Отчёт по лабораторной работе №6

Управление процессами

Руслан Алиев

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Выполнение

2.1 Управление заданиями

1. Для получения административных прав выполнен переход в режим супер-пользователя с помощью команды
   su -

2. Далее были запущены несколько процессов:
   sleep 3600 &
   dd if=/dev/zero of=/dev/null &
   sleep 7200
   Первые две команды запустились в фоновом режиме, а третья — в обычном, из-за отсутствия символа **&**.

```
raliev@raliev:~$ su
Password:
root@raliev:/home/raliev# sleep 3600 &
[1] 3447
root@raliev:/home/raliev# dd if=/dev/zero of=/dev/null &
[2] 3476
root@raliev:/home/raliev# sleep 7200
^C
root@raliev:/home/raliev# jobs
[1]-  Running                 sleep 3600 &
[2]+  Running                 dd if=/dev/zero of=/dev/null &
root@raliev:/home/raliev# bg 3
bash: bg: 3: no such job
root@raliev:/home/raliev# sleep 7200
^Z
[3]+  Stopped                 sleep 7200
root@raliev:/home/raliev# jobs
[1]   Running                 sleep 3600 &
[2]-  Running                 dd if=/dev/zero of=/dev/null &
[3]+  Stopped                 sleep 7200
root@raliev:/home/raliev# bg 3
[3]+ sleep 7200 &
root@raliev:/home/raliev# fg 1
sleep 3600
^C
root@raliev:/home/raliev#
```

Рис. 2.1: Запуск заданий

3. Так как процесс *sleep 7200* занял терминал, он был остановлен комбинацией **Ctrl + Z**.

   Проверка активных заданий выполнялась командой **jobs**.

   В результате отобразились три задания: два в состоянии *Running* и одно в состоянии *Stopped*.

4. Для возобновления выполнения остановленного задания №3 в фоновом режиме использовалась команда

   bg 3

   После этого команда **jobs** показала обновлённые статусы процессов.

5. Для возврата задания №1 на передний план была применена команда

   fg 1

   Затем процесс был прерван комбинацией **Ctrl + C**.

6. Аналогичным образом были завершены задания №2 и №3.

7. В отдельном терминале, уже под своей учётной записью пользователя, был запущен процесс

   dd if=/dev/zero of=/dev/null &

   После закрытия терминала командой **exit** процесс продолжал выполняться в системе.

8. Для мониторинга процессов была использована команда **top**.

   В списке можно увидеть активное задание **dd**, которое занимает практически 100% процессорного времени.

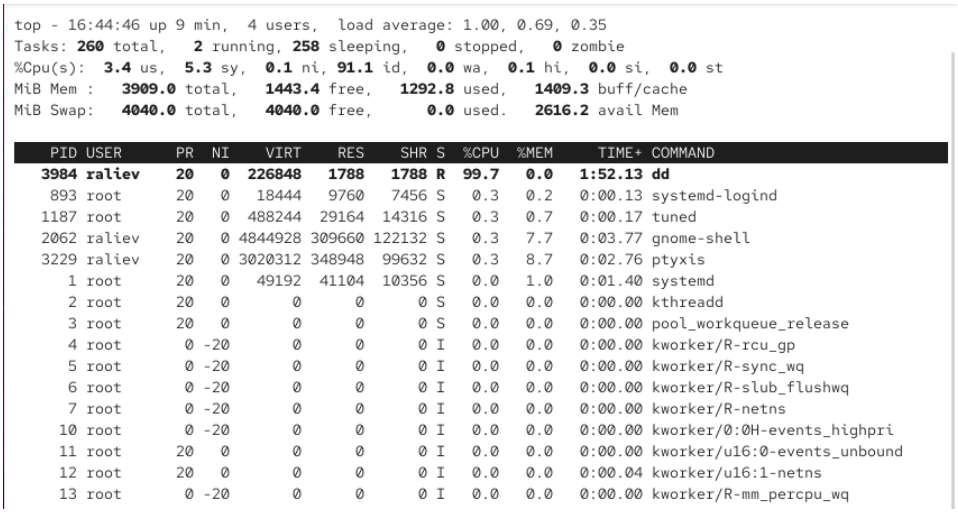

Рис. 2.2: Работа команды dd в top

9. Повторный запуск **top** позволил завершить процесс с помощью клавиши **k**, после чего программа была закрыта клавишей **q**.

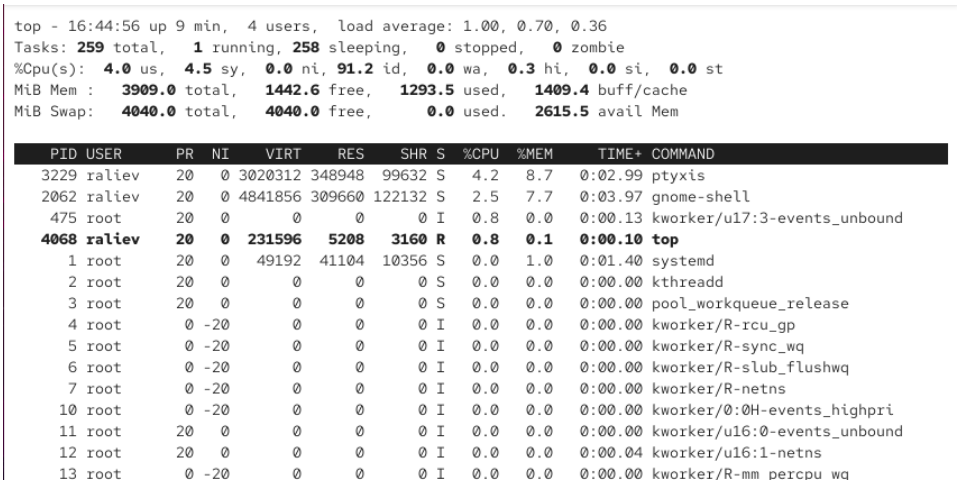

Рис. 2.3: Завершение процесса dd в top

2.2 Управление процессами

1. Для получения административных прав выполнен переход в режим супер-пользователя с помощью команды

   su -

2. Запущено несколько процессов с помощью команды

   dd if=/dev/zero of=/dev/null &

   Команда была введена трижды, что запустило три отдельных процесса dd в фоновом режиме.

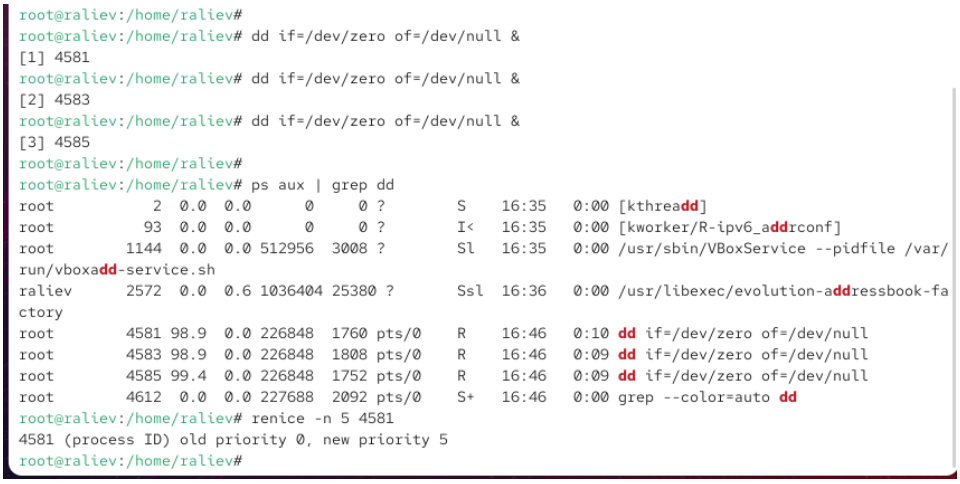

Рис. 2.4: Запуск процессов dd

3. Для просмотра списка запущенных процессов использовалась команда **ps aux | grep dd**

   В выводе отобразились все процессы, содержащие в имени *dd*, включая три активных задания.

4. При помощи PID одного из процессов был изменён его приоритет: **renice -n 5**

   После выполнения команды отображается информация о старом и новом приоритете.

5. Для просмотра иерархии процессов была применена команда **ps fax | grep -B5 dd**

   В выводе показаны процессы dd вместе с их родительской оболочкой.

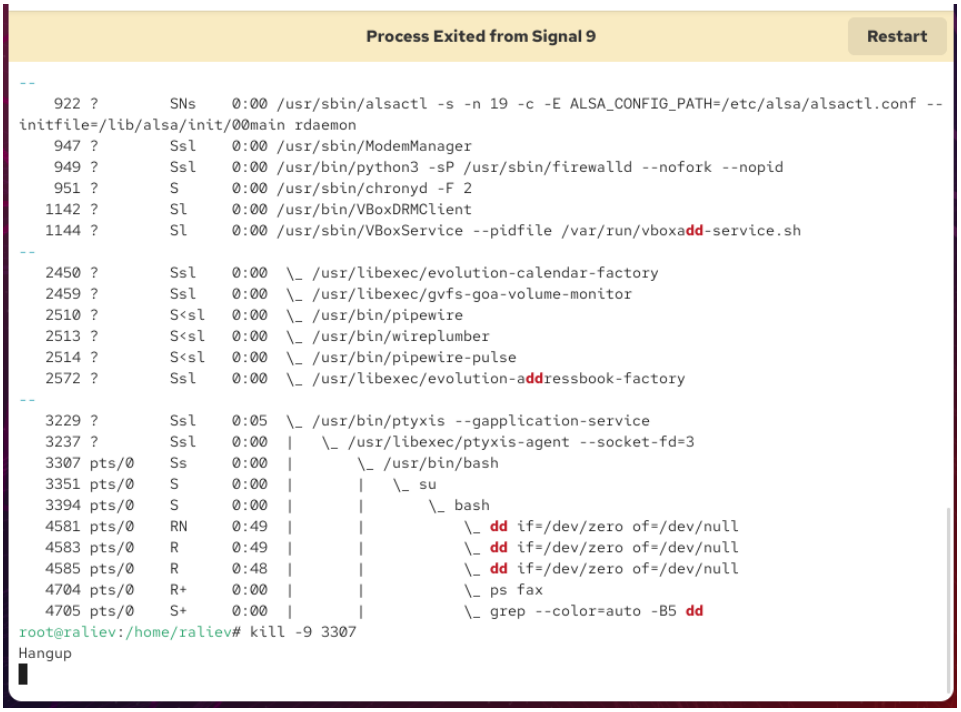

Рис. 2.5: Просмотр иерархии процессов

6. Использовав PID корневой оболочки, из которой были запущены процессы dd, была выполнена команда

   kill -9

   В результате оболочка завершилась, а вместе с ней автоматически остановились и все дочерние процессы dd.

2.3 Задание 1

1. Сначала был выполнен переход в режим суперпользователя с помощью команды

   su -

2. Трижды была запущена команда

   dd if=/dev/zero of=/dev/null &

   В результате созданы три фоновых процесса dd, каждый из которых выпол-

няет бесконечную запись в устройство `/dev/null`.

```
raliev@raliev:~$
raliev@raliev:~$ su
Password:
root@raliev:/home/raliev# dd if=/dev/zero of=/dev/null &
[1] 4982
root@raliev:/home/raliev# dd if=/dev/zero of=/dev/null &
[2] 4984
root@raliev:/home/raliev# dd if=/dev/zero of=/dev/null &
[3] 4986
root@raliev:/home/raliev# renice -n 5 4982
4982 (process ID) old priority 0, new priority 5
root@raliev:/home/raliev# renice -n 15 4982
4982 (process ID) old priority 5, new priority 15
root@raliev:/home/raliev# killall dd
[1]   Terminated              dd if=/dev/zero of=/dev/null
[2]-  Terminated              dd if=/dev/zero of=/dev/null
[3]+  Terminated              dd if=/dev/zero of=/dev/null
root@raliev:/home/raliev#
```

Рис. 2.6: Запуск процессов dd

3. Для одного из процессов (PID 4982) приоритет был изменён с 0 на **5** командой

   renice -n 5 4982

   После этого тот же процесс получил новый приоритет **15** с помощью команды

   renice -n 15 4982

   Разница заключается в том, что большее положительное значение приоритета означает меньший приоритет выполнения (процесс будет получать меньше процессорного времени по сравнению с другими).

4. Все запущенные процессы dd были завершены одной командой:

   killall dd

   В результате три фоновых задания перешли в состояние *Terminated*.

2.4 Задание 2

1. Сначала была запущена программа **yes** в фоновом режиме с подавлением потока вывода:
   yes > /dev/null &

2. Далее программа **yes** была запущена на переднем плане с теми же параметрами.
   Она была приостановлена комбинацией **Ctrl + Z**, а затем возобновлена и завершена.

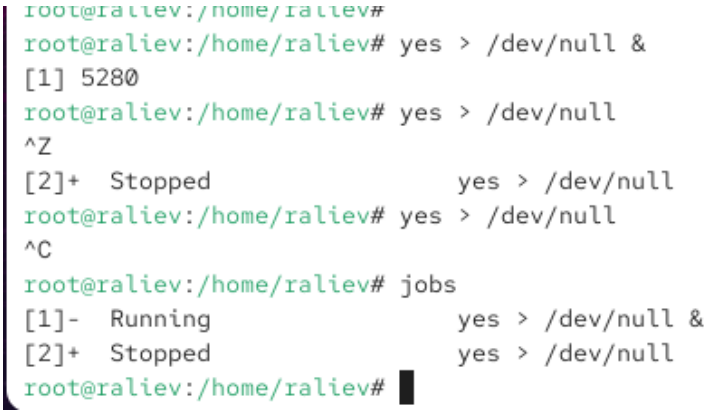

Рис. 2.7: Работа с процессами yes

3. Программа **yes** была также запущена без подавления вывода. После остановки и возобновления выполнение было завершено.

4. Проверка состояний заданий выполнялась командой **jobs**, где видно различие между состояниями *Running* и *Stopped*.

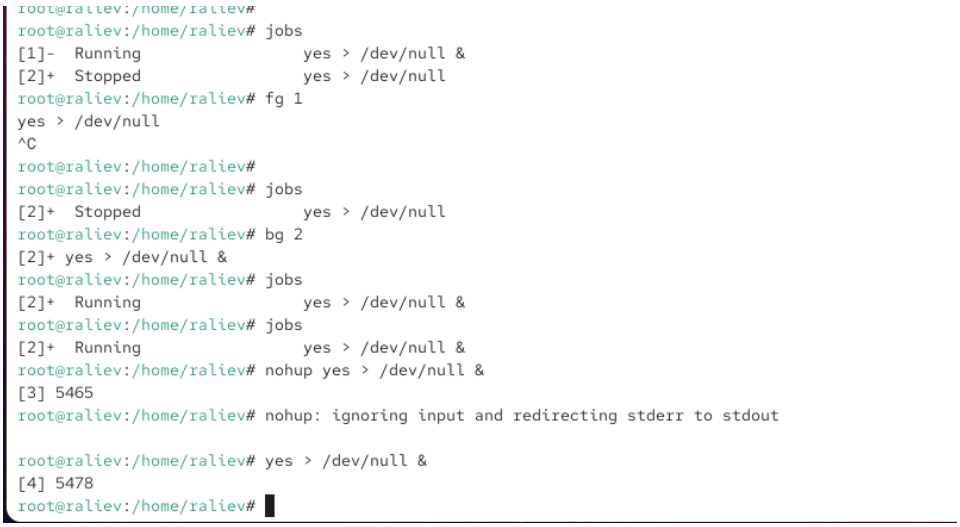

Рис. 2.8: Проверка статуса заданий

5. Один из процессов был переведён на передний план командой **fg** и остановлен.

6. Другой процесс был возвращён в фоновый режим с помощью **bg**, после чего в списке заданий он отобразился как *Running*.

7. Для запуска процесса, сохраняющего выполнение после закрытия терминала, применена команда:
   nohup yes > /dev/null &

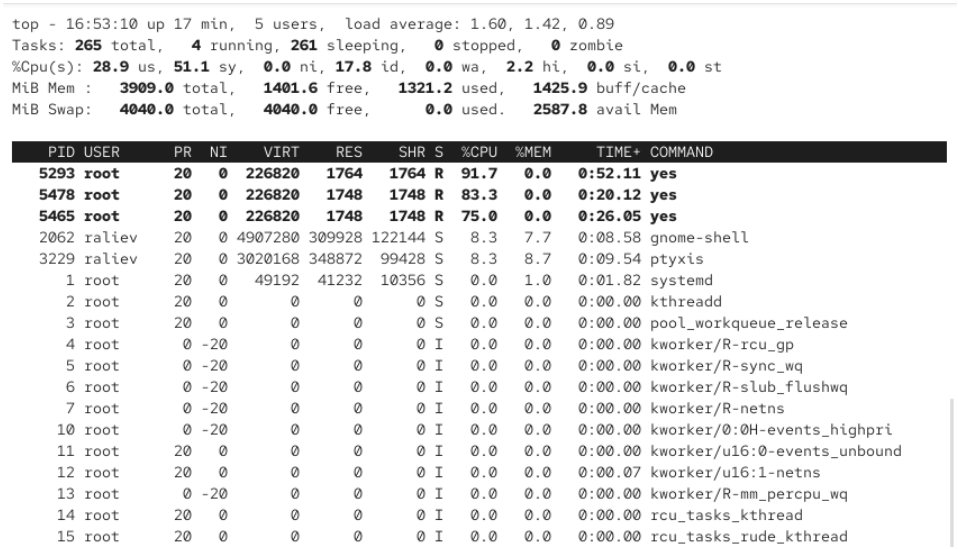

Рис. 2.9: Запуск yes с nohup

8. Проверка с помощью **top** подтвердила, что процессы **yes** продолжают выполняться, нагружая процессор.

```
root@raliev:/home/raliev# yes > /dev/null &
[1] 5642
root@raliev:/home/raliev# yes > /dev/null &
[2] 5648
root@raliev:/home/raliev# yes > /dev/null &
[3] 5652
root@raliev:/home/raliev# kill 5652
[3]+  Terminated              yes > /dev/null
root@raliev:/home/raliev# fg 2
yes > /dev/null
^C
root@raliev:/home/raliev# kill -1 5642
[1]+  Hangup                  yes > /dev/null
root@raliev:/home/raliev# kill -1 5293
root@raliev:/home/raliev# kill -1 5478
root@raliev:/home/raliev# kill -1 5465
root@raliev:/home/raliev# yes > /dev/null &
[1] 5844
root@raliev:/home/raliev# yes > /dev/null &
[2] 5846
root@raliev:/home/raliev# yes > /dev/null &
[3] 5848
root@raliev:/home/raliev# killall yes
[1]   Terminated              yes > /dev/null
[2]-  Terminated              yes > /dev/null
[3]+  Terminated              yes > /dev/null
root@raliev:/home/raliev#
```

Рис. 2.10: Работа yes в top

9. В фоновом режиме были запущены ещё три процесса **yes**. Два из них были завершены разными способами: один по PID, другой по номеру задания.

10. Для проверки работы сигналов был отправлен сигнал **SIGHUP (1)** обычному процессу и процессу, запущенному с **nohup**. Первый процесс завершился, второй продолжил выполнение.

11. Вновь было запущено несколько процессов **yes**. Все они были завершены одновременно командой **killall yes**.

12. Для запуска процесса с пониженным приоритетом была использована команда:
    nice -n 5 yes > /dev/null &

Второй процесс был запущен с обычным приоритетом для сравнения. Проверка с помощью **ps -l** показала разницу в значениях поля NI (nice).

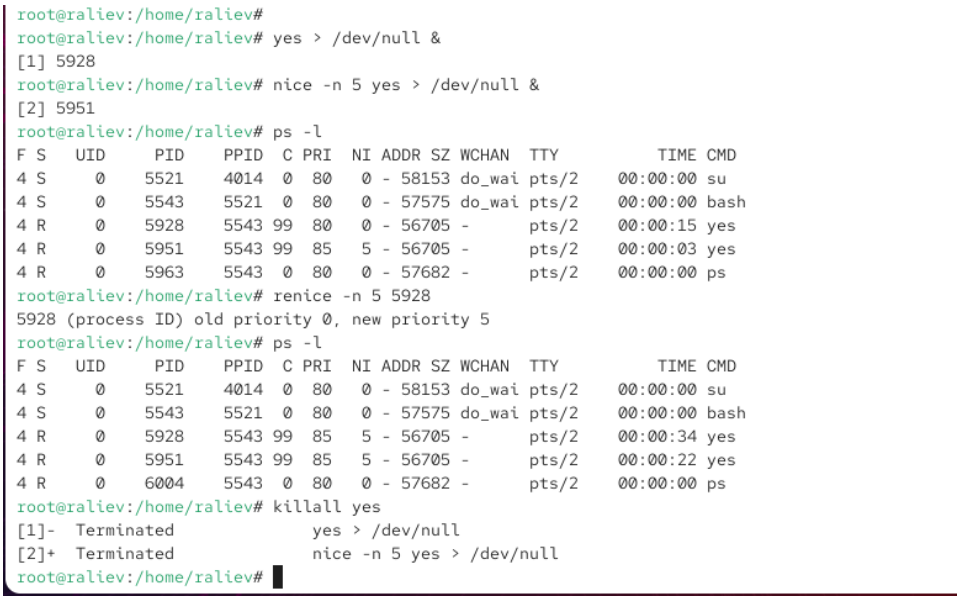

Рис. 2.11: Сравнение приоритетов процессов yes

13. С помощью утилиты **renice** приоритет одного из процессов был изменён до **5**, что подтвердилось выводом команды **ps -l**.

3 Заключение

В ходе работы были изучены основные способы управления заданиями и процессами в Linux.
Были рассмотрены приёмы запуска процессов на переднем и фоновом режиме, их приостановка, возобновление и завершение.
Особое внимание уделялось использованию команд **jobs**, **fg**, **bg**, **kill**, а также инструментов **nice** и **renice** для управления приоритетами.
Дополнительно была изучена работа с утилитой **top** для мониторинга процессов и их завершения.

4 Контрольные вопросы

1. **Какая команда даёт обзор всех текущих заданий оболочки?**
   jobs

2. **Как остановить текущее задание оболочки, чтобы продолжить его выполнение в фоновом режиме?**
   Нажать **Ctrl + Z**, затем выполнить команду **bg %номер_задания**

3. **Какую комбинацию клавиш можно использовать для отмены текущего задания оболочки?**
   Ctrl + C

4. **Необходимо отменить одно из начатых заданий. Доступ к оболочке, в которой в данный момент работает пользователь, невозможен. Что можно сделать, чтобы отменить задание?**
   Использовать команду **kill** или **killall** в другой оболочке

5. **Какая команда используется для отображения отношений между родительскими и дочерними процессами?**
   ps fax

6. **Какая команда позволит изменить приоритет процесса с идентификатором 1234 на более высокий?**
   renice -n -5 -p 1234

7. **В системе в настоящее время запущено 20 процессов dd. Как проще всего остановить их все сразу?**

killall dd

8. **Какая команда позволяет остановить команду с именем mycommand?**
   killall mycommand

9. **Какая команда используется в top, чтобы убить процесс?**
   Внутри top нажать клавишу **k**, ввести PID процесса и подтвердить

10. **Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов?**
    Использовать **nice** с положительным значением (например, nice -n 10 mycommand)