

Отчёт по лабораторной работе №6

Управление процессами

Турсунов Мухамметназар

Содержание

| | | |
|----------|--|-----------|
| 1 | Цель работы | 5 |
| 2 | Выполнение | 6 |
| 2.1 | Управление заданиями | 6 |
| 2.2 | Управление процессами | 9 |
| 2.3 | Задание 1. Управление приоритетами процессов | 11 |
| 2.4 | Задание 2. Управление заданиями и приоритетами процессов . . . | 13 |
| 3 | Контрольные вопросы | 17 |
| 4 | Заключение | 19 |

Список иллюстраций

| | | |
|------|--|----|
| 2.1 | Работа с заданиями, управление приоритетами и завершение процессов | 7 |
| 2.2 | Команда top с отображением активного процесса dd | 8 |
| 2.3 | Завершение процесса dd в top | 9 |
| 2.4 | Запуск трёх фоновых процессов dd | 10 |
| 2.5 | Просмотр иерархии процессов и изменение приоритета dd | 11 |
| 2.6 | Запуск трёх фоновых процессов dd | 12 |
| 2.7 | Запуск и управление процессами yes | 13 |
| 2.8 | Использование nohup и управление фоновыми заданиями | 14 |
| 2.9 | Мониторинг процессов yes через top | 14 |
| 2.10 | Завершение процессов yes различными сигналами | 15 |
| 2.11 | Сравнение приоритетов и изменение их с помощью renice | 16 |

Список таблиц

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.

3. 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.

4. 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.

```
mtursunov@mtursunov:~$ su
Password:
root@mtursunov:/home/mtursunov#
root@mtursunov:/home/mtursunov# sleep 3600 &
[1] 3764
root@mtursunov:/home/mtursunov# dd if=/dev/zero of=/dev/null &
[2] 3808
root@mtursunov:/home/mtursunov# sleep 7200
^Z
[3]+  Stopped                  sleep 7200
root@mtursunov:/home/mtursunov# jobs
[1]  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Stopped                  sleep 7200
root@mtursunov:/home/mtursunov# bg 3
[3]+  sleep 7200 &
root@mtursunov:/home/mtursunov# jobs
[1]  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Running                  sleep 7200 &
root@mtursunov:/home/mtursunov# fg 1
sleep 3600
^C
root@mtursunov:/home/mtursunov# fg 2
dd if=/dev/zero of=/dev/null
^C98311585+0 records in
98311585+0 records out
50335531520 bytes (50 GB, 47 GiB) copied, 67.2149 s, 749 MB/s

root@mtursunov:/home/mtursunov# fg 3
sleep 7200
^C
root@mtursunov:/home/mtursunov#
```

Рис. 2.1: Работа с заданиями, управление приоритетами и завершение процессов

5. 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.

- 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)**.

```
top - 11:39:14 up 22 min, 4 users, load average: 0.69, 0.38, 0.18
Tasks: 260 total, 2 running, 258 sleeping, 0 stopped, 0 zombie
%Cpu(s): 12.2 us, 12.2 sy, 0.0 ni, 75.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3653.0 total, 1890.9 free, 1351.7 used, 660.2 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2301.3 avail Mem
```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|------|----------|-----|-----|--------|-------|-------|---|------|------|---------|---------------------------------|
| 4374 | mtursun+ | 20 | 0 | 226848 | 1752 | 1752 | R | 91.7 | 0.0 | 1:09.16 | dd |
| 1132 | root | 20 | 0 | 574184 | 2124 | 1996 | S | 8.3 | 0.1 | 0:00.61 | VBoxDRMClient |
| 1 | root | 20 | 0 | 49196 | 41288 | 10276 | S | 0.0 | 1.1 | 0:01.47 | 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 |
| 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/u16:0-events_unbound |
| 12 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.03 | kworker/u16:1-ipv6_addrconf |
| 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 | 0:00.00 | ksoftirqd/0 |
| 18 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.29 | 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 |
| 22 | root | -51 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | idle_inject/0 |
| 23 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/0 |
| 24 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/1 |

Рис. 2.2: Команда **top** с отображением активного процесса **dd**

- 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.


```
top - 11:40:31 up 24 min, 4 users, load average: 1.00, 0.56, 0.27
Tasks: 259 total, 4 running, 255 sleeping, 0 stopped, 0 zombie
%Cpu(s): 8.4 us, 8.6 sy, 0.0 ni, 82.7 id, 0.0 wa, 0.3 hi, 0.0 si, 0.0 st
MiB Mem : 3653.0 total, 1894.8 free, 1346.6 used, 661.5 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2306.4 avail Mem
```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|------|----------|-----|-----|---------|--------|--------|---|------|------|---------|---------------------------------|
| 3514 | mtursun+ | 20 | 0 | 3029052 | 357484 | 100028 | R | 4.5 | 9.6 | 0:02.72 | ptyxis |
| 2518 | mtursun+ | 20 | 0 | 4907580 | 311508 | 122364 | S | 3.5 | 8.3 | 0:03.56 | gnome-shell |
| 3652 | root | 20 | 0 | 0 | 0 | 0 | I | 1.0 | 0.0 | 0:00.16 | kworker/u18:4-events_unbound |
| 1 | root | 20 | 0 | 49196 | 41288 | 10276 | S | 0.0 | 1.1 | 0:01.52 | 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 |
| 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/u16:0-events_unbound |
| 12 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.04 | kworker/u16:1-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 | 0:00.00 | ksoftirqd/0 |
| 18 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.29 | 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 |
| 22 | root | -51 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | idle_inject/0 |
| 23 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/0 |

Рис. 2.3: 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.

```

root@mtursunov:/home/mtursunov#
root@mtursunov:/home/mtursunov# dd if=/dev/zero of=/dev/null &
[1] 5179
root@mtursunov:/home/mtursunov# dd if=/dev/zero of=/dev/null &
[2] 5191
root@mtursunov:/home/mtursunov# dd if=/dev/zero of=/dev/null &
[3] 5193
root@mtursunov:/home/mtursunov# ps aux | grep dd
root      2  0.0  0.0      0   0 ?        S   11:16   0:00 [kthreadd]
root     94  0.0  0.0      0   0 ?        I<  11:16   0:00 [kworker/R-ipv6_addrconf]
root    1134  0.0  0.0 512956 3048 ?        Sl  11:16   0:00 /usr/sbin/VBoxService --pidfile /var/
run/vboxadd-service.sh
mtursun+ 2962  0.0  0.6 1036420 25332 ?        Ssl 11:32   0:00 /usr/libexec/evolution-addressbook-fa
ctory
root     5179 91.8  0.0 226848 1636 pts/0    R   11:43   0:07 dd if=/dev/zero of=/dev/null
root     5191 90.9  0.0 226848 1892 pts/0    R   11:43   0:05 dd if=/dev/zero of=/dev/null
root     5193 91.1  0.0 226848 1816 pts/0    R   11:43   0:05 dd if=/dev/zero of=/dev/null
root     5209  0.0  0.0 227688 2116 pts/0    S+  11:43   0:00 grep --color=auto dd
root@mtursunov:/home/mtursunov# renice -n 5 5193
5193 (process ID) old priority 0, new priority 5
root@mtursunov:/home/mtursunov#

```

Рис. 2.4: Запуск трёх фоновых процессов dd

3. 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.

4. 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.

5. 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.

```
Process Exited from Signal 9 Restart
915 ?      SNs    0:00 /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --
initfile=/lib/alsa/init/00main rdaemon
938 ?      S      0:00 /usr/sbin/chronyd -F 2
968 ?      Ssl    0:00 /usr/sbin/ModemManager
969 ?      Ssl    0:00 /usr/bin/python3 -sP /usr/sbin/firewalld --nofork --nopic
1132 ?     Sl     0:00 /usr/bin/VBoxDRMClient
1134 ?     Sl     0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
--
2891 ?     Ssl    0:00 \_ /usr/libexec/goa-daemon
2896 ?     Ssl    0:00 \_ /usr/libexec/gvfs-gphoto2-volume-monitor
2903 ?     Ssl    0:00 \_ /usr/libexec/goa-identity-service
2910 ?     Ssl    0:00 \_ /usr/libexec/evolution-calendar-factory
2911 ?     Ssl    0:00 \_ /usr/libexec/gvfs-goa-volume-monitor
2962 ?     Ssl    0:00 \_ /usr/libexec/evolution-addressbook-factory
--
3514 ?     Ssl    0:04 \_ /usr/bin/ptyxis --gaplication-service
3525 ?     Ssl    0:00 | \_ /usr/libexec/ptyxis-agent --socket-fd=3
3605 pts/0  Ss     0:00 | \_ /usr/bin/bash
3663 pts/0  S      0:00 | | \_ su
3707 pts/0  S      0:00 | | \_ bash
5179 pts/0  R      1:14 | | \_ dd if=/dev/zero of=/dev/null
5191 pts/0  R      1:12 | | \_ dd if=/dev/zero of=/dev/null
5193 pts/0  RN     1:11 | | \_ dd if=/dev/zero of=/dev/null
5359 pts/0  R+     0:00 | | \_ ps fax
5360 pts/0  S+     0:00 | | \_ grep --color=auto -B5 dd
root@mtursunov:/home/mtursunov# kill -9 3605
Hangup
```

Рис. 2.5: Просмотр иерархии процессов и изменение приоритета dd

- 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

- 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.

```

root@mtursunov:/home/mtursunov#
root@mtursunov:/home/mtursunov# dd if=/dev/zero of=/dev/null &
[1] 5699
root@mtursunov:/home/mtursunov# dd if=/dev/zero of=/dev/null &
[2] 5701
root@mtursunov:/home/mtursunov# dd if=/dev/zero of=/dev/null &
[3] 5703
root@mtursunov:/home/mtursunov# renice -n 5 5701
5701 (process ID) old priority 0, new priority 5
root@mtursunov:/home/mtursunov# renice -n 15 5701
5701 (process ID) old priority 5, new priority 15
root@mtursunov:/home/mtursunov# 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@mtursunov:/home/mtursunov# █

```

Рис. 2.6: Запуск трёх фоновых процессов dd

- 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.

- 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.

- 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 «у» 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.

```
root@mtursunov:/home/mtursunov#  
root@mtursunov:/home/mtursunov# yes > /dev/null &  
[1] 6003  
root@mtursunov:/home/mtursunov# yes > /dev/null  
^Z  
[2]+  Stopped                  yes > /dev/null  
root@mtursunov:/home/mtursunov# yes > /dev/null  
^C  
root@mtursunov:/home/mtursunov#
```

Рис. 2.7: Запуск и управление процессами yes

3. 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.
4. 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.

```

root@mtursunov:/home/mtursunov#
root@mtursunov:/home/mtursunov# jobs
[1]-  Running                  yes > /dev/null &
[2]+  Stopped                  yes > /dev/null
root@mtursunov:/home/mtursunov# fg 1
yes > /dev/null
^C
root@mtursunov:/home/mtursunov# bg 2
[2]+ yes > /dev/null &
root@mtursunov:/home/mtursunov# jobs
[2]+  Running                  yes > /dev/null &
root@mtursunov:/home/mtursunov# nohup yes > /dev/null &
[3] 6255
root@mtursunov:/home/mtursunov# nohup: ignoring input and redirecting stderr to stdout

root@mtursunov:/home/mtursunov# █

```

Рис. 2.8: Использование nohup и управление фоновыми заданиями

5. 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%).

```

top - 11:52:01 up 35 min, 5 users, load average: 1.43, 1.42, 0.93
Tasks: 264 total, 5 running, 259 sleeping, 0 stopped, 0 zombie
%Cpu(s): 9.3 us, 16.7 sy, 0.1 ni, 73.3 id, 0.0 wa, 0.5 hi, 0.1 si, 0.0 st
MiB Mem : 3653.0 total, 1843.4 free, 1373.4 used, 686.6 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2279.6 avail Mem

```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|------|----------|-----|-----|---------|--------|--------|---|------|------|---------|---------------------------------|
| 6255 | root | 20 | 0 | 226820 | 1796 | 1796 | R | 92.0 | 0.0 | 0:23.70 | yes |
| 6045 | root | 20 | 0 | 226820 | 1796 | 1796 | R | 91.7 | 0.0 | 0:47.79 | yes |
| 2518 | mtursun+ | 20 | 0 | 4973168 | 313692 | 122628 | R | 1.7 | 8.4 | 0:08.61 | gnome-shell |
| 3514 | mtursun+ | 20 | 0 | 3029072 | 358124 | 100028 | S | 1.7 | 9.6 | 0:09.84 | ptxvis |
| 3925 | root | 20 | 0 | 0 | 0 | 0 | I | 0.7 | 0.0 | 0:00.67 | kworker/u19:2-events_unbound |
| 1 | root | 20 | 0 | 49196 | 41288 | 10276 | S | 0.3 | 1.1 | 0:02.14 | 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-slab_flushwq |
| 7 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | kworker/R-netns |
| 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/u16:0-events_unbound |
| 12 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.08 | kworker/u16:1-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 | 0:00.01 | ksoftirqd/0 |
| 18 | root | 20 | 0 | 0 | 0 | 0 | R | 0.0 | 0.0 | 0:00.39 | 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 | 0:00.05 | rcu_exp_gp_kthread_worker |
| 21 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | migration/0 |
| 22 | root | -51 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | idle_inject/0 |
| 23 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/0 |

Рис. 2.9: Мониторинг процессов yes через top

6. 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.

```
root@mtursunov:/home/mtursunov#
root@mtursunov:/home/mtursunov# yes > /dev/null &
[1] 6450
root@mtursunov:/home/mtursunov# yes > /dev/null &
[2] 6452
root@mtursunov:/home/mtursunov# yes > /dev/null &
[3] 6460
root@mtursunov:/home/mtursunov# kill 6450
root@mtursunov:/home/mtursunov#
[1] Terminated          yes > /dev/null
root@mtursunov:/home/mtursunov# fg 2
yes > /dev/null
^C
root@mtursunov:/home/mtursunov#
root@mtursunov:/home/mtursunov# kill -1 6460
[3]+ Hangup              yes > /dev/null
root@mtursunov:/home/mtursunov# kill -1 6255
root@mtursunov:/home/mtursunov# yes > /dev/null &
[1] 6579
root@mtursunov:/home/mtursunov# yes > /dev/null &
[2] 6581
root@mtursunov:/home/mtursunov# yes > /dev/null &
[3] 6583
root@mtursunov:/home/mtursunov# killall yes
[2]- Terminated          yes > /dev/null
[1]- Terminated          yes > /dev/null
[3]+ Terminated          yes > /dev/null
root@mtursunov:/home/mtursunov#
```

Рис. 2.10: Завершение процессов *yes* различными сигналами

7. 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.

8. 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.

```
root@mtursunov:/home/mtursunov# ps -l | grep yes
4 R    0    6733    6321 99  80   0 - 56705 -      pts/2    00:00:16  yes
4 R    0    6746    6321 98  85   5 - 56705 -      pts/2    00:00:06  yes
root@mtursunov:/home/mtursunov# renice -n 5 6733
6733 (process ID) old priority 0, new priority 5
root@mtursunov:/home/mtursunov# ps -l | grep yes
4 R    0    6733    6321 99  85   5 - 56705 -      pts/2    00:00:39  yes
4 R    0    6746    6321 98  85   5 - 56705 -      pts/2    00:00:29  yes
root@mtursunov:/home/mtursunov# killall yes
[1]-  Terminated                  yes > /dev/null
[2]+  Terminated                  nice -n 5 yes > /dev/null
root@mtursunov:/home/mtursunov#
```

Рис. 2.11: Сравнение приоритетов и изменение их с помощью renice

9. 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**.