

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

По дисциплине «ОС Linux»

Процессы в операционной системе Linux

Студент

Чаплыгин И.С.

Группа ПИ-18

Руководитель

Доцент

Кургасов В.В.

Липецк 2020г

1. Цель работы

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

2. 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, пошлав сигнал `STOP`.
- 7) Посмотреть последовательно несколько раз `ps -f`. Записать сообщение, объяснить.
- 8) Убить процесс `loop2`, пошлав 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 11.

1. Вывести информацию о состоянии процессов системы в реальном режиме с сортировкой по убыванию значения приоритета. Отобразите информацию о состоянии процессов.
2. С помощью сигнала SIGKILL завершить все процессы, родителем которых является командный интерпретатор текущей сессии.
3. Вывести статистику использования памяти в байтах с обновлением каждые три секунды.
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.

Оглавление

1. Цель работы	2
2. Задание кафедры.....	3
3. Выполнение работы	6
3.1 Часть I	6
3.2 Часть II	13
3.3 Часть III.....	18
4. Вывод.....	22

3. Выполнение работы

3.1 Часть I

Для нахождения информации о файле с образом действующего ядра, используем команду «uname -a».

```
kosmonavtik@kosmos:~$ uname -a
Linux kosmos 5.4.0-48-generic #52-Ubuntu SMP Thu Sep 10 10:58:49 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
kosmonavtik@kosmos:~$
```

Рисунок 1 – Нахождение файла с образом ядра

Как можно увидеть, команда вывела нам информацию о файле с образом ядра: «5.4.0-48-generic». Первая цифра (5) – это мажорный номер версии, вторая цифра (4) – минорная версия. Следующая цифра (0) – номер ревизии, а число 48 относится к номеру сборки от разработчиков дистрибутива. При каждом добавлении патчей к ядру или исправлений, оно пересобирается, а к номеру добавляется это число.

После введения команды «ps -f» на экран консоли выводится список процессов, запущенных в текущей командной оболочке с подробной информацией.

```
kosmonavtik@kosmos:~$ uname -a
Linux kosmos 5.4.0-48-generic #52-Ubuntu SMP Thu Sep 10 10:58:49 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
kosmonavtik@kosmos:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kosmona+    1143      633  0 16:11 tty1      00:00:00 -bash
kosmona+    1230     1143  0 16:42 tty1      00:00:00 ps -f
kosmonavtik@kosmos:~$ _
```

Рисунок 2 – Вывод на экран списка процессов

Значения колонок в выводе:

UID – это имя пользователя, от имени которого работает процесс;

PID – идентификатор пользователя;

RPID – идентификатор родительского процесса пользователя;

C – расходование ресурсов процессора в процентах;

STIME – время, когда процесс был запущен;

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

TIME – общее время выполнения процесса;

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

После вызова редактора vi, нажимаем клавишу «i» для начала ввода текста.

```

while true; do true; done

```

Рисунок 3 – Ввод текста сценария loop

Для сохранения результата вводим команду: «: w!» и для выхода: «: q!»

Те же самые действия проводим для сценария loop2.

```
Hello
11
```



Рисунок 4 – Запуск и остановка loop2

После остановки процесса, введем команду «ps -f» несколько раз с некоторым промежутком по времени.

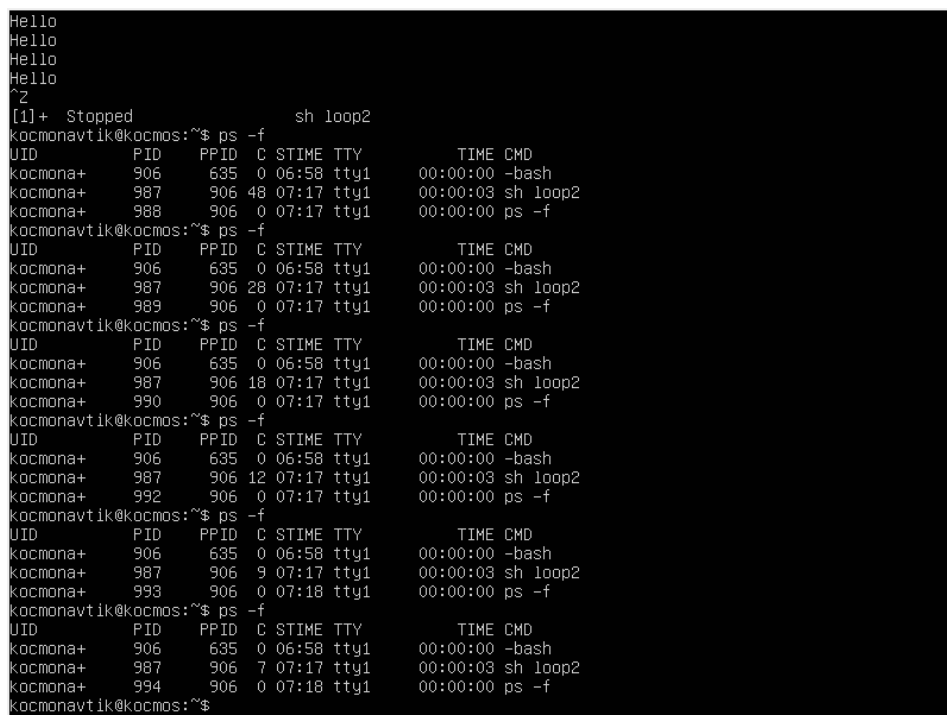


Рисунок 5 – Просмотр процессов

Как можно увидеть, после остановки процесса loop2, постепенно снижается потребление ресурса процессора этим же процессом.

Объяснить...

Можно убить процесс loop2, пошлав сигнал «kill -9 PID».

```
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  906      635  0  06:58 tty1        00:00:00 -bash
kocmona+  987      906  0  07:17 tty1        00:00:03 sh loop2
kocmona+  1060     906  0  07:34 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ kill -9 987
[1]+  Killed                  sh loop2
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  906      635  0  06:58 tty1        00:00:00 -bash
kocmona+  1061     906  0  07:34 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$
```

Рисунок 6 – Завершение процесса.

Где «-9» — это сигнал «KILL», которая убивает процесс, то есть немедленно прекращает выполнение. PID – идентификатор пользователя.

Отличие сигнала «KILL» от сигнала «STOP» в том, что в случае «STOP» процесс останавливается. В данном состоянии процесс не удаляется из таблицы процессов, но и не выполняется до тех пор, пока не получить сигнал о продолжении работы.

Для запуска процесса в фоновом режиме используется команда «sh имя&»

```
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  918      644  0  13:23 tty1        00:00:00 -bash
kocmona+  966      918  0  13:27 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ sh loop&
[1] 967
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  918      644  0  13:23 tty1        00:00:00 -bash
kocmona+  967      918  96  13:27 tty1        00:00:01 sh loop
kocmona+  968      918  0  13:27 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  918      644  0  13:23 tty1        00:00:00 -bash
kocmona+  967      918  93  13:27 tty1        00:00:04 sh loop
kocmona+  969      918  0  13:28 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  918      644  0  13:23 tty1        00:00:00 -bash
kocmona+  967      918  91  13:27 tty1        00:00:08 sh loop
kocmona+  970      918  0  13:28 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  918      644  0  13:23 tty1        00:00:00 -bash
kocmona+  967      918  99  13:27 tty1        00:00:12 sh loop
kocmona+  971      918  0  13:28 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  918      644  0  13:23 tty1        00:00:00 -bash
kocmona+  967      918  95  13:27 tty1        00:00:16 sh loop
kocmona+  972      918  0  13:28 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID      PID     PPID  C  STIME TTY          TIME CMD
kocmona+  918      644  0  13:23 tty1        00:00:00 -bash
kocmona+  967      918  99  13:27 tty1        00:00:21 sh loop
kocmona+  973      918  0  13:28 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$
```

Рисунок 7 – Запуск процесса в фоновом режиме

Как можно увидеть после запуска внешнего процесса «loop» практически весь ресурс процессора уходит на исполнение данного сценария.

Чтобы завершить процесс, напишем команду «kill -15 PID».

```
kocmonavtik@kocmos:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
kocmona+    918       644  0  13:23 tty1        00:00:00 -bash
kocmona+    967       918  99  13:27 tty1        00:01:05 sh loop
kocmona+    974       918  0  13:29 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ kill -15 967
kocmonavtik@kocmos:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
kocmona+    918       644  0  13:23 tty1        00:00:00 -bash
kocmona+    975       918  0  13:29 tty1        00:00:00 ps -f
[1]+  Terminated                  sh loop
kocmonavtik@kocmos:~$ _
```

Рисунок 8 – Завершение процесса «loop»

Данная команда отличается от «kill -9 PID» тем, что она корректно завершает выполнение процесса. Так же отличие в том, что команду «kill -9 PID» обрабатывает система, как и команду STOP.

Запускаем повторно фоновый процесс «loop» и убиваем его командой «kill -9 PID». Запускаем ещё один экземпляр оболочки bash.

```
kocmonavtik@kocmos:~$ sh loop&
[1] 1197
kocmonavtik@kocmos:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
kocmona+    1166    1061  0  14:09 tty1        00:00:00 -bash
kocmona+    1197    1166  99  14:46 tty1        00:00:11 sh loop
kocmona+    1198    1166  0  14:47 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ kill -9 1197
kocmonavtik@kocmos:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
kocmona+    1166    1061  0  14:09 tty1        00:00:00 -bash
kocmona+    1199    1166  0  14:47 tty1        00:00:00 ps -f
[1]+  Killed                        sh loop
kocmonavtik@kocmos:~$ bash
kocmonavtik@kocmos:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
kocmona+    1166    1061  0  14:09 tty1        00:00:00 -bash
kocmona+    1201    1166  1  14:56 tty1        00:00:00 bash
kocmona+    1207    1201  0  14:56 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ _
```

Рисунок 9 – Выполнение ряда команд.

На последующих рисунках приведено задание № 13.

```
kocmonavtik@kocmos:~$ sh loop&
[1] 923
kocmonavtik@kocmos:~$ sh loop&
[2] 924
kocmonavtik@kocmos:~$ sh loop&
[3] 925
kocmonavtik@kocmos:~$ sh loop&
[4] 926
kocmonavtik@kocmos:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kocmona+    902      626  0 15:06 tty1        00:00:00 -bash
kocmona+    915      902  0 15:06 tty1        00:00:00 bash
kocmona+    923      915  50 15:07 tty1        00:00:04 sh loop
kocmona+    924      915  30 15:07 tty1        00:00:01 sh loop
kocmona+    925      915  28 15:07 tty1        00:00:01 sh loop
kocmona+    926      915  28 15:07 tty1        00:00:00 sh loop
kocmona+    927      915  0 15:07 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kocmona+    902      626  0 15:06 tty1        00:00:00 -bash
kocmona+    915      902  0 15:06 tty1        00:00:00 bash
kocmona+    923      915  41 15:07 tty1        00:00:05 sh loop
kocmona+    924      915  29 15:07 tty1        00:00:03 sh loop
kocmona+    925      915  27 15:07 tty1        00:00:02 sh loop
kocmona+    926      915  27 15:07 tty1        00:00:02 sh loop
kocmona+    928      915  0 15:07 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kocmona+    902      626  0 15:06 tty1        00:00:00 -bash
kocmona+    915      902  0 15:06 tty1        00:00:00 bash
kocmona+    923      915  33 15:07 tty1        00:00:08 sh loop
kocmona+    924      915  26 15:07 tty1        00:00:06 sh loop
kocmona+    925      915  25 15:07 tty1        00:00:05 sh loop
kocmona+    926      915  25 15:07 tty1        00:00:05 sh loop
kocmona+    929      915  0 15:07 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$
```

Рисунок 10 – запуск 4 процессов

После запуска 4 процессов, как видно из рисунка, занимают все ресурсы процессора и со временем каждый процесс использует равную долю.

```
kocmonavtik@kocmos:~$ kill -19 923
kocmonavtik@kocmos:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kocmona+    902      626  0 15:06 tty1        00:00:00 -bash
kocmona+    915      902  0 15:06 tty1        00:00:00 bash
kocmona+    923      915  25 15:07 tty1        00:00:48 sh loop
kocmona+    924      915  25 15:07 tty1        00:00:47 sh loop
kocmona+    925      915  25 15:07 tty1        00:00:46 sh loop
kocmona+    926      915  25 15:07 tty1        00:00:46 sh loop
kocmona+    936      915  0 15:10 tty1        00:00:00 ps -f

[1]+  Stopped                  sh loop
kocmonavtik@kocmos:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kocmona+    902      626  0 15:06 tty1        00:00:00 -bash
kocmona+    915      902  0 15:06 tty1        00:00:00 bash
kocmona+    923      915  23 15:07 tty1        00:00:48 sh loop
kocmona+    924      915  26 15:07 tty1        00:00:53 sh loop
kocmona+    925      915  25 15:07 tty1        00:00:52 sh loop
kocmona+    926      915  25 15:07 tty1        00:00:52 sh loop
kocmona+    939      915  0 15:10 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ kill -19 925
kocmonavtik@kocmos:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kocmona+    902      626  0 15:06 tty1        00:00:00 -bash
kocmona+    915      902  0 15:06 tty1        00:00:00 bash
kocmona+    923      915  17 15:07 tty1        00:00:48 sh loop
kocmona+    924      915  28 15:07 tty1        00:01:17 sh loop
kocmona+    925      915  25 15:07 tty1        00:01:09 sh loop
kocmona+    926      915  28 15:07 tty1        00:01:16 sh loop
kocmona+    941      915  0 15:11 tty1        00:00:00 ps -f

[3]+  Stopped                  sh loop
kocmonavtik@kocmos:~$ _
```

Рисунок 11 – Остановка 2-х процессов.

После остановки 2-х процессов, потребление ими ресурсов процессора постепенно уменьшается и переходит на рабочие процессы.

```

UID      PID    PPID  C  STIME TTY          TIME CMD
kocmona+ 902     626  0 15:06 tty1        00:00:00 -bash
kocmona+ 915     902  0 15:06 tty1        00:00:00 bash
kocmona+ 923     915 18 15:07 tty1        00:01:19 sh loop
kocmona+ 924     915 33 15:07 tty1        00:02:23 sh loop
kocmona+ 925     915 16 15:07 tty1        00:01:09 sh loop
kocmona+ 926     915 33 15:07 tty1        00:02:22 sh loop
kocmona+ 947     915  0 15:14 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID      PID    PPID  C  STIME TTY          TIME CMD
kocmona+ 902     626  0 15:06 tty1        00:00:00 -bash
kocmona+ 915     902  0 15:06 tty1        00:00:00 bash
kocmona+ 923     915 19 15:07 tty1        00:01:32 sh loop
kocmona+ 924     915 33 15:07 tty1        00:02:35 sh loop
kocmona+ 925     915 14 15:07 tty1        00:01:09 sh loop
kocmona+ 926     915 33 15:07 tty1        00:02:34 sh loop
kocmona+ 948     915  0 15:15 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ kill -18 925
kocmonavtik@kocmos:~$ ps -f
UID      PID    PPID  C  STIME TTY          TIME CMD
kocmona+ 902     626  0 15:06 tty1        00:00:00 -bash
kocmona+ 915     902  0 15:06 tty1        00:00:00 bash
kocmona+ 923     915 20 15:07 tty1        00:01:46 sh loop
kocmona+ 924     915 32 15:07 tty1        00:02:50 sh loop
kocmona+ 925     915 13 15:07 tty1        00:01:11 sh loop
kocmona+ 926     915 32 15:07 tty1        00:02:49 sh loop
kocmona+ 949     915  0 15:15 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ ps -f
UID      PID    PPID  C  STIME TTY          TIME CMD
kocmona+ 902     626  0 15:06 tty1        00:00:00 -bash
kocmona+ 915     902  0 15:06 tty1        00:00:00 bash
kocmona+ 923     915 20 15:07 tty1        00:01:51 sh loop
kocmona+ 924     915 32 15:07 tty1        00:02:55 sh loop
kocmona+ 925     915 14 15:07 tty1        00:01:16 sh loop
kocmona+ 926     915 32 15:07 tty1        00:02:54 sh loop
kocmona+ 950     915  0 15:16 tty1        00:00:00 ps -f
kocmonavtik@kocmos:~$ _

```

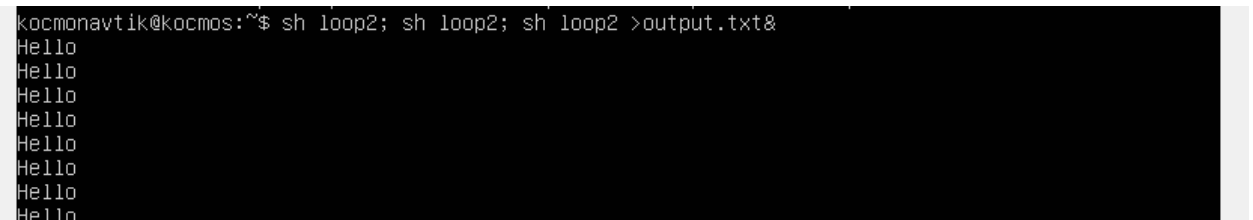
Рисунок 12 – Продолжение работы остановленных процессов

Как можно увидеть, после отправки сигнала о продолжении работы остановленных процессов, возобновляется потребление ресурсов процессора, но уже не в равных долях.

3.2 Часть II

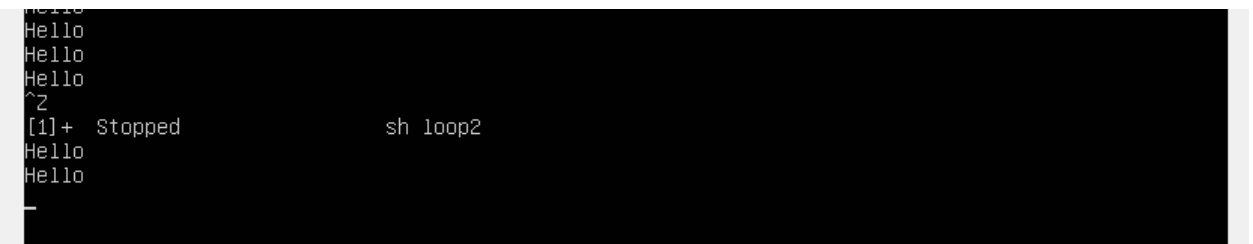
Перед выполнением задания, добавим в сценарий loop2 команду «sleep 5», которая ожидает 5 секунд после выполнения цикла. Добавлено для удобства выполнения задания.

Запустим в консоли выполнение 3-х задач. Две в интерактивном режиме, одну в фоновом.



```
kosmonavtik@kosmos:~$ sh loop2; sh loop2; sh loop2 >output.txt&
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Рисунок 13 – Запуск процессов



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

Рисунок 14 – Остановка интерактивного процесса.

Как видно по рисунку, был запущен один процесс в фоновом режиме, который перенаправляет вывод в текстовый файл. И два интерактивных, которые должны писать вывод в консоль. Но, как видно по следующему рисунку, только после остановки активного интерактивного процесса, запускается второй. Так как запустить 2 или более процессов в интерактивном режиме невозможно, т.к. процесс захватывает терминал в монопольное владение и, пока он не завершится, пользователь не имеет доступа к командной строке на этом терминале.

Остановим второй интерактивный процесс для получения доступа к терминалу.

```
^Z
[2]+  Stopped                  sh loop2
[3] 1034
kosmonavtik@kocmos:~$ _
```

Рисунок 15 – Остановка второго интерактивного процесса


Для перевода задачи из интерактивного режима в фоновый, нужно в первую очередь остановить выполнение команды, для этого нажимаем комбинацию клавиш «Ctrl+Z». После этого, ввести команду «jobs -l» чтобы увидеть все процессы, их номер и PID. И для перевода задачи на задний фон, используем команду «bg номер_задачи».

```
kosmonavtik@kocmos:~$ jobs -l
[1]-  956 Stopped                  sh loop2
[2]+  975 Stopped                  sh loop2
[3]   1034 Running                 sh loop2 > output.txt &
kosmonavtik@kocmos:~$ bg 1
[1]- sh loop2 &
kosmonavtik@kocmos:~$ Hello
Hello
```

Рисунок 16 – Перевод процесса в фоновый режим

Как можно увидеть, после перевода процесса в фоновый режим, он автоматически включается и продолжает работу.

Для перевода процесса из фонового режима в интерактивный, используем команду «fg номер_процесса».



```
Hello
Hello
Hello
Hello
kill -19Hello
 356Hello
956
kocmonavtik@kocmos:~$ fg 1
sh loop2
Hello
```

Рисунок 17 – Перевод процесса в интерактивный режим

После перевода процесса в интерактивный режим, теряется контроль над терминалом пользователя. Для того, чтобы получить контроль, остановим процесс комбинацией клавиш «Ctrl+Z».

Для создания именованного канала используется команда «mkfifo имя_файла». Для передачи списка файлов домашнего каталога вместе с подкаталогами используется команда «ls -R >«Имя_именованного_канала»». Для доступа к именованному каналу всем пользователям, создадим его в каталоге /tmp.

```
kocmonavtik@kocmos:~$ mkfifo /tmp/pipe
kocmonavtik@kocmos:~$ ls /tmp
pipe
snap.lxd
systemd-private-5eda6757eb424614bd33860bcbd3d2be-systemd-logind.service-NA6xMg
systemd-private-5eda6757eb424614bd33860bcbd3d2be-systemd-resolved.service-qgea3i
systemd-private-5eda6757eb424614bd33860bcbd3d2be-systemd-timesyncd.service-yFG3Ih
kocmonavtik@kocmos:~$ ls -R >/tmp/pipe
-
```

Рисунок 18 – Передача списка файлов.

Как можно увидеть, канал находится в ожидании, пока кто ни будь подключится к нему и примет данные.

Для этого нажмем комбинацию клавиш «Ctrl+alt+F2» для открытия новой виртуальной консоли, авторизируемся и принимаем данные командой «cat /tmp/pipe».

```
user1@kocmos:~$ cat /tmp/pipe
.:
1.txt
info.txt
loop
loop2
new
output.txt
output1.txt
output2.txt

./new:
new2
test1.txt
test2.txt

./new/new2:
user1@kocmos:~$
```

Рисунок 19 – Получение данных с канала.

После получения данных, ожидание в первой виртуальной консоли прекращается.

Для передачи данных, воспользуемся архиватором tar и напишем команду для передачи файлов в канал «tar -cvf new.tar new>/tmp/pipe». Общий вид команды архиватора tar «tar опции название_архива файлы_для_архивации».

```
kocmonavtik@kocmos:~$ ls
1.txt info.txt loop loop2 new new2 output.txt output1.txt output2.txt
kocmonavtik@kocmos:~$ ls /tmp
pipe
snap.lxd
systemd-private-ec820ec8eadc458cb5a0e0bb17d32561-systemd-logind.service-TtJR2g
systemd-private-ec820ec8eadc458cb5a0e0bb17d32561-systemd-resolved.service-8b809i
systemd-private-ec820ec8eadc458cb5a0e0bb17d32561-systemd-timesyncd.service-83R4ri
kocmonavtik@kocmos:~$ tar -cvf new.tar new >/tmp/pipe
```

Рисунок 20 – Передача файлов в канал

Канал снова находится в ожидании, пока кто ни будь подключится к каналу и получит файлы.

```
user1@kocmos:~$ ls
1.txt 2LinkH.txt 2LinkS.txt 3.txt filesTxt.tar.gz new txt.tar
user1@kocmos:~$ cat /tmp/pipe
new/
new/test2.txt
new/new2/
new/test1.txt
user1@kocmos:~$ _
```

Рисунок 21 – Получение данных

3.3 Часть III

Для вывода информации о состоянии процессов системы в реальном времени используется команда «top». Для вывода информации с сортировкой по убыванию значения приоритета, используется команда «top -o PR».

```
top - 08:48:17 up 39 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 94 total, 1 running, 93 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7962.4 total, 7474.2 free, 140.2 used, 347.9 buff/cache
MiB Swap: 4096.0 total, 4096.0 free, 0.0 used, 7581.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
23	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
1	root	20	0	101996	11636	8572	S	0.0	0.1	0:01.51	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
10	root	20	0	0	0	0	S	0.0	0.0	0:00.11	ksoftirqd/0
11	root	20	0	0	0	0	I	0.0	0.0	0:00.67	rcu_sched
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
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
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
81	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kswapd0
82	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ecryptfs-kthrea
86	root	20	0	0	0	0	S	0.0	0.0	0:00.01	scsi_eh_0
88	root	20	0	0	0	0	S	0.0	0.0	0:00.00	scsi_eh_1
161	root	20	0	0	0	0	S	0.0	0.0	0:00.02	scsi_eh_2
163	root	20	0	0	0	0	I	0.0	0.0	0:01.17	kworker/0:2-events
264	root	20	0	0	0	0	S	0.0	0.0	0:00.02	jbd2/dm-0-8
364	root	20	0	22016	6176	4044	S	0.0	0.1	0:01.21	systemd-udev
527	root	20	0	0	0	0	S	0.0	0.0	0:00.00	jbd2/sda2-8
540	systemd+	20	0	90388	6292	5420	S	0.0	0.1	0:00.12	systemd-timesyn
582	systemd+	20	0	26740	7796	6840	S	0.0	0.1	0:00.10	systemd-network
601	systemd+	20	0	24044	12200	8200	S	0.0	0.1	0:00.12	systemd-resolve
615	root	20	0	238060	9328	8296	S	0.0	0.1	0:00.09	accounts-daemon
618	root	20	0	5568	2892	2676	S	0.0	0.0	0:00.00	cron
619	message+	20	0	7444	4632	3936	S	0.0	0.1	0:00.07	dbus-daemon
627	root	20	0	26284	17824	10192	S	0.0	0.2	0:00.09	networkd-dispat
629	syslog	20	0	224324	4840	3804	S	0.0	0.1	0:00.01	rsyslogd

Рисунок 22 – Выведение информации о состоянии процессов.

Для наглядности, нажали клавишу «x» для выделения характеристики процесса, по которой происходит сортировка.

Так же имеется альтернативный способ сортировки процессов. После введения команды «top», с помощью комбинации клавиш «Shift+>» и «Shift+<» для выбора соседнего поля, по которому будет происходить сортировка.

Перед удалением всех дочерних процессов текущего командного интерпретатора, упростим просмотр этих процессов. Для этого выберем сортировку по пользователю, и нажмем комбинацию клавиш «Shift+V» для отображения процессов в виде дерева.

```
top - 09:32:51 up 9 min, 1 user, load average: 0.00, 0.09, 0.10
Tasks: 97 total, 1 running, 96 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 88.3 id, 11.7 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7962.4 total, 7521.5 free, 138.2 used, 302.7 buff/cache
MiB Swap: 4096.0 total, 4096.0 free, 0.0 used, 7586.7 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	101988	11372	8288	S	0.0	0.1	0:01.50	systemd
333	root	19	-1	68036	14908	13824	S	0.0	0.2	0:00.25	- systemd-journal
363	root	20	0	21884	6024	4068	S	0.0	0.1	0:00.67	- systemd-udevd
505	root	rt	0	280288	18096	8184	S	0.0	0.2	0:00.09	- multipathd
538	systemd+	20	0	90388	6388	5512	S	0.0	0.1	0:00.11	- systemd-timesyn
581	systemd+	20	0	26740	7732	6772	S	0.0	0.1	0:00.12	- systemd-network
599	systemd+	20	0	24044	12212	8208	S	0.0	0.1	0:00.11	- systemd-resolve
614	root	20	0	238060	9440	8404	S	0.0	0.1	0:00.04	- accounts-daemon
617	root	20	0	5568	2908	2696	S	0.0	0.0	0:00.00	- cron
618	message+	20	0	7444	4588	3892	S	0.0	0.1	0:00.06	- dbus-daemon
627	root	20	0	26284	17944	10304	S	0.0	0.2	0:00.09	- networkd-dispat
628	syslog	20	0	224324	4780	3748	S	0.0	0.1	0:00.01	- rsyslogd
630	root	20	0	627204	26208	14644	S	0.0	0.3	0:00.97	- snapd
636	root	20	0	16804	7856	6880	S	0.0	0.1	0:00.09	- systemd-logind
639	daemon	20	0	3792	2188	2008	S	0.0	0.0	0:00.00	- atd
645	root	20	0	6112	4108	3292	S	0.0	0.1	0:00.05	- login
927	kocmona+	20	0	7068	5112	3412	S	0.0	0.1	0:00.05	- bash
944	kocmona+	20	0	7036	4920	3288	S	0.0	0.1	0:00.04	- bash
957	kocmona+	20	0	2608	1664	1568	S	0.0	0.0	0:00.01	- sh
1054	kocmona+	20	0	4260	580	516	S	0.0	0.0	0:00.00	- sleep
960	kocmona+	20	0	2608	1636	1540	S	0.0	0.0	0:00.00	- sh
1055	kocmona+	20	0	4260	592	528	S	0.0	0.0	0:00.00	- sleep
964	kocmona+	20	0	7916	3632	3104	R	0.0	0.0	0:00.88	- top
664	root	20	0	105100	20524	12856	S	0.0	0.3	0:00.08	- unattended-upgr
667	root	20	0	12160	6828	5988	S	0.0	0.1	0:00.00	- sshd
687	root	20	0	236292	8960	8044	S	0.0	0.1	0:00.02	- polkitd
917	kocmona+	20	0	18556	9904	8228	S	0.0	0.1	0:00.10	- systemd
922	kocmona+	20	0	103200	3452	12	S	0.0	0.0	0:00.00	- (sd-pam)
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	- rcu_gp

Рисунок 23 – Отображение процессов в виде дерева.

Для удаления процесса, используется клавиша «k» после которой нужно ввести PID процесса и после этого, сигнал SIGKILL (9).

```
top - 09:40:12 up 17 min, 1 user, load average: 0.01, 0.03, 0.06
Tasks: 97 total, 1 running, 96 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7962.4 total, 7520.6 free, 138.6 used, 303.2 buff/cache
MiB Swap: 4096.0 total, 4096.0 free, 0.0 used, 7586.2 avail Mem
PID to signal/kill [default pid = 1] 957
```

Рисунок 24 – Ввод PID процесса

После нажатия клавиши «Enter», нужно ввести сигнал. В случае пропуска данного пункта, автоматически отправляется сигнал SIGTERM (15)

```
top - 09:36:34 up 13 min, 1 user, load average: 0.01, 0.05, 0.08
Tasks: 98 total, 2 running, 96 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.6 id, 0.3 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7962.4 total, 7523.2 free, 136.5 used, 302.7 buff/cache
MiB Swap: 4096.0 total, 4096.0 free, 0.0 used, 7588.4 avail Mem
Send pid 1144 signal [15/sigterm] 9_
```

Рисунок 25 – Ввод сигнала для процесса

После ввода SIGKILL (9), процесс удаляется.

В случае удаления процесса «top», прекращается мониторинг состояния процессов.

Для вывода статистики использования памяти используется команда «free». При вводе команды без ключей, она отобразит статистику в килобайтах и только 1 раз. Для вывода информации в байтах и каждые 3 секунды, используется команда «free -b -s 3».

```

kocmonavtik@kocmos:~$ free -b -s 3

```

	total	used	free	shared	buff/cache	available
Mem:	8349188096	144699392	7880384512	1052672	324104192	7954898944
Swap:	4294963200	0	4294963200			

	total	used	free	shared	buff/cache	available
Mem:	8349188096	144707584	7880376320	1052672	324104192	7954890752
Swap:	4294963200	0	4294963200			

	total	used	free	shared	buff/cache	available
Mem:	8349188096	144707584	7880376320	1052672	324104192	7954890752
Swap:	4294963200	0	4294963200			

	total	used	free	shared	buff/cache	available
Mem:	8349188096	144707584	7880376320	1052672	324104192	7954890752
Swap:	4294963200	0	4294963200			

	total	used	free	shared	buff/cache	available
Mem:	8349188096	144707584	7880376320	1052672	324104192	7954890752
Swap:	4294963200	0	4294963200			

	total	used	free	shared	buff/cache	available
Mem:	8349188096	144707584	7880376320	1052672	324104192	7954890752
Swap:	4294963200	0	4294963200			

Рисунок 26 – Статистика используемой и свободной памяти.

Где «Mem» – физическая память, «Swap» виртуальная память (Paging).

total — общее количество памяти;

used — реально используемая в данный момент и зарезервированная системой память;

free — свободная память ;

shared — Shared memory или *Разделяемая память*.

buffers — буферы в памяти — страницы памяти, зарезервированные системой для выделения их процессам, когда они затребуют этого.

cached — файлы, которые недавно были использованы

системой/процессами и хранящиеся в памяти на случай если вскоре они снова потребуются.

Для остановки процесса, нажать комбинацию клавиш «Ctrl+z».

4. Вывод

В ходе выполнения лабораторной работы ознакомились с процессами в операционной системе. Приобрели опыт и навыки управления процессами в операционной системе Linux.