Отчёт по лабораторной работе №6

Управление процессами

Агджабекова Эся Рустамовна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Ход выполнения работы

2.1 Управление заданиями

1. Получены права администратора с помощью команды `su`.

2. Запущены процессы: `sleep 3600 &`, `dd if=/dev/zero of=/dev/null &`, а также `sleep 7200` без амперсанда в конце. Последний процесс был остановлен комбинацией `Ctrl+Z`.

3. Просмотрен список заданий командой `jobs`, из которого видно, что два первых процесса находятся в состоянии **Running**, а третий — **Stopped** (см. рис. fig. 2.1).

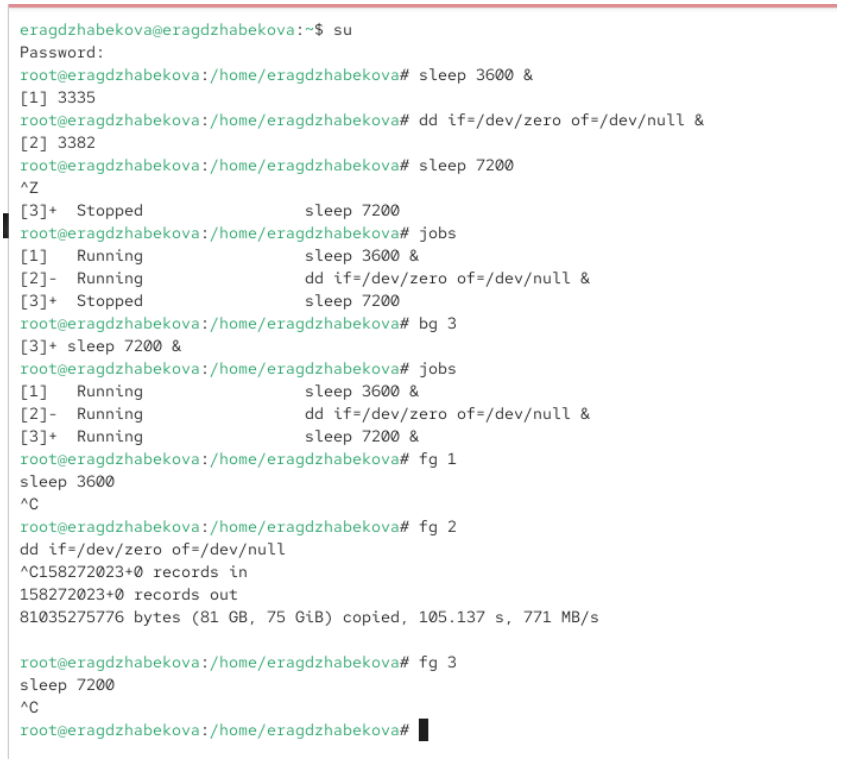

Рис. 2.1: Запуск и управление заданиями

4. Задание 3 переведено в фоновый режим с помощью команды `bg 3`, после чего его статус изменился на **Running**.

5. Задание 1 возвращено на передний план командой `fg 1`, а затем прервано комбинацией `Ctrl+C`. Аналогично были завершены задания 2 и 3 (см. рис. fig. 2.2).

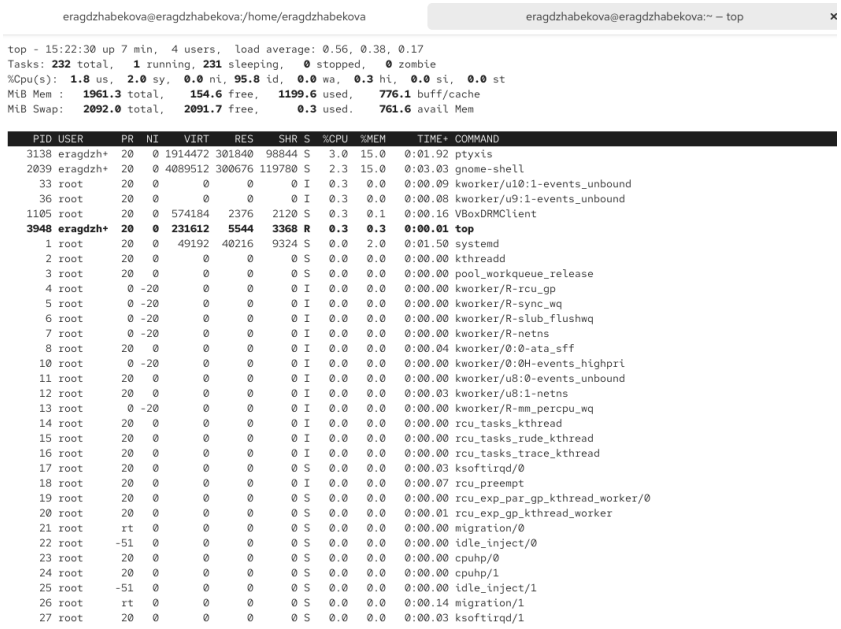

Рис. 2.2: Работа с заданиями в фоне и переднем плане

6. В отдельном терминале, под своей учётной записью, был запущен процесс `dd if=/dev/zero of=/dev/null &`, затем сессия завершена командой `exit`.

7. В другом терминале выполнена команда `top`, которая показала активный процесс `dd` (см. рис. fig. 2.3).

8. С помощью встроенной команды `k` в `top` процесс `dd` был завершён, после чего программа `top` закрыта.

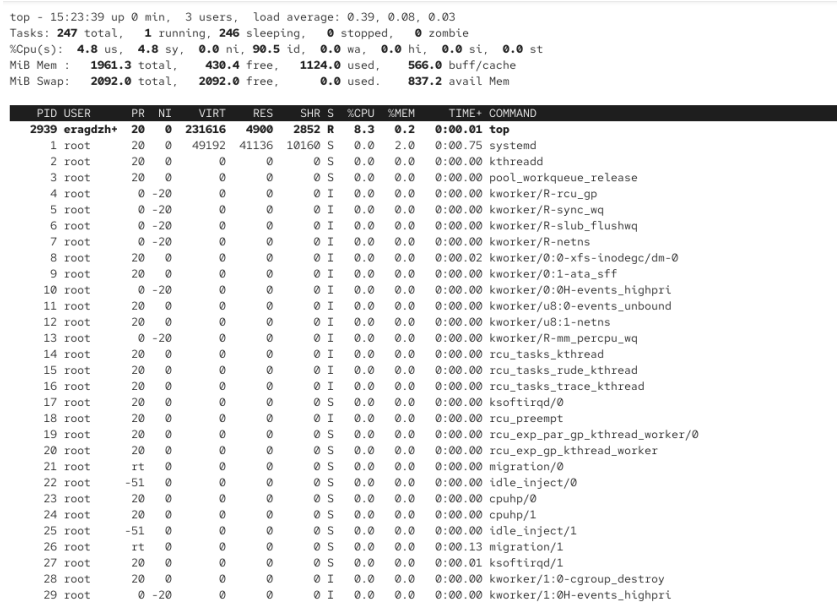

Рис. 2.3: Мониторинг процессов в top

2.2 Управление процессами

1. Получены права администратора с помощью команды `su`.

2. Запущены три процесса с использованием команды `dd if=/dev/zero of=/dev/null &`.

3. Для просмотра информации о запущенных процессах применена команда `ps aux | grep dd`, которая вывела список процессов, содержащих подстроку **dd**. В нём видны три запущенных процесса dd (см. рис. fig. 2.4).

Рис. 2.4: Просмотр процессов с помощью ps aux

4. Приоритет одного из процессов изменён командой `renice -n 5 <PID>`, где `<PID>` — идентификатор одного из процессов dd. В результате его приоритет был изменён с 0 на 5.

5. Для отображения иерархии процессов применена команда `ps fax | grep -B5 dd`. На выходе видно дерево процессов, в котором процессы dd являются дочерними по отношению к оболочке root (см. рис. fig. 2.5).

Рис. 2.5: Отображение иерархии процессов

6. Определён PID родительской оболочки, из которой были запущены процессы dd. Командой

`kill -9 <PID>` оболочка завершена, вследствие чего завершились и все её дочерние процессы. Такой способ позволяет быстро остановить группу связанных процессов.

2.3 Изменение приоритета процессов

1. Запущены три процесса командой `dd if=/dev/zero of=/dev/null &` в фоновом режиме.

2. Приоритет одного из процессов изменён командой `renice -n -5 <PID>`, что повысило его приоритет относительно остальных (см. рис. fig. 2.6).

3. Тот же процесс был повторно изменён с использованием команды `renice -n -15 <PID>`. В отличие от предыдущего изменения, значение `-15` ещё больше повышает приоритет выполнения процесса по сравнению с другими заданиями в системе. Чем **ниже числовое значение nice**, тем выше приоритет.

4. Все процессы `dd` были завершены командой `killall dd` (см. рис. fig. 2.7).

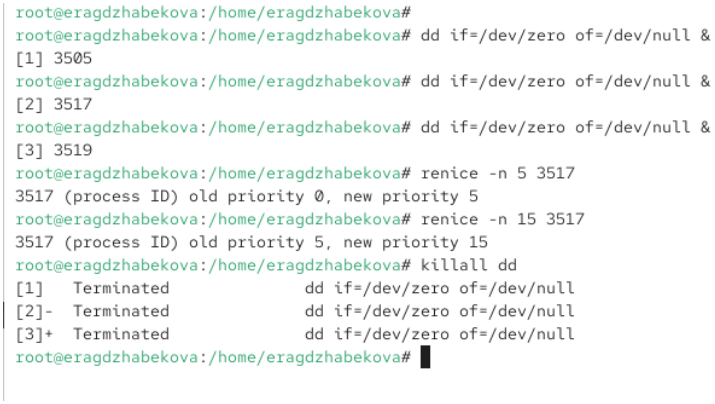

Рис. 2.6: Изменение приоритета и завершение процессов

2.4 Работа с процессами и приоритетами (команда yes)

1. Программа `yes` была запущена в фоновом режиме с перенаправлением вывода в `/dev/null`.

2. Та же программа была запущена на переднем плане, затем приостановлена с помощью `Ctrl+Z`, возвращена в выполнение и остановлена комбинацией `Ctrl+C` (см. рис. fig. **??**).

3. Аналогичные действия были выполнены для варианта запуска без перенаправления потока вывода.

4. С помощью команды `jobs` проверено состояние заданий.

5. Один из фоновых процессов был возвращён на передний план (`fg`) и остановлен.

6. Другой процесс был переведён в фоновый режим (`bg`) и отмечен как **Running** (см. рис. fig. **??**).

7. Процесс был запущен с использованием nohup, что позволило ему продолжать выполнение даже после выхода из терминала.

```
root@eragdzhabekova:/home/eragdzhabekova#
root@eragdzhabekova:/home/eragdzhabekova# yes > /dev/null &
[1] 4105
root@eragdzhabekova:/home/eragdzhabekova# yes > /dev/null
^Z
[2]+  Stopped                 yes > /dev/null
root@eragdzhabekova:/home/eragdzhabekova# fg 2
yes > /dev/null
^C
root@eragdzhabekova:/home/eragdzhabekova#
```

```
root@eragdzhabekova:/home/eragdzhabekova# jobs
[1]+  Running                 yes > /dev/null &
root@eragdzhabekova:/home/eragdzhabekova# fg 1
yes > /dev/null
^Z
[1]+  Stopped                 yes > /dev/null
root@eragdzhabekova:/home/eragdzhabekova# jobs
[1]+  Stopped                 yes > /dev/null
root@eragdzhabekova:/home/eragdzhabekova# bg 1
[1]+ yes > /dev/null &
root@eragdzhabekova:/home/eragdzhabekova# jobs
[1]+  Running                 yes > /dev/null &
root@eragdzhabekova:/home/eragdzhabekova# nohup
[2] 4394
nohup: ignoring input and redirecting stderr to
root@eragdzhabekova:/home/eragdzhabekova#
```

8. В утилите top было подтверждено выполнение процесса yes, занимающего значительную долю CPU (см. рис. fig. 2.7).

```
top - 15:34:01 up 10 min,  5 users,  load average: 1.19, 0.99, 0.66
Tasks: 234 total,   3 running, 231 sleeping,   0 stopped,   0 zombie
%Cpu(s): 33.3 us, 66.7 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1961.3 total,    227.0 free,   1090.3 used,    810.0 buff/cache
MiB Swap:   2092.0 total,   2091.5 free,      0.5 used.    871.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   4394 root      20   0  226820   1752   1752 R  90.9   0.1   0:27.82 yes
   4105 root      20   0  226820   1796   1796 R  81.8   0.1   2:29.30 yes
      1 root      20   0   49192  39024   7920 S   0.0   1.9   0:01.31 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-sync_wq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-slub_flushwq
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
      8 root      20   0       0      0      0 I   0.0   0.0   0:00.06 kworker/0:0-events
      9 root      20   0       0      0      0 I   0.0   0.0   0:00.06 kworker/0:1-cgroup_destroy
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/u8:0-events_unbound
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.02 kworker/u8:1-netns
     13 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-mm_percpu_wq
     14 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthread
     15 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     16 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
     17 root      20   0       0      0      0 S   0.0   0.0   0:00.02 ksoftirqd/0
     18 root      20   0       0      0      0 I   0.0   0.0   0:00.07 rcu_preempt
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_exp_par_gp_kthread_worker/0
     20 root      20   0       0      0      0 S   0.0   0.0   0:00.01 rcu_exp_gp_kthread_worker
     21 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
```

Рис. 2.7: Отслеживание процесса yes в top

9. В фоновом режиме были запущены ещё три процесса yes с перенаправлением вывода. Два из них были завершены разными способами:

   - по PID (kill -9 <PID>),

- по идентификатору задания (`kill -l <job_id>`).

10. Процесс, запущенный с `nohup`, корректно обработал сигнал `SIGHUP`, продолжая работу, в отличие от обычного процесса, который был завершён (см. рис. fig. 2.8).

Рис. 2.8: Завершение процессов yes

11. Все запущенные процессы `yes` были завершены одновременно с помощью команды `killall yes`.

12. Для сравнения приоритетов одна программа `yes` была запущена в фоне с обычным приоритетом, а вторая — с помощью команды `nice -n 5`, что задало ей более низкий приоритет. Проверка с помощью `ps -l` показала различие значений **NI** и **PRI**.

13. Командой `renice` приоритет одного процесса был изменён, в результате чего оба процесса получили одинаковый приоритет (см. рис. fig. 2.9).

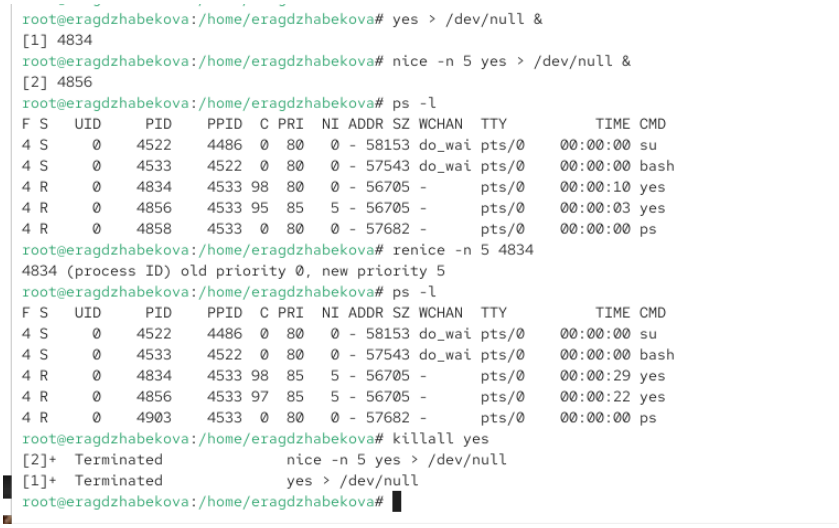

Рис. 2.9: Изменение приоритета процессов yes

3 Контрольные вопросы

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

2. **Как остановить текущее задание оболочки, чтобы продолжить его выполнение в фоновом режиме?**
   Комбинация клавиш `Ctrl+Z` останавливает выполнение, а затем командой `bg` можно продолжить работу в фоне.

3. **Какую комбинацию клавиш можно использовать для отмены текущего задания оболочки?**
   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?**
   Использовать команду `kill <PID>` или `killall <имя_процесса>` в другой оболочке/терминале.

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

6. **Какая команда позволит изменить приоритет процесса с идентификатором 1234 на более высокий?**

Команда: `renice -n -5 -p 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?**
   Команда: `killall dd`.

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

9. **Какая команда используется в top, чтобы убить процесс?**
   В `top` используется клавиша `k`, затем вводится PID процесса.

10. **Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов?**
    Использовать команду `nice` с положительным значением:
    `nice -n 10 <команда>` — это снижает приоритет процесса и уменьшает нагрузку на систему.

4 Заключение

В ходе лабораторной работы были изучены основные приёмы управления заданиями и процессами в Linux. Освоены способы запуска процессов в фоновом и переднем режиме, их приостановки, возобновления и завершения. Получены практические навыки использования команд `jobs`, `fg`, `bg`, `kill`, `killall`, `ps`, `top`, а также изменения приоритетов процессов с помощью `nice` и `renice`.
Закреплены умения отслеживать взаимосвязь родительских и дочерних процессов и контролировать использование системных ресурсов.