemserver:~$ _

```

Рисунок 10 – Запуск фонового процесса

#### 1.10 Завершить процесс loop командой kill -15 PID.

При использовании команды kill -15 происходит корректное завершение процесса loop. Результат выполнения показан на рисунке 11.

```

artem@artemserver:~$ kill -15 1096
artem@artemserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
artem         951       675  0  16:49 tty1      00:00:00 -bash
artem        1105      951  0  17:52 tty1      00:00:00 ps -f
[1]+  Terminated              sh loop
artem@artemserver:~$ _

```

Рисунок 11 – Корректное завершение процесса loop

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

Результат выполнения команд sh loop& и kill -9 1114 представлены на рисунке 12.

```

artem@artemserver:~$ sh loop&
[1] 1114
artem@artemserver:~$ kill -9 1114
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1115      951  0  18:02 tty1        00:00:00 ps -f
[1]+  Killed                  sh loop
artem@artemserver:~$

```

Рисунок 12 – Запуск в фоне loop

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

Было запущено три процесса `loop` в фоновом режиме. Спустя некоторое время первый и третий процесс были остановлены сигналом `STOP`. Мы видим, что параметр `C` у данных процессов уменьшился, параметр `TIME` остановился. Результат выполнения показан на рисунке 13

```

artem@artemserver:~$ sh loop&
[1] 1122
artem@artemserver:~$ sh loop&
[2] 1123
artem@artemserver:~$ sh loop&
[3] 1124
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1116      951  0  18:04 tty1        00:00:00 bash
artem        1122     1116  42  18:06 tty1        00:00:05 sh loop
artem        1123     1116  36  18:06 tty1        00:00:03 sh loop
artem        1124     1116  32  18:06 tty1        00:00:02 sh loop
artem        1125     1116  0  18:06 tty1        00:00:00 ps -f
artem@artemserver:~$ kill -19 1122
artem@artemserver:~$ kill -19 1124

[1]+  Stopped                  sh loop
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1116      951  0  18:04 tty1        00:00:00 bash
artem        1122     1116  30  18:06 tty1        00:00:19 sh loop
artem        1123     1116  38  18:06 tty1        00:00:24 sh loop
artem        1124     1116  32  18:06 tty1        00:00:19 sh loop
artem        1127     1116  0  18:07 tty1        00:00:00 ps -f

[3]+  Stopped                  sh loop
artem@artemserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
artem         951      675  0  16:49 tty1        00:00:00 -bash
artem        1116      951  0  18:04 tty1        00:00:00 bash
artem        1122     1116  21  18:06 tty1        00:00:19 sh loop
artem        1123     1116  59  18:06 tty1        00:00:54 sh loop
artem        1124     1116  21  18:06 tty1        00:00:19 sh loop
artem        1128     1116  0  18:07 tty1        00:00:00 ps -f
artem@artemserver:~$

```

Рисунок 13 – Работа с тремя процессами в фоновом режиме

Затем запустим первый процесс и остановим второй. Мы видим, что первый процесс продолжил работу, второй остановился, а третий так и остался остановлен. Результат выполнения показан на рисунке 14.

```

artem@artemserver:~$ ps -f
UID          PID     PPID  C STIME TTY          TIME CMD
artem         951       675  0 16:49 tty1        00:00:00 -bash
artem        1116       951  0 18:04 tty1        00:00:00 bash
artem        1122      1116  8 18:06 tty1        00:00:38 sh loop
artem        1123      1116 86 18:06 tty1        00:06:23 sh loop
artem        1124      1116  4 18:06 tty1        00:00:19 sh loop
artem        1136      1116  0 18:13 tty1        00:00:00 ps -f
artem@artemserver:~$ kill -18 1122
artem@artemserver:~$ kill -19 1123
artem@artemserver:~$ ps -f
UID          PID     PPID  C STIME TTY          TIME CMD
artem         951       675  0 16:49 tty1        00:00:00 -bash
artem        1116       951  0 18:04 tty1        00:00:00 bash
artem        1122      1116 11 18:06 tty1        00:00:55 sh loop
artem        1123      1116 83 18:06 tty1        00:06:37 sh loop
artem        1124      1116  4 18:06 tty1        00:00:19 sh loop
artem        1137      1116  0 18:14 tty1        00:00:00 ps -f

[2]+  Stopped                  sh loop
artem@artemserver:~$ ps -f
UID          PID     PPID  C STIME TTY          TIME CMD
artem         951       675  0 16:49 tty1        00:00:00 -bash
artem        1116       951  0 18:04 tty1        00:00:00 bash
artem        1122      1116 14 18:06 tty1        00:01:10 sh loop
artem        1123      1116 81 18:06 tty1        00:06:37 sh loop
artem        1124      1116  3 18:06 tty1        00:00:19 sh loop
artem        1138      1116  0 18:14 tty1        00:00:00 ps -f
artem@artemserver:~$ ps -f
UID          PID     PPID  C STIME TTY          TIME CMD
artem         951       675  0 16:49 tty1        00:00:00 -bash
artem        1116       951  0 18:04 tty1        00:00:00 bash
artem        1122      1116 16 18:06 tty1        00:01:22 sh loop
artem        1123      1116 79 18:06 tty1        00:06:37 sh loop
artem        1124      1116  3 18:06 tty1        00:00:19 sh loop
artem        1139      1116  0 18:14 tty1        00:00:00 ps -f
artem@artemserver:~$ _

```

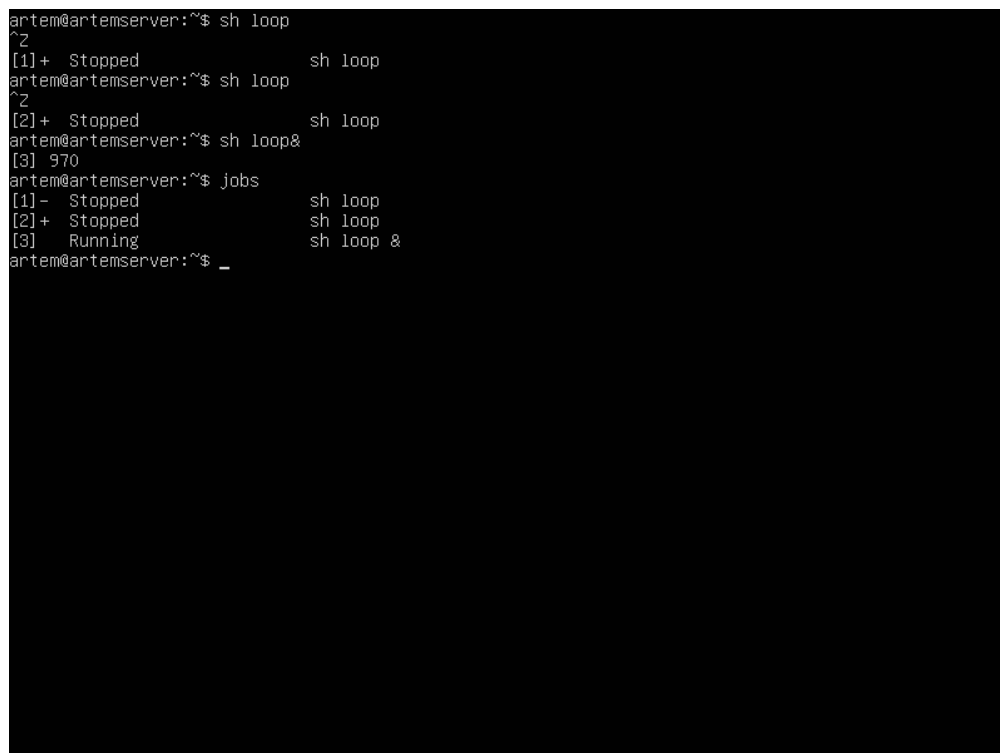
Рисунок 14 – Запуск первого и остановка второго процесса

## 2. Задание 2

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

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

Запустим два процесса `loop` в интерактивном режиме командой `sh loop`, и один процесс `loop` в фоновом режиме командой `sh loop&`. Результат выполнений команд представлен на рисунке 15.



```
artem@artemserver:~$ sh loop
^Z
[1]+  Stopped                  sh loop
artem@artemserver:~$ sh loop
^Z
[2]+  Stopped                  sh loop
artem@artemserver:~$ sh loop&
[3]  970
artem@artemserver:~$ jobs
[1]-  Stopped                  sh loop
[2]+  Stopped                  sh loop
[3]   Running                  sh loop &
artem@artemserver:~$ _
```

Рисунок 15 – Запуск трех задач в разных режимах

2.2 Перевести одну из задач, выполняющихся в интерактивном режиме, в фоновый режим

Переведем первый процесс в фоновый режим командой `bg 1`. Результат выполнения проверим командой `jobs`. Результат перевода первого процесса в фоновый режим показан на рисунке 16.



```

artem@artemserver:~$ sh loop
^Z
[1]+  Stopped                  sh loop
artem@artemserver:~$ sh loop
^Z
[2]+  Stopped                  sh loop
artem@artemserver:~$ sh loop&
[3] 970
artem@artemserver:~$ jobs
[1]-  Stopped                  sh loop
[2]+  Stopped                  sh loop
[3]   Running                  sh loop &
artem@artemserver:~$ bg 1
[1]- sh loop &
artem@artemserver:~$ jobs
[1]   Running                  sh loop &
[2]+  Stopped                  sh loop
[3]-  Running                  sh loop &
artem@artemserver:~$ _

```

Рисунок 16 – Перевод первого процесса в фоновый режим

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

Переведем второй процесс в фоновый режим, затем первый и третий процессы переведем в интерактивный режим. Результат выполнения показан на рисунке 17

```

^Z
[2]+  Stopped                  sh loop
artem@artemserver:~$ sh loop&
[3]  970
artem@artemserver:~$ jobs
[1]-  Stopped                  sh loop
[2]+  Stopped                  sh loop
[3]   Running                  sh loop &
artem@artemserver:~$ bg 1
[1]-  sh loop &
artem@artemserver:~$ jobs
[1]   Running                  sh loop &
[2]+  Stopped                  sh loop
[3]-  Running                  sh loop &
artem@artemserver:~$ bg 2
[2]+  sh loop &
artem@artemserver:~$ jobs
[1]   Running                  sh loop &
[2]-  Running                  sh loop &
[3]+  Running                  sh loop &
artem@artemserver:~$ fg 3
sh loop
^Z
[3]+  Stopped                  sh loop
artem@artemserver:~$ jobs
[1]   Running                  sh loop &
[2]-  Running                  sh loop &
[3]+  Stopped                  sh loop
artem@artemserver:~$ fg 1
sh loop
^Z
[1]+  Stopped                  sh loop
artem@artemserver:~$ jobs
[1]+  Stopped                  sh loop
[2]   Running                  sh loop &
[3]-  Stopped                  sh loop
artem@artemserver:~$

```

Рисунок 17 – Перевод задач в различные режимы работы

2.4 Создать именованный канал для архивирования и осуществить передачу в канал

Создадим новый именованный канал `newchannel` командой `mkfifo`. Для передачи списка файлов домашнего каталога вместе с подкаталогами воспользуемся командами `gzip -9 -c < newchan > result` и `ls -R >newchannel`. Результат команд проверим командой `zcat result`. Результат выполнения показан на рисунках 18 и 19.

```

artem@artemserver:~$ ls
loop  loop2  newchannel  newdir
artem@artemserver:~$ gzip -9 -c < newchannel > result
artem@artemserver:~$ zcat result
.:
loop
loop2
newchannel
newdir
result

./newdir:
file1.txt
newdir2

./newdir/newdir2:
file2.txt
artem@artemserver:~$

```

Рисунок 18 – Полученной результат

```

Ubuntu 20.04.1 LTS artemserver tty2
artemserver login: artem
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

 * Introducing self-healing high availability clustering for MicroK8s!
   Super simple, hardened and opinionated Kubernetes for production.

   https://microk8s.io/high-availability

39 updates can be installed immediately.
9 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection
or proxy settings

Last login: Fri Oct 30 14:49:33 UTC 2020 on tty1
artem@artemserver:~$ ls -R > newchannel
artem@artemserver:~$ _

```

Рисунок 19 – Ввод команды ls -R >newchannel

Для передачи одного каталога вместе с файлами и подкаталогами воспользуемся командами `gzip -9 -c < newchan > result_new`, `ls -R >newchannel`,

ls -l >newchanel и ls -l newdir/ >newchanel, результат проверим командой zcat result\_new. Пример каталога показан на рисунке 20. Результат выполнения представлен на рисунка 21,22,23.

```
artem@artemserver:~$ ls
loop  loop2  newchanel  newdir
artem@artemserver:~$ gzip -9 -c < newchanel > result
artem@artemserver:~$ zcat result
.:
loop
loop2
newchanel
newdir
result

./newdir:
file1.txt
newdir2

./newdir/newdir2:
file2.txt
artem@artemserver:~$ cd newdir
artem@artemserver:~/newdir$ ls -l
total 4
-rw-rw-r-- 1 artem artem  0 Oct 30 15:09 file1.txt
drwxrwxr-x 2 artem artem 4096 Oct 30 15:10 newdir2
artem@artemserver:~/newdir$ cd newdir2
artem@artemserver:~/newdir/newdir2$ ls -l
total 0
-rw-rw-r-- 1 artem artem 0 Oct 30 15:10 file2.txt
artem@artemserver:~/newdir/newdir2$
```

Рисунок 20 – Пример каталога

```
artem@artemserver:~$ cd newdir/
artem@artemserver:~/newdir$ ls -R > newchanel
artem@artemserver:~/newdir$ ls -l > newchanel
artem@artemserver:~/newdir$ _
```

Рисунок 20 – Ввод команд ls -R >newchanel, ls -l >newchanel

```
artem@artemserver:~$ ls -l newdir/ > newchanel
artem@artemserver:~$ ls -l newdir/ > newchanel
artem@artemserver:~$ _
```

Рисунок 22 – Ввод команды `ls -l newdir/ >newchanel`

```
artem@artemserver:~$ gzip -9 -c < newchanel > result_new
artem@artemserver:~$ zcat result_new
total 8
-rw-rw-r-- 1 artem artem    0 Oct 30 15:09 file1.txt
-rw-rw-r-- 1 artem artem  165 Oct 30 16:33 newchanel
drwxrwxr-x 2 artem artem 4096 Oct 30 15:10 newdir2
artem@artemserver:~$ _
```

Рисунок 23 – Результат выполнения

### 3. Задание 3

#### Вариант 9.

1. Отобразить информацию о трех заданных процессах в реальном режиме, одному из процессов переназначить приоритет, не выходя из команды.

2. Послать сигнал на безусловное завершение (по имени и по номеру сигнала) процессу по его имени, установить подтверждение завершения.

3. Выведите статистику работы системы с момента последней загрузки.

3.1 Отобразить информацию о трех заданных процессах в реальном режиме, одному из процессов переназначить приоритет, не выходя из команды

Запустим процессы loop, loop2, loop3 в фоновом режиме. Командой renice изменим приоритет на 10 у процесса loop. Результат выполнения показан на рисунке 24.

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

* Introducing self-healing high availability clustering for MicroK8s!
  Super simple, hardened and opinionated Kubernetes for production.

  https://microk8s.io/high-availability

39 updates can be installed immediately.
9 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Fri Oct 30 17:05:08 UTC 2020 on tty1
artem@artemserver:~$ sudo su
[sudo] password for artem:
root@artemserver:/home/artem# sh loop&
[1] 959
root@artemserver:/home/artem# sh loop2&
[2] 960
root@artemserver:/home/artem# sh loop3&
[3] 961
root@artemserver:/home/artem# ps -l
F S  UID      PID      PPID  C PRI  NI ADDR S2  WCHAN  TTY          TIME CMD
4 S   0       677        1  0  80   0 - 1493 do_wai tty1      00:00:00 login
4 S   0       947      938  0  80   0 - 2051 poll_s tty1      00:00:00 sudo
4 S   0       951      947  0  80   0 - 1781 do_wai tty1      00:00:00 su
4 S   0       952      951  0  80   0 - 1498 do_wai tty1      00:00:00 bash
0 R   0       959      952 51  80   0 - 652 -      tty1      00:00:09 sh
0 R   0       960      952 39  80   0 - 652 -      tty1      00:00:05 sh
0 R   0       961      952 33  80   0 - 652 -      tty1      00:00:03 sh
0 R   0       962      952  0  80   0 - 1890 -      tty1      00:00:00 ps
root@artemserver:/home/artem# renice -n 10 -p 959
```

Рисунок 24 – Изменение приоритета

3.2 Послать сигнал на безусловное завершение (по имени и по номеру сигнала) процессу по его имени, установить подтверждение завершения.

С помощью команды `pkill` мы можем послать сигнал по имени процесса. Воспользуемся командами `pkill -KILL loop` и `pkill -KILL` для безусловного завершения процессов `loop` и `loop2`. Результат выполнения показан на рисунках 25, 26.

```
root@artemserver:/home/artem# pkill -KILL loop
root@artemserver:/home/artem# jobs
[1]  Killed                  sh loop
[2]-  Running                sh loop2 &
[3]+  Running                sh loop3 &
```

Рисунок 25 – Результат команды `pkill -KILL loop`

```
root@artemserver:/home/artem# pkill -9 loop2
root@artemserver:/home/artem# jobs
[2]-  Killed                  sh loop2
[3]+  Running                sh loop3 &
```

Рисунок 26 – Результат команды `pkill -9 loop`

3.3 Выведите статистику работы системы с момента последней загрузки.

Первая строка выводит данные по порядку: текущее время, время работы системы, количество открытых пользовательских сессий, среднюю загрузку системы (load average), три значения соответствуют загрузке в последнюю минуту, пять минут и пятнадцать минут соответственно.

Вторая строка выводит следующие данные: общее количество процессов в системе, количество работающих в данный момент процессов, количество ожидающих событий процессов, количество остановленных процессов, количество процессов, ожидающих родительский процесс для передачи статуса завершения.

В третьей строке приводится информация об использовании центрального процессора. Если вы просуммируете все значения, в результате у вас должно получиться 100%. Давайте рассмотрим предназначение полей этой строки по порядку: процент использования центрального процессора пользовательскими процессам, процент использования центрального процессора системными процессами, процент использования центрального

процессора процессами с приоритетом, процент использования центрального процессора процессами, ожидающими завершения операций ввода-вывода, процент использования центрального процессора обработчиками аппаратных прерываний (аппаратные прерывания), процент использования центрального процессора обработчиками программных прерываний (программные прерывания), количество ресурсов центрального процессора "заимствованных" у виртуальной машины гипервизором для других задач (таких, как запуск другой виртуальной машины); это значение будет равно нулю на настольных компьютерах и серверах, не использующих виртуальные машины.

В четвертой и пятой строке выводится информация об использовании физической оперативной памяти и раздела подкачки соответственно. Значения в порядке следования: общее количество памяти, количество используемой памяти, количество свободной памяти, количество памяти в кэше буферов.

Последним источником информации является список процессов, отсортированный по степени использования центрального процессора (по умолчанию). Давайте рассмотрим значения столбцов списка:

PID - идентификатор процесса;

USER - имя пользователя, который является владельцем процесса;

PR - приоритет процесса;

NI - значение "NICE", влияющие на приоритет процесса;

VIRT - объем виртуальной памяти, используемый процессом;

RES - объем физической памяти, используемый процессом;

SHR - объем разделяемой памяти процесса;

S - указывает на статус процесса: S=sleep (ожидает событий) R=running (работает) Z=zombie (ожидает родительский процесс);

%CPU - процент использования центрального процессора данным процессом;

%MEM - процент использования оперативной памяти данным процессом;



TIME+ - общее время активности процесса;

COMMAND - имя процесса;

Пример команды top показан на рисунке 27.

```
top - 18:13:29 up 17 min, 1 user, load average: 1.09, 1.54, 1.39
Tasks: 96 total, 2 running, 94 sleeping, 0 stopped, 0 zombie
%Cpu(s): 66.7 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 33.3 si, 0.0 st
MiB Mem : 3936.4 total, 3522.1 free, 133.0 used, 281.3 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 3582.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
961	root	20	0	2608	544	476	R	99.9	0.0	9:51.68	sh
1	root	20	0	101996	11700	8632	S	0.0	0.3	0:06.28	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblockd
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.89	ksoftirqd/0
10	root	20	0	0	0	0	I	0.0	0.0	0:00.51	rcu_sched
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.03	migration/0
12	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
16	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
23	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
24	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
70	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
71	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
72	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
73	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
74	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
75	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
76	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
77	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq

Рисунок 27 – Пример команды top

## Вывод

В ходе выполнения данной лабораторной работы были получены знания о создании процессов в ОС Linux, сигналах, работе процессов в разных режимах.