

**Липецкий государственный технический университет**

**Факультет автоматизации и информатики**

**Кафедра автоматизированных систем управления**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**по дисциплине «Операционная система Linux»**

**Процессы в операционной системе Linux**

Студент

Мастылина А.А.

Группа АИ-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

## Оглавление

Цель работы .....	3
Задание кафедры.....	4
Ход работы .....	6
Вывод .....	17

## Цель работы

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

## Задание кафедры

### Часть I

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

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

  
Loop2:  

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

### Часть II

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

режиме, одну - в фоновом.

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

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

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

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

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

### Часть III Индивидуальное задание

#### Вариант 1

1. Сгенерировать информацию — полный листинг о всех процессах системы.

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

3. Определить идентификаторы процессов, владельцем которых не является root.

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

## Ход работы

### Часть I

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

```
Last login: Fri Oct 30 07:49:38 UTC 2020 on tty1
anna@annaserver:~$ cd /boot
anna@annaserver:/boot$ ls
System.map-5.4.0-52-generic  initrd.img          lost+found             vmlinuz.old
config-5.4.0-52-generic     initrd.img-5.4.0-52-generic  vmlinuz
grub                        initrd.img.old      vmlinuz-5.4.0-52-generic
```

Рисунок 1 – Загрузка пользователем, версия ядра

Исходя из рисунка 1 ядро имеет версию 5.4.0

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

```
anna@annaserver:/boot$ ps -f
UID          PID    PPID  C   STIME TTY          TIME CMD
anna         885      614  0   06:39 tty1        00:00:00 -bash
anna         926      885  0   06:57 tty1        00:00:00 ps -f
anna@annaserver:/boot$
```

Рисунок 2 – Просмотр процессов `ps -f`

Поясним просмотр процессов.

UID - имя пользователя, от имени которого работает процесс;

PID - идентификатор пользователя;

PPID - идентификатор родительского процесса пользователя;

C - расходование ресурсов процессора, в процентах;

STIME - время, когда процесс был запущен;

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

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

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

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

Loop:

while true; do true; done



Рисунок – 3 Написание сценария loop



Рисунок 4 – Написание сценария loop2

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

Следующим шагом приведём пример каталога на наличие `loop.sh` и `loop2.sh`. Результат работы представлен на рисунке 6.

[illegible]

8



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

[illegible]

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

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

```
anna@annaserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
anna         867     616  0 07:15 tty1          00:00:00 -bash
anna         903     867  4 07:25 tty1          00:00:23 sh loop2.sh
anna         918     867  0 07:34 tty1          00:00:00 ps -f
anna@annaserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
anna         867     616  0 07:15 tty1          00:00:00 -bash
anna         903     867  3 07:25 tty1          00:00:23 sh loop2.sh
anna         919     867  0 07:35 tty1          00:00:00 ps -f
anna@annaserver:~$ _
```

Рисунок 8 – Просмотр листинга процессов

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

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

```
anna@annaserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
anna         867       616  0  07:15 tty1        00:00:00 -bash
anna         903       867  2  07:25 tty1        00:00:23 sh loop2.sh
anna         921       867  0  07:39 tty1        00:00:00 ps -f
anna@annaserver:~$ kill -9 903
[1]+  Killed                  sh loop2.sh
anna@annaserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
anna         867       616  0  07:15 tty1        00:00:00 -bash
anna         922       867  0  07:40 tty1        00:00:00 ps -f
anna@annaserver:~$ _
```

Рисунок 9 – Уничтожение процесса loop2.sh

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

```
anna@annaserver:~$ cd /
anna@annaserver:/$ cd -
/home/anna
anna@annaserver:~$ sh loop.sh&
[1] 891
anna@annaserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
anna         874       613  0  15:37 tty1        00:00:00 -bash
anna         891       874  99  15:46 tty1        00:00:03 sh loop.sh
anna         892       874  0  15:46 tty1        00:00:00 ps -f
anna@annaserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
anna         874       613  0  15:37 tty1        00:00:00 -bash
anna         891       874  99  15:46 tty1        00:00:13 sh loop.sh
anna         894       874  0  15:46 tty1        00:00:00 ps -f
```

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

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

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

```
anna@annaserver:~$ kill -15 891
anna@annaserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
anna         874       613  0  15:37 tty1        00:00:00 -bash
anna         897       874  0  15:52 tty1        00:00:00 ps -f
[1]+  Terminated              sh loop.sh
anna@annaserver:~$
```

Рисунок 11 – Заверение процесса loop.sh

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

```
anna@annaserver:~$ sh loop.sh&
[1] 909
anna@annaserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
anna         874      613  0 15:37 tty1        00:00:00 -bash
anna         909      874  88 15:56 tty1        00:00:02 sh loop.sh
anna         910      874  0 15:56 tty1        00:00:00 ps -f
anna@annaserver:~$ kill -9 909
anna@annaserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
anna         874      613  0 15:37 tty1        00:00:00 -bash
anna         911      874  0 15:57 tty1        00:00:00 ps -f
[1]+  Killed                  sh loop.sh
anna@annaserver:~$
```

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

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

```
anna@annaserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
anna         874      613  0 15:37 tty1        00:00:00 -bash
anna         914      874  0 15:59 tty1        00:00:00 ps -f
anna@annaserver:~$ bash
anna@annaserver:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
anna         874      613  0 15:37 tty1        00:00:00 -bash
anna         915      874  1 15:59 tty1        00:00:00 bash
anna         921      915  0 15:59 tty1        00:00:00 ps -f
anna@annaserver:~$
```

Рисунок 13 – Запуск еще одного экземпляра

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

```
anna@annaserver:~$ sh loop.sh&
[2] 900
anna@annaserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
anna         880     617  0  15:03 tty1          00:00:00 -bash
anna         899     880  82  15:06 tty1          00:00:09 sh loop.sh
anna         900     880  37  15:06 tty1          00:00:01 sh loop.sh
anna         901     880  0  15:06 tty1          00:00:00 ps -f
anna@annaserver:~$ kill -19 899 900
anna@annaserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
anna         880     617  0  15:03 tty1          00:00:00 -bash
anna         899     880  54  15:06 tty1          00:00:32 sh loop.sh
anna         900     880  46  15:06 tty1          00:00:23 sh loop.sh
anna         902     880  0  15:07 tty1          00:00:00 ps -f

[1]+  Stopped                  sh loop.sh
[2]-  Stopped                  sh loop.sh
anna@annaserver:~$ sh loop.sh&
[3] 903
anna@annaserver:~$ sh loop.sh&
[4] 904
anna@annaserver:~$ kill -19 903 904
anna@annaserver:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
anna         880     617  0  15:03 tty1          00:00:00 -bash
anna         899     880  27  15:06 tty1          00:00:32 sh loop.sh
anna         900     880  21  15:06 tty1          00:00:23 sh loop.sh
anna         903     880  62  15:07 tty1          00:00:22 sh loop.sh
anna         904     880  45  15:07 tty1          00:00:11 sh loop.sh
anna         907     880  0  15:08 tty1          00:00:00 ps -f

[3]+  Stopped                  sh loop.sh
[4]-  Stopped                  sh loop.sh
anna@annaserver:~$
```

Рисунок 14 – Запуск процессов в фоне и их остановка

## Часть II

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

```
anna@annaserver:~$ sh loop.sh
^Z
[1]+  Stopped                  sh loop.sh
anna@annaserver:~$
anna@annaserver:~$ sh loop.sh
^Z
[2]+  Stopped                  sh loop.sh
anna@annaserver:~$ sh loop.sh&
[3] 912
anna@annaserver:~$ jobs
[1]-  Stopped                  sh loop.sh
[2]+  Stopped                  sh loop.sh
[3]   Running                  sh loop.sh &
anna@annaserver:~$
```

Рисунок 15 – Запуск на выполнение трёх задач

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

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

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

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

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

Рисунок 17 – Эксперименты по переводу задач

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

```
anna@annaserver:~$ ls -l
total 8
-rw-rw-r-- 1 anna anna 26 Oct 31 13:25 loop.sh
-rw-rw-r-- 1 anna anna 40 Oct 31 13:25 loop2.sh
anna@annaserver:~$ mkfifo channel
anna@annaserver:~$ ls -l
total 8
-rw-rw-r-- 1 anna anna  0 Nov  2 18:05 channel
-rw-rw-r-- 1 anna anna 26 Oct 31 13:25 loop.sh
-rw-rw-r-- 1 anna anna 40 Oct 31 13:25 loop2.sh
anna@annaserver:~$ _
```

Рисунок 18 – Создание именованного канала

```
anna@annaserver:~$ ls -l
total 12
-rw-rw-r-- 1 anna anna  0 Nov  2 18:05 channel
-rw-rw-r-- 1 anna anna 26 Oct 31 13:25 loop.sh
-rw-rw-r-- 1 anna anna 40 Oct 31 13:25 loop2.sh
drwxrwxr-x 3 anna anna 4096 Nov  2 18:12 new
anna@annaserver:~$ ls -R > channel&
[1] 903
```

Рисунок 19 – Передача в канал листинг домашнего каталога

```
anna@annaserver:~$ mkdir new
anna@annaserver:~$ cd new
anna@annaserver:~/new$ touch 1.txt
anna@annaserver:~/new$ touch 2.txt
anna@annaserver:~/new$ mkdir new1
anna@annaserver:~/new$ cd new1
anna@annaserver:~/new/new1$ touch 3.txt
anna@annaserver:~/new/new1$ cd ..
anna@annaserver:~/new$ ls -l
total 4
-rw-rw-r-- 1 anna anna  0 Nov  2 18:12 1.txt
-rw-rw-r-- 1 anna anna  0 Nov  2 18:12 2.txt
drwxrwxr-x 2 anna anna 4096 Nov  2 18:13 new1
anna@annaserver:~/new$
```

Рисунок 20 – Создание нового каталога

```

anna@annaserver:~$ gzip -9 -c < channel > res
[1]+  Done                  ls --color=auto -R > channel
anna@annaserver:~$ zcat res
.:
channel
loop.sh
loop2.sh
new

./new:
1.txt
2.txt
new1

./new/new1:
3.txt
anna@annaserver:~$ _

```

Рисунок 21 – Листинг домашнего каталога

```

anna@annaserver:~$ ls -l new/ > channel8
[1] 879
anna@annaserver:~$ gzip -9 -c < channel > res
[1]+  Done                  ls --color=auto -l new/ > channel
anna@annaserver:~$ zcat res
total 4
-rw-rw-r-- 1 anna anna    0 Nov  2 18:12 1.txt
-rw-rw-r-- 1 anna anna    0 Nov  2 18:12 2.txt
drwxrwxr-x 2 anna anna 4096 Nov  2 18:13 new1
anna@annaserver:~$

```

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

### Часть III

Сгенерируем информацию — полный листинг о всех процессах системы. `ps -eF`. Результат работы представлен на рисунке 23.

```

root      368      2  0      0      0  0 07:14 ?        00:00:00 [iprt-VBoxHQueue]
root      488      2  0      0      0  0 07:14 ?        00:00:00 [kaluad]
root      489      2  0      0      0  0 07:14 ?        00:00:00 [kmpath_rdcad]
root      490      2  0      0      0  0 07:14 ?        00:00:00 [kmpathd]
root      491      2  0      0      0  0 07:14 ?        00:00:00 [kmpath_handlerd]
root      492      1  0 70072 18096  0 07:14 ?        00:00:00 /sbin/multipathd -d -s
root      502      2  0      0      0  0 07:14 ?        00:00:00 [loop0]
root      505      2  0      0      0  0 07:14 ?        00:00:00 [loop1]
root      508      2  0      0      0  0 07:14 ?        00:00:00 [loop2]
root      509      2  0      0      0  0 07:14 ?        00:00:00 [loop3]
root      510      2  0      0      0  0 07:14 ?        00:00:00 [loop4]
root      511      2  0      0      0  0 07:14 ?        00:00:00 [loop5]
root      512      2  0      0      0  0 07:14 ?        00:00:00 [jbd2/sda2-8]
root      513      2  0      0      0  0 07:14 ?        00:00:00 [ext4-rsv-conver]
systemd+  525      1  0 22597 6288   0 07:14 ?        00:00:00 /lib/systemd/systemd-timesyncd
systemd+  566      1  0 6685 7632   0 07:15 ?        00:00:00 /lib/systemd/systemd-networkd
systemd+  568      1  0 5978 12240  0 07:15 ?        00:00:00 /lib/systemd/systemd-resolved
root      582      1  0 59511 9316   0 07:15 ?        00:00:00 /usr/lib/accountsservice/account
root      585      1  0 1392 2904   0 07:15 ?        00:00:00 /usr/sbin/cron -f
message+  586      1  0 1903 4628   0 07:15 ?        00:00:00 /usr/bin/dbus-daemon --system --
root      594      1  0 6570 17860  0 07:15 ?        00:00:00 /usr/bin/python3 /usr/bin/networ
syslog    596      1  0 56081 4756   0 07:15 ?        00:00:00 /usr/sbin/rsyslogd -n -iNONE
root      598      1  0 156858 28788  0 07:15 ?        00:00:01 /usr/lib/snapd/snapd
root      599      1  0 4201 8024   0 07:15 ?        00:00:00 /lib/systemd/systemd-logind
daemon    606      1  0 948 2284   0 07:15 ?        00:00:00 /usr/sbin/atd -f
root      616      1  0 1493 3944   0 07:15 tty1      00:00:00 /bin/login -p --
root      637      1  0 26274 20936  0 07:15 ?        00:00:00 /usr/bin/python3 /usr/share/unat
root      656      1  0 59100 9172   0 07:15 ?        00:00:00 /usr/lib/policykit-1/polkitd --n
anna      861      1  0 4638 9748   0 07:15 ?        00:00:00 /lib/systemd/systemd --user
anna      862      861  0 42182 3468   0 07:15 ?        00:00:00 (sd-pam)
anna      867      616  0 1800 5132   0 07:15 tty1      00:00:00 -bash
root      898      2  0      0      0  0 07:20 ?        00:00:00 [kworker/u2:2-events_power_effic
root      915      2  0      0      0  0 07:29 ?        00:00:00 [kworker/0:2]
root      928      1  0 715 1800   0 07:57 tty2      00:00:00 /sbin/agetty -o -p -- \u --nocr
root      941      2  0      0      0  0 08:00 ?        00:00:00 [kworker/u2:0-events_unbound]
anna      943      867  0 1908 3220   0 08:00 tty1      00:00:00 ps -eF
anna@annaserver:~$ _

```

Рисунок 23 – Полный листинг о всех процессах системы

Завершим выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершим с помощью сигнала SIGTERM, задав его имя, второй — с помощью сигнала SIGKILL, задав его номер. Результат работы представлен на рисунке 24.

```
anna@annaserver:~$ sh loop.sh&
[1] 961
anna@annaserver:~$ sh loop.sh&
[2] 962
anna@annaserver:~$ ps -f
UID      PID    PPID  C  STIME TTY          TIME CMD
anna      867      616  0  07:15 tty1        00:00:00 -bash
anna      961      867  67  08:04 tty1        00:00:05 sh loop.sh
anna      962      867  48  08:04 tty1        00:00:02 sh loop.sh
anna      963      867  0  08:04 tty1        00:00:00 ps -f
anna@annaserver:~$ kill -9 961
anna@annaserver:~$ kill -15 962
[1]-  Killed                  sh loop.sh
anna@annaserver:~$ ps -f
UID      PID    PPID  C  STIME TTY          TIME CMD
anna      867      616  0  07:15 tty1        00:00:00 -bash
anna      964      867  0  08:04 tty1        00:00:00 ps -f
[2]+  Terminated            sh loop.sh
anna@annaserver:~$ _
```

Рисунок 24 – Завершение выполнения двух процессов

Определим идентификаторы процессов, владельцем которых не является root. Результат работы представлен на рисунке 25.

```
anna@annaserver:~$ ps -fu anna
UID      PID    PPID  C  STIME TTY          TIME CMD
anna      861       1  0  07:15 ?          00:00:00 /lib/systemd/systemd --user
anna      862      861  0  07:15 ?          00:00:00 (sd-pam)
anna      867      616  0  07:15 tty1        00:00:00 -bash
anna      968      867  0  08:06 tty1        00:00:00 ps -fu anna
anna@annaserver:~$ _
```

Рисунок 25 – Идификаторы процессов владельцем которых не является root



## Вывод

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