Отчёт по лабораторной работе №6

Управление процессами

Щемелев Илья Владимирович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Ход выполнения

2.1 Управление заданиями и процессами в Linux

1. Для выполнения административных операций получены полномочия суперпользователя с использованием команды `su`. После успешной аутентификации работа продолжена в корневой оболочке.

2. В терминале последовательно запущены три процесса.
   Первый процесс `sleep 3600` был запущен в фоновом режиме. Далее в фоне запущен процесс `dddd if=/dev/zero of=/dev/null`, создающий высокую нагрузку на процессор.
   Последняя команда `sleep 7200` была запущена без использования символа фонового выполнения, в результате чего управление оболочкой было заблокировано.

3. Для возврата управления оболочкой использовано сочетание клавиш **Ctrl + Z**, после чего выполнение процесса `sleep 7200` было приостановлено и переведено в состояние **Stopped**.

4. С помощью команды `jobs` выведен список активных заданий.
   В результате отображены три задания: два в состоянии **Running** и одно в состоянии **Stopped**, что подтверждает корректную работу механизма управления заданиями оболочки.

```
ivschemelev@ivschemelev:~$ su
Password:
root@ivschemelev:/home/ivschemelev# sleep 3600 &
[1] 4305
root@ivschemelev:/home/ivschemelev# dd if=/dev/zero of=/dev/null &
[2] 4350
root@ivschemelev:/home/ivschemelev# sleep 7200
^Z
[3]+  Stopped                 sleep 7200
root@ivschemelev:/home/ivschemelev# jobs
[1]   Running                 sleep 3600 &
[2]-  Running                 dd if=/dev/zero of=/dev/null &
[3]+  Stopped                 sleep 7200
root@ivschemelev:/home/ivschemelev# bg 3
[3]+ sleep 7200 &
root@ivschemelev:/home/ivschemelev# fg 1
sleep 3600
^C
root@ivschemelev:/home/ivschemelev# fg 2
dd if=/dev/zero of=/dev/null
^C186450855+0 records in
186450854+0 records out
95462837248 bytes (95 GB, 89 GiB) copied, 70.0073 s, 1.4 GB/s

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

Рис. 2.1: Просмотр списка заданий

5. Для продолжения выполнения третьего задания в фоновом режиме использована команда `bg 3`.

   Повторный просмотр списка заданий показал, что задание перешло в состояние **Running**.

6. Для переноса первого задания на передний план выполнена команда `fg 1`, в результате чего процесс `sleep 3600` стал активным в текущем терминале.

7. Выполнение задания 1 было завершено с помощью сочетания клавиш **Ctrl + C**.

   Повторная проверка списка заданий показала отсутствие данного процесса.

8. Аналогичным образом были завершены задания 2 и 3, после чего активные задания в текущей сессии отсутствовали.

9. Во втором терминале под учётной записью обычного пользователя был запущен процесс `dd if=/dev/zero of=/dev/null` в фоновом режиме. После этого терминал был закрыт командой выхода из оболочки.

10. В основном терминале под той же учётной записью запущена утилита мониторинга процессов `top`.
    Несмотря на закрытие второго терминала, процесс `dd` продолжал выполняться, что подтверждает его независимость от интерактивной сессии оболочки.

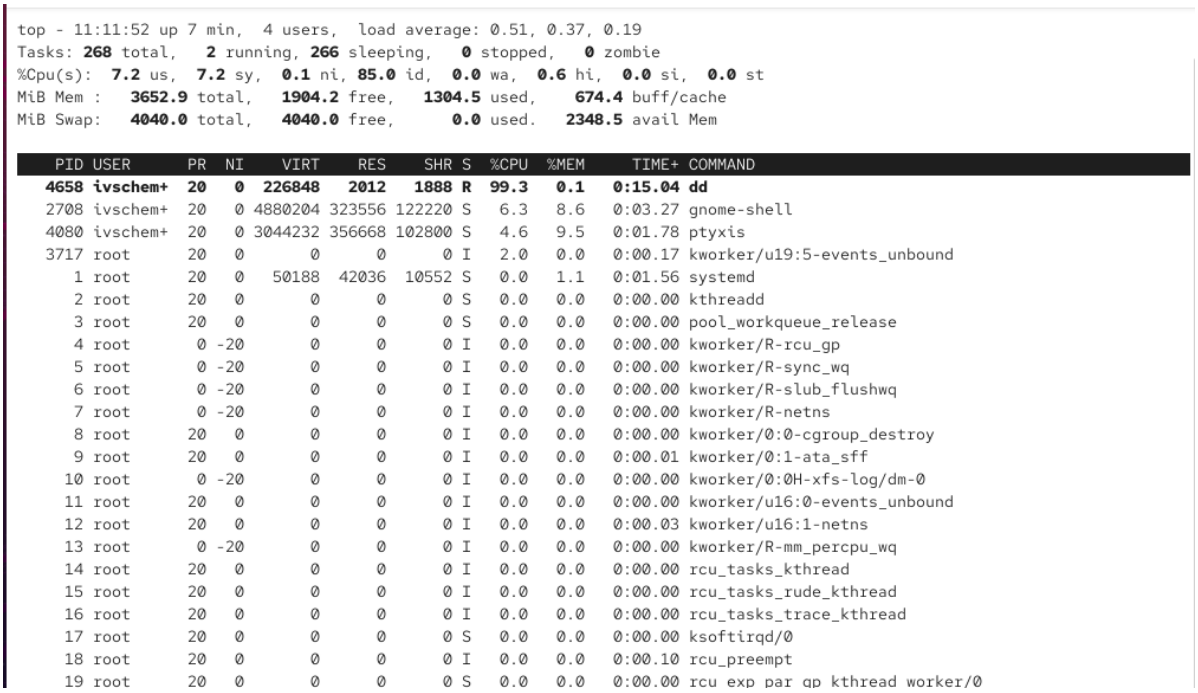

Рис. 2.2: Отображение процесса dd в top

11. После повторного запуска `top` с помощью встроенной команды завершения процесса был принудительно остановлен процесс `dd`. После этого выполнен выход из утилиты мониторинга.

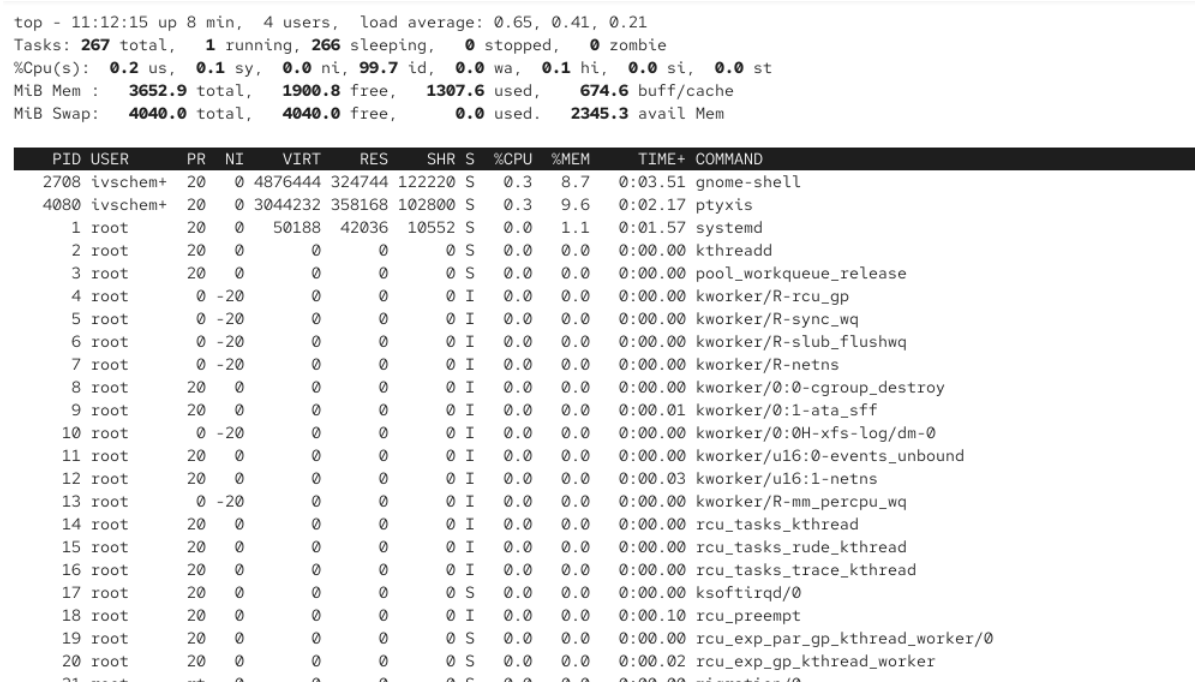

Рис. 2.3: Завершение процесса dd через top

2.2 Управление процессами

1. Повторно получены полномочия администратора.

2. В терминале запущены три процесса `dd if=/dev/zero of=/dev/null` в фоновом режиме, создающие высокую нагрузку на вычислительные ресурсы системы.

3. С помощью просмотра списка процессов, содержащих имя `dd`, были определены идентификаторы всех запущенных процессов данного типа.

4. Используя идентификатор одного из процессов `dd`, выполнено изменение его приоритета. В результате приоритет выбранного процесса был понижен, что подтверждено сообщением системы.

```
root@ivschemelev:/home/ivschemelev# dd if=/dev/zero of=/dev/null &
[1] 5032
root@ivschemelev:/home/ivschemelev# dd if=/dev/zero of=/dev/null &
[2] 5034
root@ivschemelev:/home/ivschemelev# dd if=/dev/zero of=/dev/null &
[3] 5036
root@ivschemelev:/home/ivschemelev# ps aux | grep dd
root           2  0.0  0.0      0     0 ?        S    11:04   0:00 [kthreadd]
root          93  0.0  0.0      0     0 ?        I<   11:04   0:00 [kworker/R-ipv6_addrconf]
dbus        1083  0.0  0.1   8200  6020 ?        S    11:04   0:00 dbus-broker --log 4 --controller 9 --machine-id 473c978a8
05e47e9bc9a702cdd313842 --max-bytes 536870912 --max-fds 4096 --max-matches 131072 --audit
root        1354  0.0  0.0 512956  3476 ?        Sl   11:04   0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.
sh
ivschem+    2618  0.0  0.1   6492  4524 ?        S    11:04   0:00 dbus-broker --log 4 --controller 9 --machine-id 473c978a8
05e47e9bc9a702cdd313842 --max-bytes 100000000000000 --max-fds 25000000000000 --max-matches 5000000000
ivschem+    2761  0.0  0.0   4768  2708 ?        S    11:04   0:00 dbus-broker --log 4 --controller 9 --machine-id 473c978a8
05e47e9bc9a702cdd313842 --max-bytes 100000000000000 --max-fds 6400000 --max-matches 5000000000
ivschem+    3095  0.0  0.6 963600 26000 ?        Ssl  11:04   0:00 /usr/libexec/evolution-addressbook-factory
root        5032 99.4  0.0 226848  1860 pts/0    R    11:13   0:12 dd if=/dev/zero of=/dev/null
root        5034 99.1  0.0 226848  2028 pts/0    R    11:13   0:11 dd if=/dev/zero of=/dev/null
root        5036 98.8  0.0 226848  1888 pts/0    R    11:13   0:10 dd if=/dev/zero of=/dev/null
root        5071  0.0  0.0 227688  2192 pts/0    S+   11:14   0:00 grep --color=auto dd
root@ivschemelev:/home/ivschemelev# renice -n 5 5032
5032 (process ID) old priority 0, new priority 5
root@ivschemelev:/home/ivschemelev#
```

Рис. 2.4: Изменение приоритета процесса

5. Для анализа иерархии процессов выполнен вывод древовидного списка процессов с отображением родительских и дочерних связей. Это позволило определить оболочку, из которой были запущены все процессы dd.

6. После определения идентификатора родительской оболочки выполнено её принудительное завершение. В результате вместе с родительским процессом были автоматически завершены все дочерние процессы dd.

```
--
   3010 ?        Ssl    0:00  \_ /usr/libexec/goa-daemon
   3023 ?        Ssl    0:00  \_ /usr/libexec/gvfs-gphoto2-volume-monitor
   3051 ?        Ssl    0:00  \_ /usr/libexec/evolution-calendar-factory
   3052 ?        Ssl    0:00  \_ /usr/libexec/goa-identity-service
   3053 ?        Ssl    0:00  \_ /usr/libexec/gvfs-goa-volume-monitor
   3095 ?        Ssl    0:00  \_ /usr/libexec/evolution-addressbook-factory
--
   4080 ?        Ssl    0:03  \_ /usr/bin/ptyxis --gapplication-service
   4088 ?        Ssl    0:00  |   \_ /usr/libexec/ptyxis-agent --socket-fd=3 --rlimit-nofile=1024
   4156 pts/0    Ss     0:00  |       \_ /usr/bin/bash
   4205 pts/0    S      0:00  |       |   \_ su
   4249 pts/0    S      0:00  |       |       \_ bash
   5032 pts/0    RN     1:11  |       |           \_ dd if=/dev/zero of=/dev/null
   5034 pts/0    R      1:10  |       |           \_ dd if=/dev/zero of=/dev/null
   5036 pts/0    R      1:08  |       |           \_ dd if=/dev/zero of=/dev/null
   5195 pts/0    R+     0:00  |       |           \_ ps fax
   5196 pts/0    S+     0:00  |       |           \_ grep --color=auto -B5 dd
root@ivschemelev:/home/ivschemelev# kill -9 4249
Killed
```

Рис. 2.5: Завершение родительского процесса и всех дочерних процессов

2.3 Самостоятельная работа

2.3.1 Задание 1. Управление приоритетами процессов

1. В корневой оболочке трижды был запущен процесс, создающий высокую нагрузку на процессор и подсистему ввода-вывода, в фоновом режиме. В результате в системе одновременно выполнялись три процесса одного типа.

```
root@ivschemelev:/home/ivschemelev# dd if=/dev/zero of=/dev/null &
[3] 5551
root@ivschemelev:/home/ivschemelev# renice -n 5 5547
5547 (process ID) old priority 0, new priority 5
root@ivschemelev:/home/ivschemelev# renice -n 15 5547
5547 (process ID) old priority 5, new priority 15
root@ivschemelev:/home/ivschemelev# ps | grep dd
   5032 pts/0    00:03:46 dd
   5034 pts/0    00:04:03 dd
   5036 pts/0    00:04:01 dd
   5547 pts/0    00:00:12 dd
   5549 pts/0    00:00:27 dd
   5551 pts/0    00:00:26 dd
root@ivschemelev:/home/ivschemelev# ps aux| grep dd
root           2  0.0  0.0      0     0 ?        S    11:04   0:00 [kthreadd]
root          93  0.0  0.0      0     0 ?        I<   11:04   0:00 [kworker/R-ipv6_addrconf]
dbus        1083  0.0  0.1   8200  6020 ?        S    11:04   0:00 dbus-broker --log 4 --controller 9 --machine-id 473c978a805e47e9bc9a7
dd313842 --max-bytes 536870912 --max-fds 4096 --max-matches 131072 --audit
root        1354  0.0  0.0 512956  3476 ?        Sl   11:04   0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
ivschem+    2618  0.0  0.1   6492  4524 ?        S    11:04   0:00 dbus-broker --log 4 --controller 9 --machine-id 473c978a805e47e9bc9a7
dd313842 --max-bytes 100000000000000 --max-fds 25000000000000 --max-matches 5000000000
ivschem+    2761  0.0  0.0   4768  2708 ?        S    11:04   0:00 dbus-broker --log 4 --controller 9 --machine-id 473c978a805e47e9bc9a7
dd313842 --max-bytes 100000000000000 --max-fds 6400000 --max-matches 5000000000
ivschem+    3095  0.0  0.6 963600 26000 ?        Ssl  11:04   0:00 /usr/libexec/evolution-addressbook-factory
root        5032 88.8  0.0 226848  1860 pts/0    RN   11:13   3:48 dd if=/dev/zero of=/dev/null
root        5034 96.9  0.0 226848  2028 pts/0    R    11:13   4:08 dd if=/dev/zero of=/dev/null
root        5036 96.4  0.0 226848  1888 pts/0    R    11:13   4:05 dd if=/dev/zero of=/dev/null
root        5547 33.1  0.0 226848  1924 pts/0    RN   11:17   0:13 dd if=/dev/zero of=/dev/null
root        5549 83.6  0.0 226848  1864 pts/0    R    11:17   0:32 dd if=/dev/zero of=/dev/null
root        5551 83.6  0.0 226848  1924 pts/0    R    11:17   0:31 dd if=/dev/zero of=/dev/null
root        5651  0.0  0.0 227688  2132 pts/0    S+   11:18   0:00 grep --color=auto dd
root@ivschemelev:/home/ivschemelev# killall dd
[2]-  Terminated              dd if=/dev/zero of=/dev/null
[3]+  Terminated              dd if=/dev/zero of=/dev/null
[1]+  Terminated              dd if=/dev/zero of=/dev/null
root@ivschemelev:/home/ivschemelev#
```

Рис. 2.6: Запуск процессов dd в фоновом режиме

2. Для одного из запущенных процессов был изменён приоритет выполнения. Сначала значение приоритета было увеличено до −5, что привело к повышению относительного приоритета процесса по сравнению с остальными пользовательскими процессами.

3. Затем приоритет того же процесса был изменён повторно, на значение −15. Разница между значениями −5 и −15 заключается в степени приоритетно-

сти: при значении –15 процесс получает значительно больше процессорного времени, вытесняя другие процессы с более низким приоритетом. Чем меньше числовое значение приоритета, тем выше фактический приоритет выполнения.

4. После завершения эксперимента все запущенные процессы данного типа были корректно остановлены одновременно, что подтвердило возможность массового завершения однотипных процессов.

2.3.2 Задание 2. Управление заданиями и сигналами

1. Программа yes была запущена в фоновом режиме с перенаправлением вывода в специальное устройство, что позволило исключить вывод данных в терминал.

```
root@ivschemelev:/home/ivschemelev#
root@ivschemelev:/home/ivschemelev# yes > /dev/null &
[1] 5985
root@ivschemelev:/home/ivschemelev# yes > /dev/null
^Z
[2]+  Stopped                 yes > /dev/null
root@ivschemelev:/home/ivschemelev# fg 2
yes > /dev/null
^C
root@ivschemelev:/home/ivschemelev# jobs
[1]+  Running                 yes > /dev/null &
root@ivschemelev:/home/ivschemelev#
```

Рис. 2.7: Фоновый запуск yes с подавлением вывода

2. Далее программа yes была запущена на переднем плане с подавлением вывода.
   Выполнение программы было приостановлено, после чего она была повторно запущена с теми же параметрами и завершена принудительно.

3. Затем программа yes была запущена на переднем плане без подавления вывода.

Активный вывод в терминал подтвердил корректность работы программы. Процесс был приостановлен, повторно запущен и завершён.

```
y
y
y
^C
root@ivschemelev:/home/ivschemelev#
root@ivschemelev:/home/ivschemelev# jobs
[1]+  Running                 yes > /dev/null &
root@ivschemelev:/home/ivschemelev# fg 1
yes > /dev/null
^C
root@ivschemelev:/home/ivschemelev# yes > /dev/null &
[1] 6237
root@ivschemelev:/home/ivschemelev# fg 1
yes > /dev/null
^C
root@ivschemelev:/home/ivschemelev# yes > /dev/null
^Z
[1]+  Stopped                 yes > /dev/null
root@ivschemelev:/home/ivschemelev# bg 1
[1]+ yes > /dev/null &
root@ivschemelev:/home/ivschemelev# jobs
[1]+  Running                 yes > /dev/null &
root@ivschemelev:/home/ivschemelev# nohup yes > /dev/null &
[2] 6347
root@ivschemelev:/home/ivschemelev# nohup: ignoring input and redirecting stderr to stdout

root@ivschemelev:/home/ivschemelev#
```

Рис. 2.8: Запуск yes без подавления вывода

4. С помощью просмотра списка заданий определено текущее состояние процессов.
   Было подтверждено наличие фонового задания в состоянии **Running**.

5. Процесс, выполнявшийся в фоновом режиме, был переведён на передний план, после чего его выполнение было остановлено.

6. Один из процессов с подавлением потока вывода был переведён в фоновый режим, после чего повторно проверено состояние заданий.

7. Просмотр состояний заданий показал, что процесс успешно выполняется в фоновом режиме и имеет статус **Running**.

```
top - 11:24:53 up 20 min,  5 users,  load average: 1.78, 1.48, 1.05
Tasks: 264 total,   3 running, 261 sleeping,   0 stopped,   0 zombie
%Cpu(s): 27.0 us, 29.7 sy,  0.0 ni, 43.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3652.9 total,   1871.3 free,   1326.1 used,    685.8 buff/cache
MiB Swap:   4040.0 total,   4040.0 free,      0.0 used.   2326.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   6271 root      20   0  226820   1892   1772 R 100.0   0.1   1:49.52 yes
   6347 root      20   0  226820   1928   1808 R  90.9   0.1   1:15.34 yes
   2708 ivschem+  20   0 4941812 325532 122864 S  27.3   8.7   0:09.04 gnome-shell
   4080 ivschem+  20   0 3044848 365472 102580 S   9.1   9.8   0:09.32 ptyxis
      1 root      20   0   50188  42036  10552 S   0.0   1.1   0:02.13 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
      9 root      20   0       0      0      0 I   0.0   0.0   0:00.01 kworker/0:1-events
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.01 kworker/0:0H-xfs-log/dm-0
     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   0:00.07 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
```

Рис. 2.9: Состояние заданий после перевода в фон

8. Один из процессов был запущен в фоновом режиме таким образом, чтобы продолжать выполнение даже после завершения терминальной сессии.

9. После закрытия окна терминала и повторного запуска консоли было подтверждено, что процесс продолжил своё выполнение, что свидетельствует о его независимости от пользовательской сессии.

10. С помощью утилиты мониторинга процессов был получен обзор текущего состояния системы, включая активные процессы, загрузку процессора и использование памяти.

Рис. 2.10: Мониторинг процессов с помощью top

11. Дополнительно были запущены ещё три процесса `yes` в фоновом режиме с подавлением вывода.

12. Два процесса были завершены различными способами: один — по идентификатору процесса, другой — по идентификатору задания, что продемонстрировало разные подходы к управлению процессами.

13. Выполнена попытка отправки сигнала SIGHUP процессу, запущенному с сохранением работы после выхода из терминала, и обычному процессу.
    В результате обычный процесс завершился, тогда как защищённый процесс продолжил выполнение.

14. После этого было запущено ещё несколько процессов `yes` в фоновом режиме с подавлением вывода.

15. Все запущенные процессы данного типа были завершены одновременно, что подтвердило возможность группового управления процессами.

```
root@ivschemelev:/home/ivschemelev# yes > /dev/null &
[1] 6960
root@ivschemelev:/home/ivschemelev# nice -n 5 yes > /dev/null &
[2] 6983
root@ivschemelev:/home/ivschemelev# ps -l | grep yes
4 R     0    6960    6517 99  80   0 - 56705 -      pts/2    00:00:21 yes
4 R     0    6983    6517 99  85   5 - 56705 -      pts/2    00:00:10 yes
root@ivschemelev:/home/ivschemelev# renice -n 5 6960
6960 (process ID) old priority 0, new priority 5
root@ivschemelev:/home/ivschemelev# ps -l | grep yes
4 R     0    6960    6517 99  85   5 - 56705 -      pts/2    00:00:42 yes
4 R     0    6983    6517 99  85   5 - 56705 -      pts/2    00:00:30 yes
root@ivschemelev:/home/ivschemelev# killall yes
[1]-  Terminated              yes > /dev/null
[2]+  Terminated              nice -n 5 yes > /dev/null
root@ivschemelev:/home/ivschemelev#
```

Рис. 2.11: Завершение процессов yes

16. Программа `yes` была запущена в фоновом режиме, после чего аналогичный процесс был запущен с повышенным относительным приоритетом. Сравнение абсолютных и относительных приоритетов показало различие в распределении процессорного времени между процессами.

17. С помощью изменения приоритета одного из процессов подтверждена возможность динамического управления приоритетами уже запущенных процессов.

3 Контрольные вопросы

1. **Какая команда даёт обзор всех текущих заданий оболочки?**
   Для просмотра всех заданий, запущенных в текущей оболочке, используется команда **jobs**.
   Она отображает список заданий, их номера и текущее состояние (Running, Stopped и т.д.).

2. **Как остановить текущее задание оболочки, чтобы продолжить его выполнение в фоновом режиме?**
   Для этого необходимо:

   - приостановить выполнение текущего задания с помощью сочетания клавиш **Ctrl + Z**;
   - затем возобновить его выполнение в фоновом режиме с помощью команды **bg**.

3. **Какую комбинацию клавиш можно использовать для отмены текущего задания оболочки?**
   Для немедленного завершения текущего задания используется сочетание клавиш **Ctrl + C**, которое отправляет процессу сигнал прерывания.

4. **Необходимо отменить одно из начатых заданий. Доступ к оболочке, в которой в данный момент работает пользователь, невозможен. Что можно сделать, чтобы отменить задание?**
   В этом случае можно:

- определить идентификатор процесса (PID) с помощью утилит просмотра процессов;
- завершить процесс из другой оболочки, отправив ему сигнал завершения.

  Таким образом, задание будет остановлено независимо от доступа к исходной оболочке.

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

   Для отображения иерархии процессов применяется команда **ps** с параметрами, позволяющими вывести древовидную структуру процессов, где видны связи между родительскими и дочерними процессами.

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

   Для изменения приоритета уже запущенного процесса используется команда **renice**.

   Уменьшение числового значения приоритета приводит к его повышению.

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

   Наиболее простой способ — использовать команду **killall**, указав имя процесса.

   Это позволит завершить все процессы с заданным именем одной командой.

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

   Для завершения процесса по его имени используется команда **killall mycommand**, которая отправляет сигнал завершения всем процессам с данным именем.

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

   В интерактивном режиме утилиты **top** для завершения процесса использу-

ется команда **k**, после чего указывается идентификатор процесса и сигнал.

10. **Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов?**
    Для этого используется команда **nice**, позволяющая задать повышенный приоритет при запуске процесса, но в допустимых пределах.
    Такой подход обеспечивает более эффективное выполнение процесса без критического вытеснения других задач системы.

4 Заключение

В ходе выполнения лабораторной работы были изучены и закреплены основные принципы управления заданиями и процессами в операционной системе Linux. Освоены приёмы запуска процессов в фоновом и переднем режиме, их приостановки и завершения, изменения приоритетов выполнения, а также работы с сигналами и мониторинга состояния системы. Полученные навыки позволяют эффективно контролировать выполнение программ и рационально распределять вычислительные ресурсы системы.