Отчёт по лабораторной работе №6

Управление процессами

Турсунов Мухамметназар

Содержание

# 1 Цель работы

Получить навыки управления процессами операционной системы.

# 2 Выполнение

## 2.1 Управление заданиями

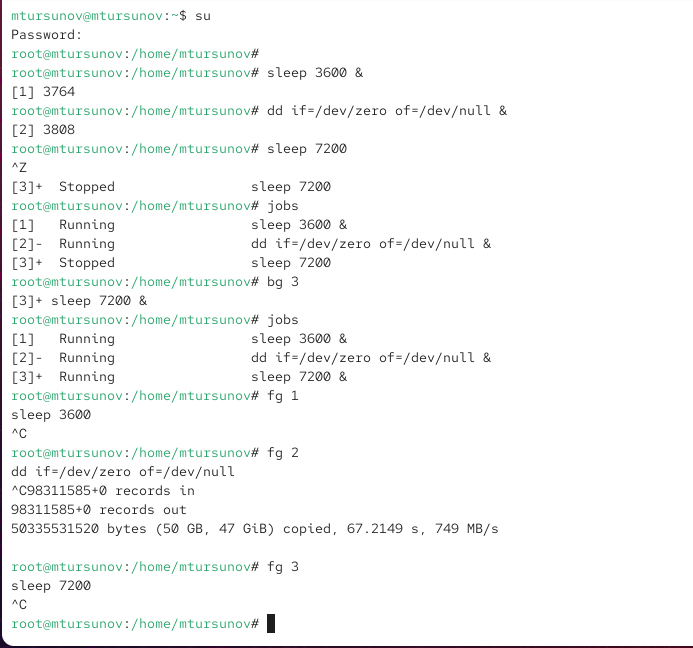
1. Для начала были получены права суперпользователя с помощью команды **su -**.  
   После ввода пароля приглашение оболочки изменилось на root@mtursunov, что подтверждает успешный переход в режим администратора.
2. Запущены три задания:
   * **sleep 3600 &** — команда переведена в фоновый режим, где процесс «спит» один час;
   * **dd if=/dev/zero of=/dev/null &** — выполняется запись нулевых байт в «чёрную дыру» /dev/null;
   * **sleep 7200** — процесс запущен в обычном (переднем) режиме, поэтому управление терминалом временно заблокировано.

* После нажатия **Ctrl + Z** задание sleep 7200 было приостановлено.

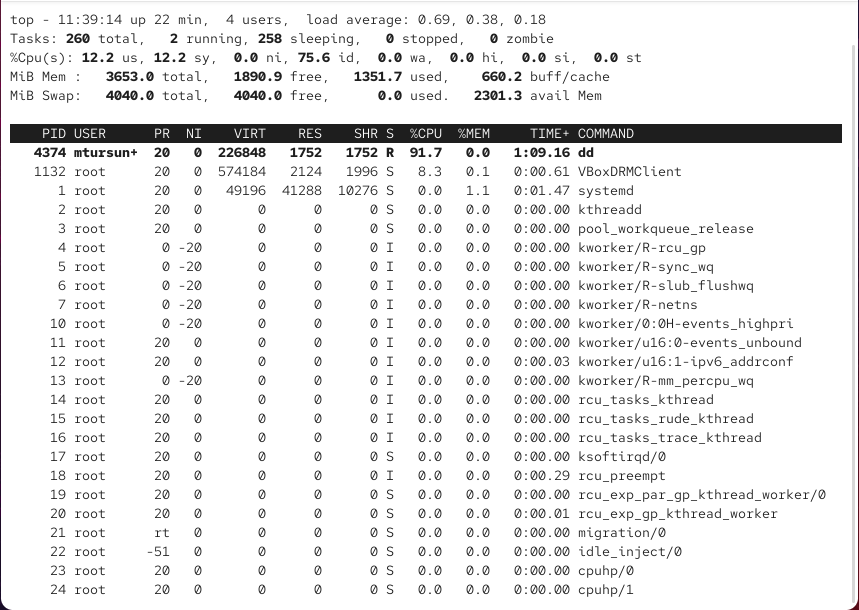
1. С помощью команды **jobs** выведен список текущих заданий:
   * [1] sleep 3600 — *Running*
   * [2] dd if=/dev/zero of=/dev/null — *Running*
   * [3] sleep 7200 — *Stopped*

* Для перевода третьего задания обратно в фоновый режим использована команда **bg 3**, после чего все три задания выполнялись в фоне.

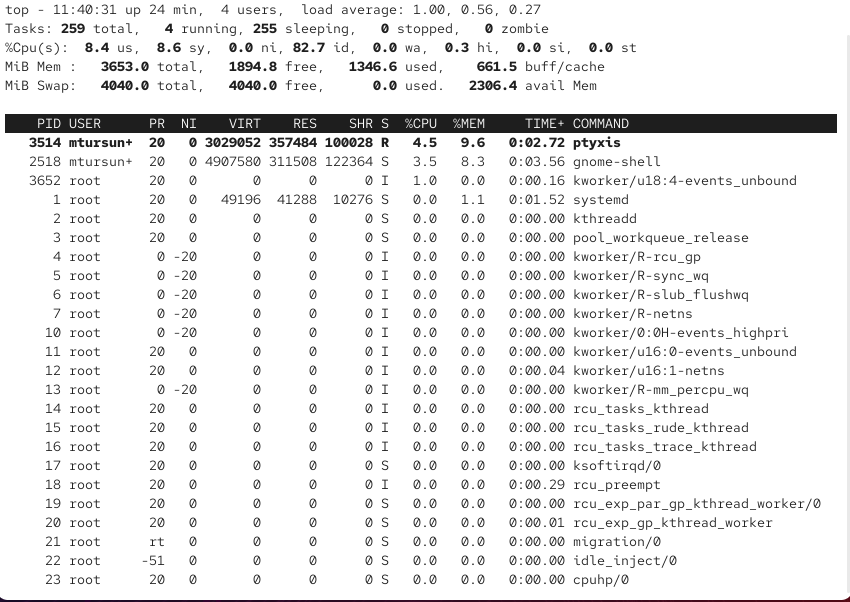
1. Команда **fg 1** вернула первое задание на передний план. Оно было остановлено сочетанием **Ctrl + C**, что подтвердилось выводом команды **jobs** — процесс 1 исчез из списка активных.  
   Аналогично были завершены задания 2 и 3 с помощью **fg 2** и **fg 3** соответственно.

* 
* Рис. 1: Работа с заданиями, управление приоритетами и завершение процессов

1. В другом терминале, уже под обычной учётной записью пользователя, запущено задание:  
   **dd if=/dev/zero of=/dev/null &**  
   После закрытия терминала с помощью команды **exit** процесс продолжил выполняться в фоне.
2. Для проверки текущей загрузки системы была запущена команда **top**.  
   В таблице процессов видно, что команда dd активно использует процессор (около 91,7% CPU), а в списке задач отображается в состоянии **R (Running)**.

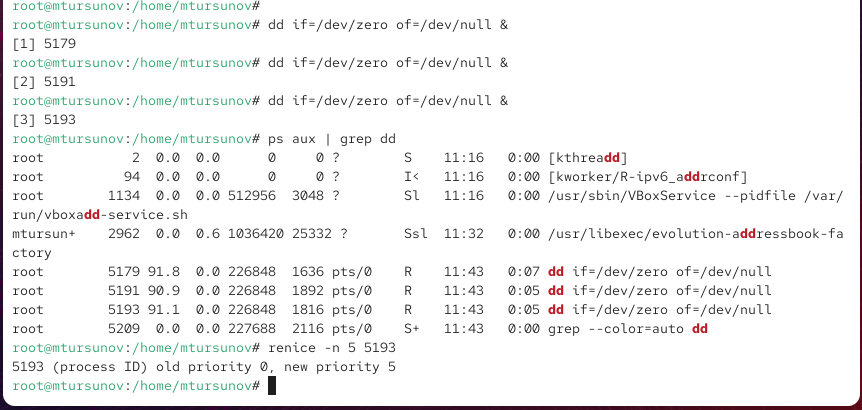
* 
* Рис. 2: Команда top с отображением активного процесса dd

1. После выхода из программы **top** и повторного её запуска было выполнено завершение процесса с помощью клавиши **k** и указания PID соответствующего процесса dd.  
   После завершения процесса система освободила ресурсы, что отразилось на снижении нагрузки процессора.

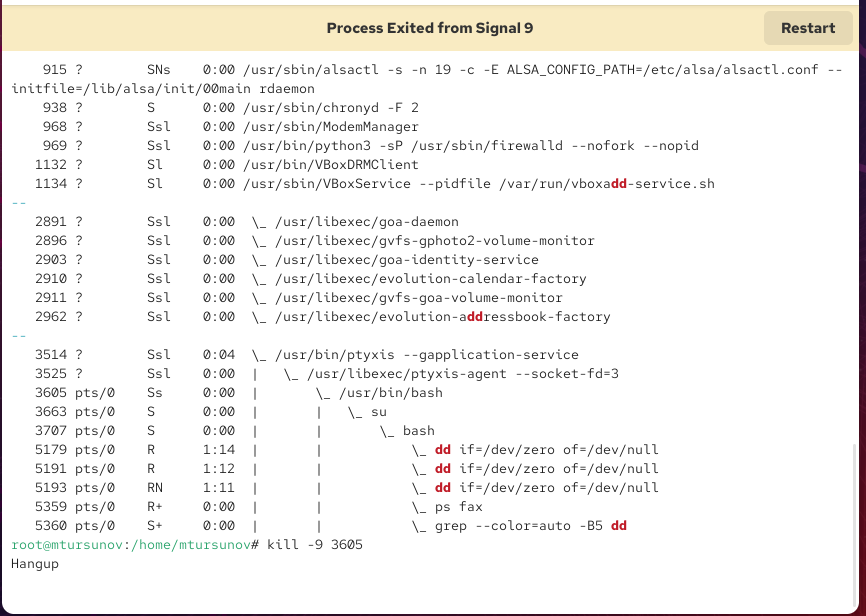
* 
* Рис. 3: Завершение процесса dd в top

## 2.2 Управление процессами

1. Для начала были получены права суперпользователя с помощью команды **su -**.  
   После успешной аутентификации приглашение оболочки изменилось на root@mtursunov, что подтверждает переход в режим администратора.
2. Далее были запущены три одинаковых процесса:
   * **dd if=/dev/zero of=/dev/null &**
   * **dd if=/dev/zero of=/dev/null &**
   * **dd if=/dev/zero of=/dev/null &**  
     Каждый из них выполняет запись бесконечного потока нулей в «чёрную дыру» /dev/null, полностью загружая процессор.

* 
* Рис. 4: Запуск трёх фоновых процессов dd

1. Для просмотра списка активных процессов использовалась команда **ps aux | grep dd**.  
   В результате отображены все процессы, содержащие в названии dd. В нижней части списка видны три активных процесса dd, каждый из которых имеет собственный PID.
2. Для изменения приоритета одного из процессов применена команда **renice -n 5 5193**, где 5193 — идентификатор выбранного процесса.  
   В ответ система вывела сообщение:  
   5193 (process ID) old priority 0, new priority 5  
   Это означает, что приоритет процесса был понижен, и теперь он будет получать меньше процессорного времени по сравнению с другими задачами.
3. Для анализа структуры процессов выполнена команда **ps fax | grep -B5 dd**.  
   Ключ -B5 позволил вывести пять строк перед найденными записями, благодаря чему видна иерархия процессов и родительская оболочка, из которой были запущены процессы dd.

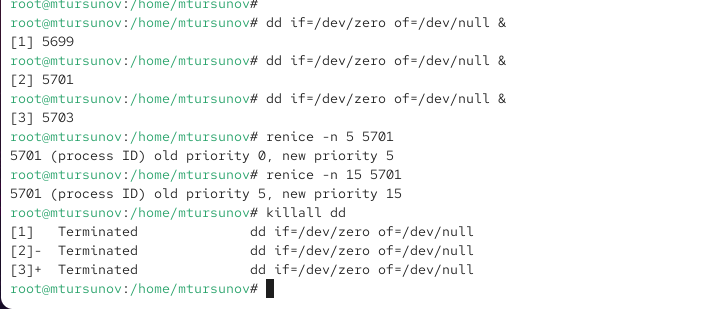
* 
* Рис. 5: Просмотр иерархии процессов и изменение приоритета dd

1. После определения PID родительской оболочки был выполнен её принудительный останов с помощью команды **kill -9 3605**.  
   Это действие завершило не только саму оболочку, но и все дочерние процессы dd, запущенные из неё. Такой способ является удобным решением для массового завершения связанных процессов.

* После выполнения команды в терминале появилось сообщение **Hangup**, что подтверждает завершение сеанса и всех зависимых задач.

## 2.3 Задание 1. Управление приоритетами процессов

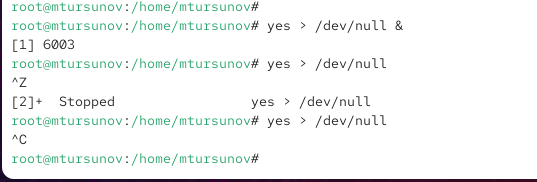
1. Для начала было трижды запущено однотипное задание **dd if=/dev/zero of=/dev/null &**, выполняющее запись потока нулей в устройство /dev/null.  
   Каждый процесс выполнялся в фоновом режиме и получил собственный PID.

* 
* Рис. 6: Запуск трёх фоновых процессов dd

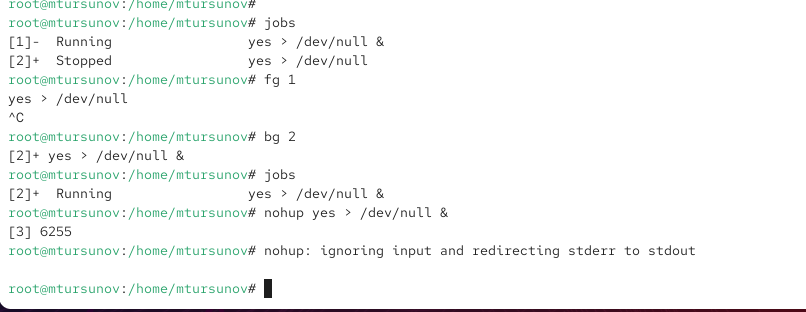
1. Далее для одного из процессов с PID 5701 был изменён приоритет командой **renice -n 5 5701**.  
   В результате система сообщила:  
   5701 (process ID) old priority 0, new priority 5  
   Это означает, что приоритет был **понижен**, и процесс стал менее приоритетным при распределении процессорного времени.
2. Затем приоритет того же процесса был изменён повторно — **renice -n 15 5701**.  
   Теперь система вывела:  
   5701 (process ID) old priority 5, new priority 15  
   Приоритет увеличился ещё сильнее (численно большее значение означает **меньший приоритет**).  
   Таким образом, процесс с nice = 15 будет получать значительно меньше процессорного времени по сравнению с другими.
3. Для завершения всех процессов dd использовалась команда **killall dd**.  
   Система уведомила о завершении трёх процессов, выводя сообщения *Terminated* для каждого из них.  
   Это подтверждает, что все фоновые задания были успешно остановлены.

## 2.4 Задание 2. Управление заданиями и приоритетами процессов

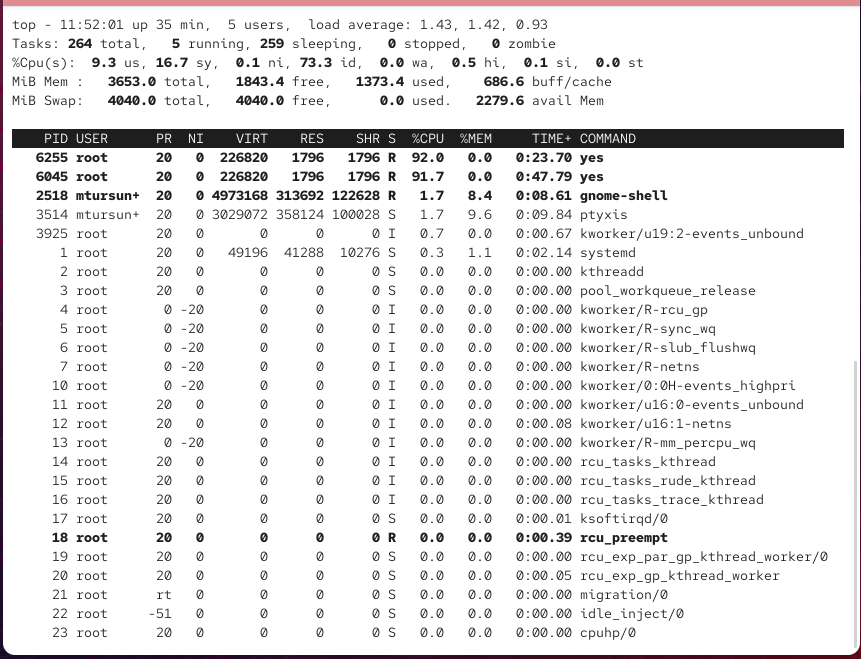
1. Сначала была запущена программа **yes > /dev/null &**, выполняющая бесконечный вывод строки «y» с перенаправлением потока вывода в /dev/null.  
   Программа была помещена в фоновый режим и получила PID 6003.
2. Затем та же команда была выполнена на переднем плане. После приостановки процесса сочетанием **Ctrl + Z** программа отобразилась в состоянии *Stopped*.  
   Повторный запуск и завершение через **Ctrl + C** остановили выполнение.

* 
* Рис. 7: Запуск и управление процессами yes

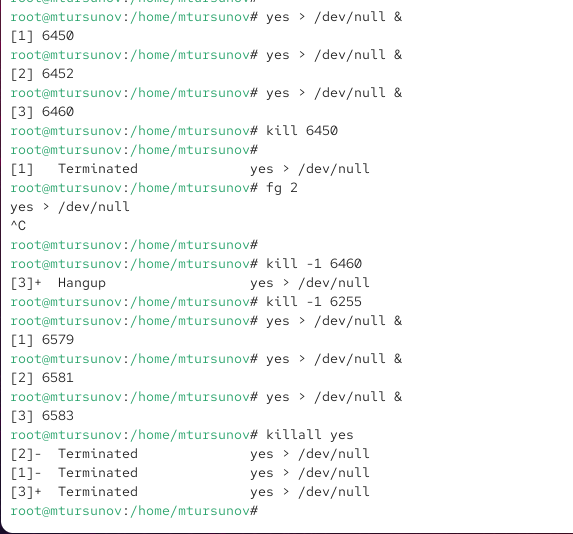
1. С помощью команды **jobs** проверено состояние заданий — одно из них выполнялось, другое находилось в состоянии *Stopped*.  
   После команды **fg 1** процесс был переведён на передний план и остановлен.  
   Аналогично, задание 2 было возвращено в фоновый режим командой **bg 2** и возобновило выполнение.
2. Для запуска процесса, сохраняющегося после выхода из терминала, использована команда **nohup yes > /dev/null &**.  
   Программа nohup перенаправила стандартные потоки и позволила процессу продолжать работу даже после закрытия терминала.

* 
* Рис. 8: Использование nohup и управление фоновыми заданиями

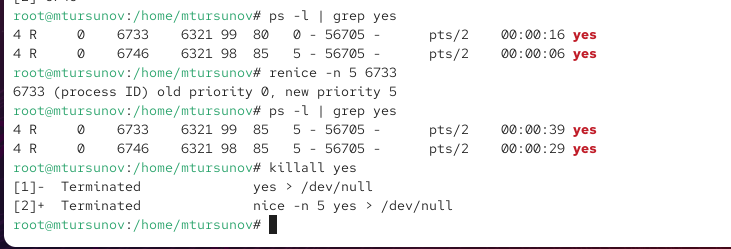
1. После закрытия и повторного открытия консоли с помощью команды **top** было подтверждено, что процессы yes продолжают выполнение.  
   В таблице процессов видно, что они активно используют CPU (около 90–92%).

* 
* Рис. 9: Мониторинг процессов yes через top

1. Далее было запущено три новых процесса yes > /dev/null &.  
   Один из них завершён по PID через **kill 6450**, второй — по идентификатору задания с помощью **fg 2** и **Ctrl + C**.  
   Для отправки сигнала завершения группе процессов использованы команды:
   * **kill -1 6460** — сигнал *SIGHUP*;
   * **kill -1 6255** — завершение процесса, запущенного через nohup, без его прерывания.

* 
* Рис. 10: Завершение процессов yes различными сигналами

1. После этого были вновь запущены три процесса yes > /dev/null &, которые затем были завершены одновременно командой **killall yes**.  
   Система вывела сообщения *Terminated*, подтверждая успешное завершение всех процессов.
2. В заключение запущены два процесса:
   * обычный **yes > /dev/null &**;
   * с повышенным приоритетом: **nice -n 5 yes > /dev/null &**.  
     Проверка приоритетов с помощью **ps -l | grep yes** показала различие значений NI (0 и 5).  
     Затем приоритет одного из процессов был дополнительно изменён с помощью **renice -n 5 6733**, что подтвердилось сообщением:  
     6733 (process ID) old priority 0, new priority 5.

* 
* Рис. 11: Сравнение приоритетов и изменение их с помощью renice

1. В завершение все процессы yes были остановлены командой **killall yes**.  
   В выводе терминала для каждого процесса появилось сообщение *Terminated*, подтверждающее полное завершение всех активных заданий.

# 3 Контрольные вопросы

1. **Какая команда даёт обзор всех текущих заданий оболочки?**  
   Команда **jobs** показывает список всех активных и приостановленных заданий текущей оболочки.
2. **Как остановить текущее задание оболочки, чтобы продолжить его выполнение в фоновом режиме?**  
   Сначала приостановить задание сочетанием клавиш **Ctrl + Z**, затем возобновить его в фоновом режиме с помощью команды **bg**.
3. **Какую комбинацию клавиш можно использовать для отмены текущего задания оболочки?**  
   Для принудительного завершения текущего задания используется комбинация **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?**  
   Можно использовать команду **kill**  или **killall**  из другой сессии или под другим пользователем с правами администратора.
5. **Какая команда используется для отображения отношений между родительскими и дочерними процессами?**  
   Команда **ps fax** показывает древовидную структуру процессов и их взаимосвязи.
6. **Какая команда позволит изменить приоритет процесса с идентификатором 1234 на более высокий?**  
   Команда **renice -n -5 -p 1234** повышает приоритет процесса с PID 1234 (чем меньше значение nice, тем выше приоритет).
7. **В системе в настоящее время запущено 20 процессов dd. Как проще всего остановить их все сразу?**  
   Проще всего использовать команду **killall dd**, которая завершит все процессы с именем dd.
8. **Какая команда позволяет остановить команду с именем mycommand?**  
   Для этого используется **killall mycommand** — она завершает все процессы с указанным именем.
9. **Какая команда используется в top, чтобы убить процесс?**  
   Внутри программы **top** для завершения процесса используется клавиша **k**, после чего нужно ввести PID процесса.
10. **Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов?**  
    Следует использовать утилиту **nice**, указав положительное значение приоритета, например:  
    **nice -n 10**  — это уменьшит приоритет задачи и позволит другим процессам получать больше процессорного времени.

# 4 Заключение

В ходе работы были изучены основные приёмы управления процессами и заданиями в операционной системе **Linux**.  
Были выполнены практические действия по запуску, приостановке, возобновлению и завершению процессов, а также изменению их приоритетов с помощью команд **nice** и **renice**.  
Рассмотрены способы управления заданиями в оболочке, включая перевод процессов между фоновым и передним режимами, использование команд **jobs**, **fg**, **bg**, **kill**, **killall** и **nohup**.  
Также было исследовано влияние приоритетов на распределение процессорного времени и освоены методы мониторинга активности процессов с помощью утилиты **top**.