Отчёт по лабораторной работе №6

Управление процессами

Анна Саенко

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Ход выполнения работы

2.1 Управление заданиями в Linux

Сначала я получила права суперпользователя с помощью команды `su -`.

Затем я запустила несколько процессов:

- `sleep 3600 &` – процесс ожидания на 1 час в фоновом режиме
- `dd if=/dev/zero of=/dev/null &` – утилита для теста скорости ввода-вывода
- `sleep 7200` – процесс ожидания на 2 часа, который изначально запустился на переднем плане

Поскольку последняя команда блокировала терминал, я остановила её комбинацией **Ctrl+Z**. После проверки активных заданий через команду `jobs` я увидела два процесса в состоянии *Running* и один в состоянии *Stopped*.

Чтобы возобновить выполнение третьего задания в фоне, я выполнила команду `bg 3`. После этого все три процесса работали параллельно в фоновом режиме. Затем я последовательно переносила каждое задание на передний план с помощью команды `fg` и завершала его комбинацией **Ctrl+C**.

На скриншоте ниже показана работа с заданиями, их остановка и завершение:

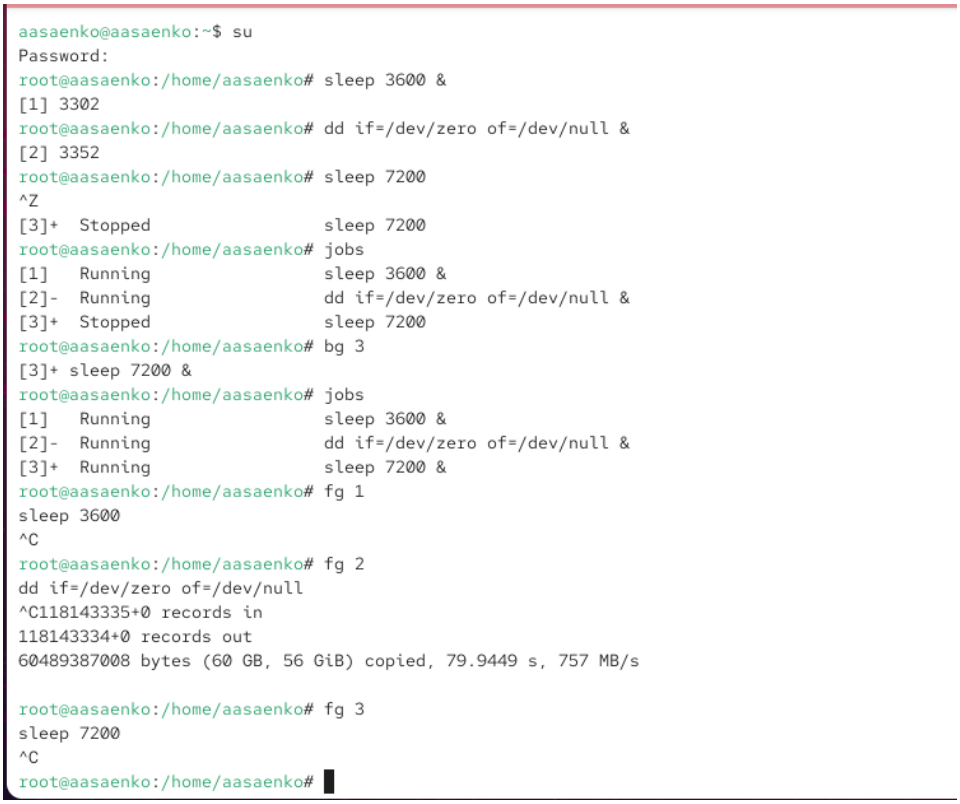

Рис. 2.1: Перемещение заданий в передний план и завершение

Далее я открыла второй терминал под своей учётной записью и запустила команду `dd if=/dev/zero of=/dev/null &`. После выхода из терминала процесс продолжал выполняться.

Чтобы убедиться в этом, я использовала команду `top` в другом терминале и увидела, что процесс `dd` всё ещё активен и занимает 100% CPU.

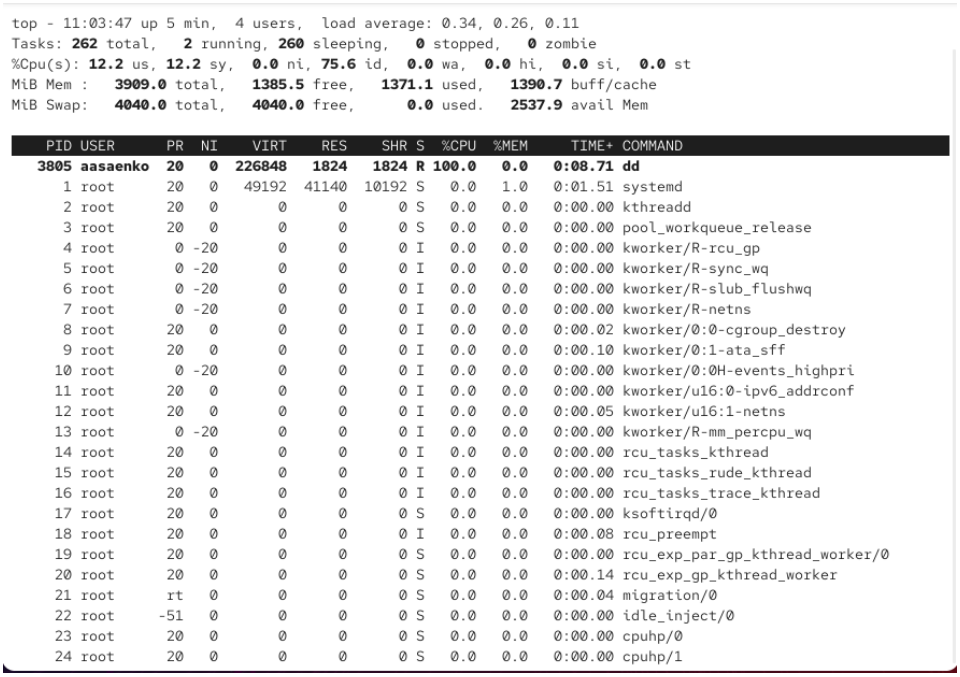

Рис. 2.2: Отображение процесса dd в top

Затем я снова открыла `top` и с помощью клавиши **k** завершила процесс `dd`. После этого он исчез из списка выполняющихся заданий.

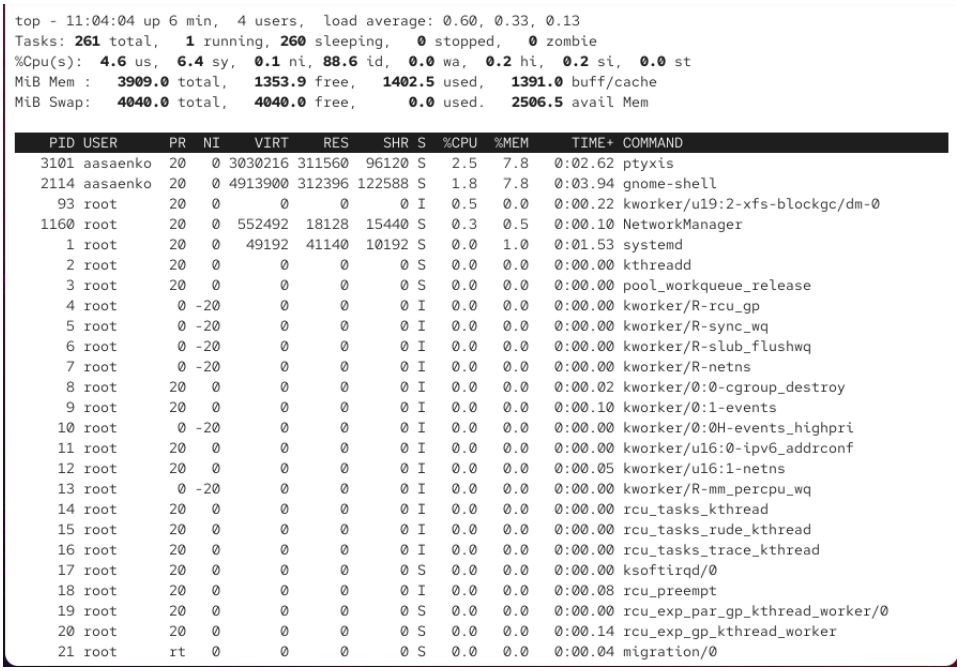

Рис. 2.3: Завершение процесса dd через top

2.2 Управление процессами в Linux

Сначала я получила права суперпользователя с помощью команды `su -`.

Затем я запустила три процесса `dd if=/dev/zero of=/dev/null &`, которые выполняли интенсивные операции записи в `/dev/null` в фоновом режиме.

После этого я проверила список процессов командой `ps aux | grep dd`. В выводе были видны три активных процесса dd, запущенные в системе.

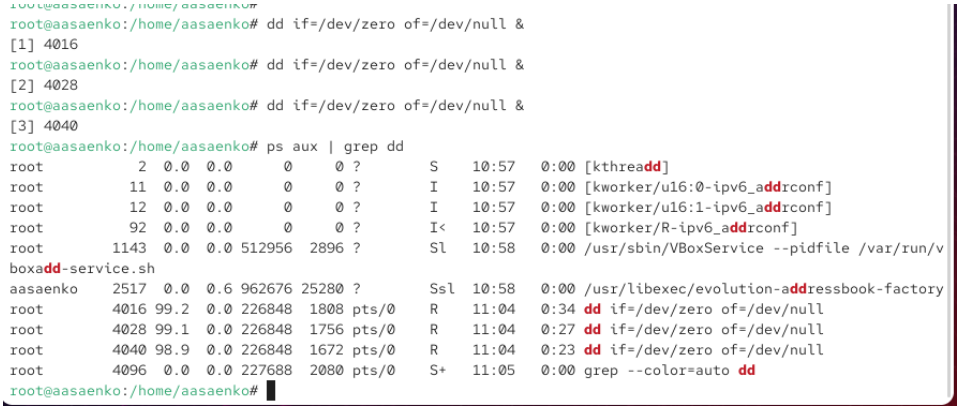

Рис. 2.4: Просмотр процессов dd через ps aux

Далее я использовала команду `renice -n 5 <PID>`, чтобы изменить приоритет одного из процессов dd. Это позволяет управлять нагрузкой на систему, задавая процессу больший или меньший приоритет выполнения.

Чтобы посмотреть дерево процессов и их взаимосвязи, я выполнила команду `ps fax | grep -B5 dd`. Благодаря параметру `-B5` вывод содержал также строки, предшествующие найденным, что позволило увидеть родительский процесс, из которого были запущены процессы dd.

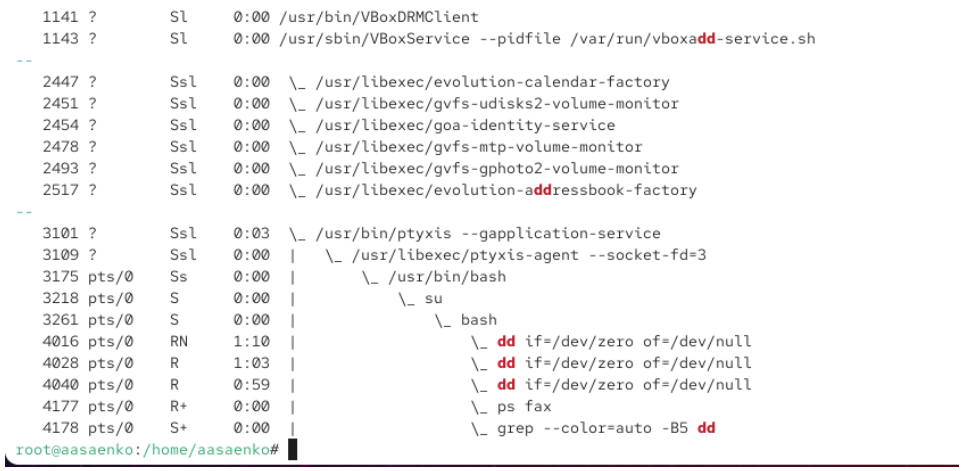

Рис. 2.5: Дерево процессов dd через ps fax

После этого я нашла PID родительской оболочки, из которой запускались процессы dd, и завершила её с помощью команды `kill -9 <PID>`. В результате оболочка завершила работу, а вместе с ней автоматически остановились и все её дочерние процессы dd.

2.3 Задание 1.

Сначала я получила права суперпользователя с помощью команды `su -`.

Затем трижды запустила процесс `dd if=/dev/zero of=/dev/null &` в фоновом режиме. В результате были созданы три параллельных процесса dd.

После этого я изменила приоритет одного из процессов с помощью команды `renice -n 5 <PID>`. Его старый приоритет был равен 0, а новый стал равен 5.

Затем я повторно изменила приоритет этого же процесса на более низкий уровень, выполнив `renice -n 15 <PID>`. В выводе видно, что приоритет изменился с 5 на 15.

Чем больше положительное значение nice, тем **ниже приоритет** процесса: системе позволяется уделять ему меньше процессорного времени. В то же время уменьшение значения nice (например, до -5 или -15) повышает приоритет, что делает процесс более «агрессивным» в использовании CPU.

Для завершения всех запущенных процессов `dd` я использовала команду `killall dd`. Все три процесса были корректно остановлены.

```
aasaenko@aasaenko:~$ su
Password:
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[1] 4500
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[2] 4512
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[3] 4514
root@aasaenko:/home/aasaenko# renice -n 5 4500
4500 (process ID) old priority 0, new priority 5
root@aasaenko:/home/aasaenko# renice -n 15 4500
4500 (process ID) old priority 5, new priority 15
root@aasaenko:/home/aasaenko# 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@aasaenko:/home/aasaenko#
```

Рис. 2.6: Запуск процессов dd, изменение приоритетов и завершение

2.4 Задание 2.

Сначала я запустила программу `yes` в фоновом режиме с перенаправлением вывода в `/dev/null`.

Затем я запустила её на переднем плане и приостановила выполнение комбинацией **Ctrl+Z**. После проверки списка заданий с помощью `jobs` было видно одно задание в состоянии *Running* и одно в состоянии *Stopped*.

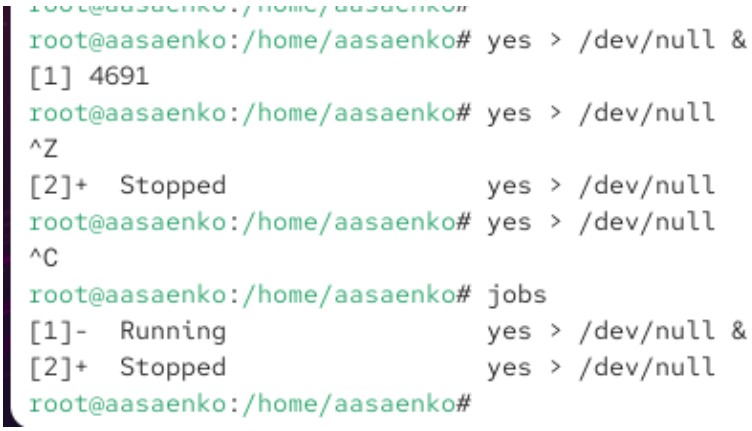

Рис. 2.7: Запуск yes и просмотр заданий

Я возобновила выполнение второго процесса в фоне командой `bg 2`. После этого оба процесса находились в состоянии *Running*. Также я запустила процесс `yes` с помощью `nohup`, что позволило ему продолжить выполнение даже после закрытия терминала.

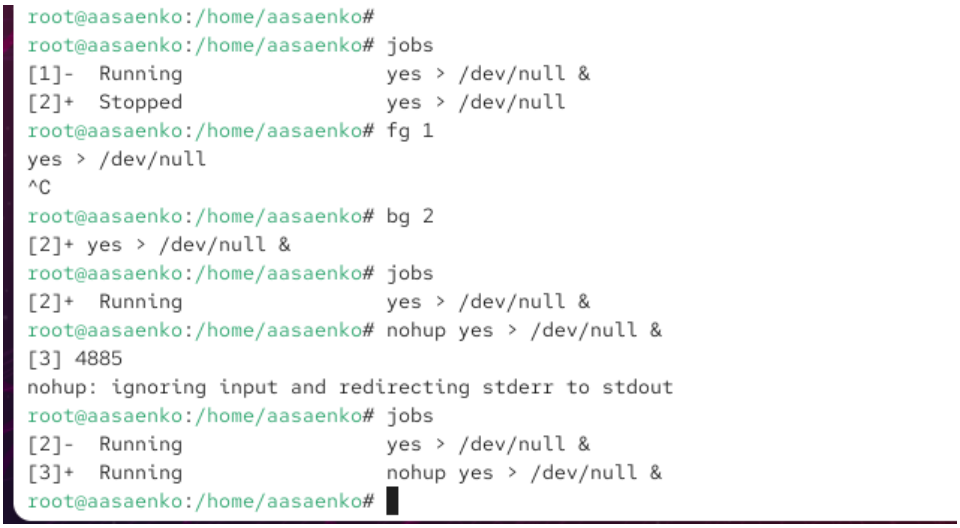

Рис. 2.8: Перезапуск процесса yes и использование nohup

Далее я открыла утилиту `top` и убедилась, что процессы `yes` действительно продолжают работать и занимают процессорное время.

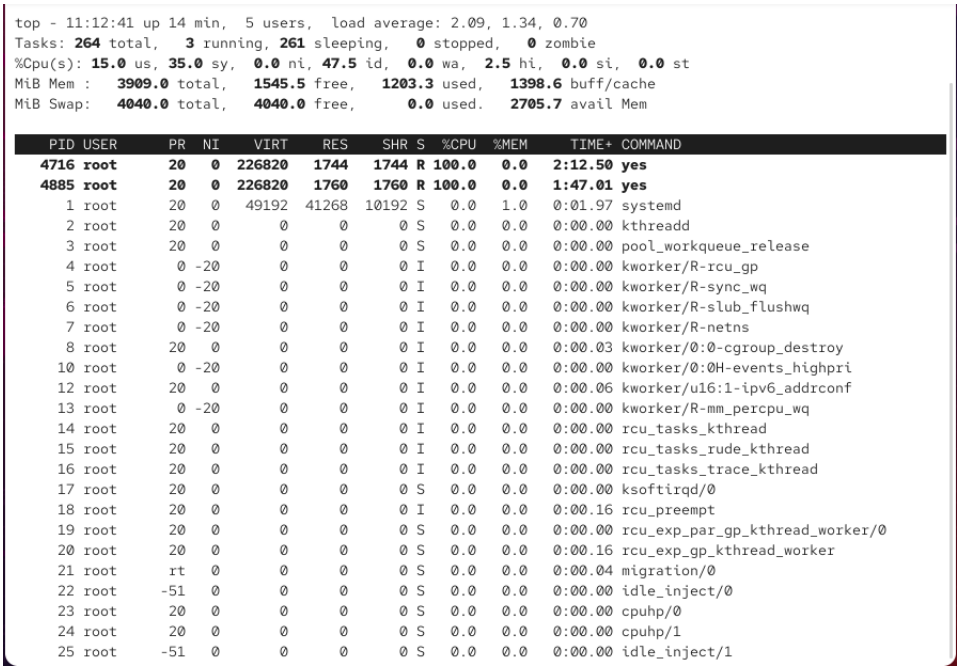

Рис. 2.9: Просмотр процессов yes через top

После этого я запустила ещё несколько экземпляров программы `yes` в фоновом режиме и завершила их разными способами:

- через команду `kill <PID>` для конкретного процесса,
- через `kill -1 <PID>` (сигнал SIGHUP),
- через `kill %<номер_задания>` для завершения по идентификатору задания,
- командой `killall yes` для остановки всех процессов программы.

Рис. 2.10: Завершение процессов yes разными методами

В завершение я сравнила приоритеты процессов. Один из них был запущен с обычным приоритетом, второй — с помощью команды `nice -n 5 yes > /dev/null &`, что установило ему более низкий приоритет (меньшее потребление CPU). Затем с помощью `renice` я изменила приоритет так, чтобы оба процесса имели одинаковые значения.

```
root@aasaenko:/home/aasaenko# yes > /dev/null &
[1] 5568
root@aasaenko:/home/aasaenko# nice -n 5 yes > /dev/null &
[2] 5594
root@aasaenko:/home/aasaenko# ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0    5190    5155  0  80   0 - 58153 do_wai pts/1    00:00:00 su
4 S     0    5201    5190  0  80   0 - 57575 do_wai pts/1    00:00:00 bash
4 R     0    5568    5201 99  80   0 - 56705 -      pts/1    00:00:14 yes
4 R     0    5594    5201 99  85   5 - 56705 -      pts/1    00:00:05 yes
4 R     0    5606    5201  0  80   0 - 57682 -      pts/1    00:00:00 ps
root@aasaenko:/home/aasaenko# renice -n 5 5568
5568 (process ID) old priority 0, new priority 5
root@aasaenko:/home/aasaenko# ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0    5190    5155  0  80   0 - 58153 do_wai pts/1    00:00:00 su
4 S     0    5201    5190  0  80   0 - 57575 do_wai pts/1    00:00:00 bash
4 R     0    5568    5201 99  85   5 - 56705 -      pts/1    00:00:54 yes
4 R     0    5594    5201 99  85   5 - 56705 -      pts/1    00:00:46 yes
4 R     0    5692    5201  0  80   0 - 57682 -      pts/1    00:00:00 ps
root@aasaenko:/home/aasaenko# killall yes
[1]-  Terminated              yes > /dev/null
[2]+  Terminated              nice -n 5 yes > /dev/null
root@aasaenko:/home/aasaenko#
```

Рис. 2.11: Сравнение приоритетов yes и изменение их через renice

3 Контрольные вопросы

1. **Какая команда даёт обзор всех текущих заданий оболочки?**
   Для этого используется команда `jobs`, которая показывает список всех фоновых и приостановленных заданий текущей оболочки.

2. **Как остановить текущее задание оболочки, чтобы продолжить его выполнение в фоновом режиме?**
   Сначала нужно приостановить задание комбинацией клавиш **Ctrl+Z**, а затем возобновить его выполнение в фоне командой `bg`.

3. **Какую комбинацию клавиш можно использовать для отмены текущего задания оболочки?**
   Для этого используется комбинация **Ctrl+C**, которая посылает процессу сигнал прерывания (SIGINT).

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 другую оболочку или терминал и завершить процесс командой `kill <PID>` или `killall <имя_процесса>`.

5. **Какая команда используется для отображения отношений между родительскими и дочерними процессами?**
   Для этого применяется команда `ps fax`, которая выводит дерево процессов.

6. **Какая команда позволит изменить приоритет процесса с идентифика-**

тором 1234 на более высокий?

Для этого используется команда `renice -n -5 -p 1234`, где отрицательное значение nice повышает приоритет.

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

   Самый простой способ — использовать команду `killall dd`.

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

   Для этого применяется команда `killall mycommand`.

9. **Какая команда используется в top, чтобы убить процесс?**

   В утилите `top` используется клавиша **k**, после чего нужно указать PID процесса.

10. **Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов?**

    Для этого используют команду `nice` с положительным значением приоритета, например:

    `nice -n 10 <команда>`.

    Это уменьшает приоритет процесса, освобождая ресурсы для остальных.

4 Заключение

В ходе выполнения лабораторной работы я освоила основы управления процессами и заданиями в Linux.

Были выполнены следующие действия:

- запуск процессов на переднем и фоновом режиме;
- приостановка заданий с помощью комбинации клавиш **Ctrl+Z** и их возобновление через `bg`;
- завершение процессов с использованием **Ctrl+C**, `kill`, `kill -9`, `killall` и встроенных возможностей утилиты `top`;
- использование команд `jobs`, `ps aux` и `ps fax` для мониторинга и анализа процессов;
- управление приоритетами с помощью `nice` и `renice`;
- применение `nohup` для продолжения работы процессов после закрытия терминала.

В процессе работы я закрепила знания о взаимодействии между родительскими и дочерними процессами, а также о влиянии приоритета на использование ресурсов системы.

Полученный опыт показал, как с помощью стандартных инструментов Linux можно эффективно управлять задачами, контролировать нагрузку и завершать процессы разными способами.