

Лабораторная работа №6

Управление процессами

Перфилов Александр Константинович | Группа НПИбд-03-24

Российский университет дружбы народов, Москва, Россия

7 октября 2025

Раздел 1

Информация

Докладчик

- Перфилов Александр Константинович
- Группа НПИбд-03-24
- Российский университет дружбы народов
- <https://github.com/AlexanderPErfilovKonstantinivich?tab=repositories>

Цель работы

Целью данной работы является получение навыков управления процессами операционной системы.

Задание

- 1 Продemonстрируйте навыки управления заданиями операционной системы
- 2 Продemonстрируйте навыки управления процессами операционной системы
- 3 Выполните задания для самостоятельной работы

Раздел 2

Выполнение лабораторной работы

Управление заданиями

Для начала получим полномочия администратора su – и введём следующие команды:

```
sleep 3600 & dd if=/dev/zero of=/dev/null & sleep 7200
```

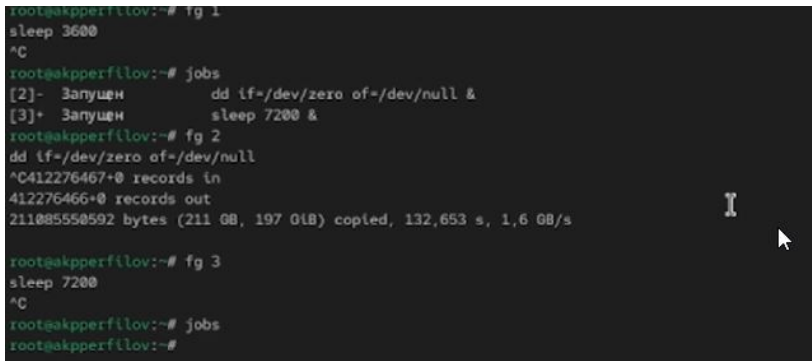
Поскольку мы запустили последнюю команду без & после неё, у нас есть 2 часа, прежде чем мы снова получим контроль над оболочкой. Введём Ctrl + z , чтобы остановить процесс. Затем введём jobs и увидим три задания, которые мы только что запустили. Первые два имеют состояние Running, а последнее задание в настоящее время находится в состоянии Stopped. Для продолжения выполнения задания 3 в фоновом режиме введём bg 3 и с помощью команды jobs посмотрим изменения в статусе заданий (

Управление заданиями

```
akpperfilov@akpperfilov:~$ su -
Пароль:
su: Сбой при проверке подлинности
akpperfilov@akpperfilov:~$ su -
Пароль:
Последний вход в систему: Пт окт  3 23:02:33 MSK 2025 на pts/0
Последняя неудачная попытка входа в систему: Вт окт  7 21:33:46 MSK 2025 на pts/0
Со времени последнего входа была 1 неудачная попытка.
root@akpperfilov:~# sleep 3600 &
[1] 4477
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &
[2] 4496
root@akpperfilov:~# sleep 7200
^Z
[3]+  Остановлен      sleep 7200
root@akpperfilov:~# jobs
[1]  Запущен          sleep 3600 &
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Остановлен      sleep 7200
root@akpperfilov:~# bg 3
[3]+  sleep 7200 &
root@akpperfilov:~# jobs
[1]  Запущен          sleep 3600 &
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Запущен          sleep 7200 &
```


Управление заданиями

Для перемещения задания 1 на передний план введём `fg 1`, далее введём `Ctrl+C`, чтобы отменить задание 1. С помощью команды `jobs` посмотрим изменения в статусе заданий и сделаем то же самое для отмены заданий 2 и 3




```
root@akpperfilov:~# fg 1
sleep 3600
^C
root@akpperfilov:~# jobs
[2]-  Запущен      dd if=/dev/zero of=/dev/null &
[3]+  Запущен      sleep 7200 &
root@akpperfilov:~# fg 2
dd if=/dev/zero of=/dev/null
^C412276467+0 records in
412276466+0 records out
211085550592 bytes (211 GB, 197 GiB) copied, 132.653 s, 1.6 GB/s

root@akpperfilov:~# fg 3
sleep 7200
^C
root@akpperfilov:~# jobs
root@akpperfilov:~#
```

Рис. 2: Перемещение заданий на передний план и их последующая отмена.

Управление заданиями

Теперь откроем второй терминал и под учётной записью пользователя введём в нём: `dd if=/dev/zero of=/dev/null &`. После введём `exit`, чтобы закрыть второй терминал



```
akpperfilov@akpperfilov:~$ dd if=/dev/zero of=/dev/null &  
[1] 4633  
akpperfilov@akpperfilov:~$ exit
```

Рис. 3: Ввод команды и закрытие терминала.

Управление заданиями

На другом терминале под учётной записью своего пользователя запустим `top`. Мы увидим, что задание `dd` всё ещё запущено. Для выхода из `top` используем `q` и вновь запускаем `top`, в нём используем `k`, чтобы убить задание `dd`. После этого выйдем из `top`

```
top - 21:40:11 up 12 min, 4 users, load average: 1.31, 1.83, 0.58
Tasks: 284 total, 2 running, 282 sleeping, 0 stopped, 0 zombie
MiB Mem: 6.6 av, 11.3 sw, 0.0 bu, 81.7 id, 0.0 us, 0.3 sy, 0.0 st, 0.0 sr
MiB Mem: 7678.5 total, 4455.5 free, 2196.6 used, 1431.6 buff/cache
MiB Swap: 3312.0 total, 3312.0 free, 0.0 used, 5571.9 avail Mem
```

PID	USER	PR	NI	VTST	RES	SHR	S	CPU	MEM	TIME+	COMMAND
4633	akpperf*	20	0	227388	2164	8	99	0	0	2154.25	dd
4215	akpperf*	20	0	2964588	361728	98236	5	7.3	4.0	0:00.44	ptx/xs
2269	akpperf*	20	0	1086148	307936	132596	5	4.7	5.1	1:24.30	gnome-shell
67	root	20	0	0	0	0	1	0.3	0.0	0:01.02	kwazex/z20:2-even*
728	root	20	0	0	0	0	1	0.3	0.0	0:00.02	kwazex/z29:2-even*
4716	root	20	0	231684	5500	3324	8	0.3	0.1	0:00.83	top
1	root	20	0	40192	41812	18148	5	0.0	0.5	0:01.04	systemd
2	root	20	0	0	0	0	5	0.0	0.0	0:00.01	kthread
3	root	20	0	0	0	0	5	0.0	0.0	0:00.00	pool_worikuew_rel*
4	root	0-20	0	0	0	0	1	0.0	0.0	0:00.00	kwazex/R-rcu_gp
5	root	0-20	0	0	0	0	1	0.0	0.0	0:00.00	kwazex/R-sync_wq
6	root	0-20	0	0	0	0	1	0.0	0.0	0:00.00	kwazex/R-slab_flu*
7	root	0-20	0	0	0	0	1	0.0	0.0	0:00.00	kwazex/R-netns
18	root	0-20	0	0	0	0	1	0.0	0.0	0:00.00	kwazex/0:0H-event*
11	root	20	0	0	0	0	1	0.0	0.0	0:00.00	kwazex/z24:0-even*
12	root	20	0	0	0	0	1	0.0	0.0	0:00.01	kwazex/z24:1-ipv6*
13	root	0-20	0	0	0	0	1	0.0	0.0	0:00.00	kwazex/R-se_percp*
14	root	20	0	0	0	0	1	0.0	0.0	0:00.00	rcu_tasks_kthread
15	root	20	0	0	0	0	1	0.0	0.0	0:00.00	rcu_tasks_rude_kth*
16	root	20	0	0	0	0	1	0.0	0.0	0:00.00	rcu_tasks_trace_kt*
17	root	20	0	0	0	0	5	0.0	0.0	0:00.00	ksoftirqd/0
18	root	20	0	0	0	0	1	0.0	0.0	0:00.20	rcu_greent
19	root	20	0	0	0	0	5	0.0	0.0	0:00.00	rcu_exp_par_gp_kth*
20	root	20	0	0	0	0	5	0.0	0.0	0:00.01	rcu_exp_gp_kthread*
21	root	rt	0	0	0	0	5	0.0	0.0	0:00.00	migration/0
22	root	-51	0	0	0	0	5	0.0	0.0	0:00.00	idle_inject/0
23	root	20	0	0	0	0	5	0.0	0.0	0:00.00	cpupd/0
24	root	20	0	0	0	0	5	0.0	0.0	0:00.00	cpupd/1
25	root	-51	0	0	0	0	5	0.0	0.0	0:00.00	idle_inject/1
26	root	rt	0	0	0	0	5	0.0	0.0	0:00.36	migration/1

Рис. 4: Убийство задания `dd` в `top`.

Управление процессами

Получим полномочия администратора su - и введём следующие команды:

```
dd if=/dev/zero of=/dev/null & dd if=/dev/zero of=/dev/null & dd if=/dev/zero of=/dev/null &
```

После чего введём `ps aux | grep dd`, которое показывает все строки, в которых есть буквы dd. Запущенные процессы dd идут последними. Используем PID первого процесса dd, чтобы изменить приоритет (`renice -n 5`)

```
root@akpperfilov:~# ps aux | grep dd
root      2  0.0  0.0      0   0 ?        S   21:27   0:00 [kthreadd]
root     12  0.0  0.0      0   0 ?        I   21:27   0:00 [kworker/u24:1-tpv6_addr
conf]
root    114  0.0  0.0      0   0 ?        I<  21:27   0:00 [kworker/R-ipv6_addrconf
]
akpperf+ 2771  0.0  0.3 1037064 26248 ?        Ssl  21:28   0:00 /usr/libexec/evolution-a
ddressbook-factory
akpperf+ 3556  0.0  0.4 260964 34792 ?        Sl   21:28   0:00 /usr/lib64/firefox/firef
ox -contentproc -parentBuildID 20250918202509 -prefsHandle 0:35214 -prefMapHandle 1:271119
-sandboxReporter 2 -chrootClient 3 -ipcHandle 4 -initialChannelId {f60f3ef7-e618-481b-aa2b-
2cf88708562c} -parentPid 3437 -appDir /usr/lib64/firefox/browser 3 rdd
akpperf+ 3697  4.3  3.0 2985700 236156 ?        Sl   21:28   0:33 /usr/lib64/firefox/firef
ox -contentproc -isForBrowser -prefsHandle 0:40970 -prefMapHandle 1:271119 -jsInitHandle 2:
242716 -parentBuildID 20250918202509 -sandboxReporter 3 -chrootClient 4 -ipcHandle 5 -initi
alChannelId {8318f7a5-7dd0-4bfe-8e1b-1e1962fc2447} -parentPid 3437 -greomni /usr/lib64/fire
fox/omni.is -appomni /usr/lib64/firefox/browser/omni.is -appDir /usr/lib64/firefox/browser
```

Управление процессами

Введём `ps fax | grep -B5 dd`. Параметр `-B5` показывает соответствующие запросу строки, включая пять строк до этого. Поскольку `ps fax` показывает иерархию отношений между процессами, мы также видим оболочку, из которой были запущены все процессы `dd`, и её PID

```
root@kali:~# ps fax | grep -B5 dd
PID TTY          STAT       TIME COMMAND
  2  ?        S          0:00 [kthreadd]

108 ?        I<         0:00 \. [kworker/R-acpi_thermal_pm]
109 ?        I<         0:00 \. [kworker/R-knpath_rtdcd]
110 ?        I<         0:00 \. [kworker/R-kaliud]
111 ?        I<         0:00 \. [kworker/R-mlq]
112 ?        I<         0:00 \. [kworker/4:llc-afa-log/ds-0]
114 ?        I<         0:00 \. [kworker/R-ipw6_addrconf]

2365 ?        Ssl        0:00 \. /usr/libexec/gvfsd
4004 ?        Sl         0:00 | \. /usr/libexec/gvfsd-trash --spawner :1.17 /org/gtk/gvfs/e
exec_spawn/0 4072 ?        Sl         0:00 | \. /usr/libexec/gvfsd-recent --spawner :1.17 /org/gtk/gvfs/
exec_spawn/1 4073 ?        Sl         0:00 | \. /usr/libexec/gvfsd-network --spawner :1.17 /org/gtk/gvfs
fexec_spawn/2 4086 ?        Sl         0:00 | \. /usr/libexec/gvfsd-dnssd --spawner :1.17 /org/gtk/gvfs/e
exec_spawn/3 4093 ?        Sl         0:00 | \. /usr/libexec/gvfsd-wsdd --spawner :1.17 /org/gtk/gvfs/ex
rc_spawn/4

2648 ?        Ssl        0:00 \. /usr/bin/gjs -m /usr/share/gnome-shell/org.gnome.Screenaver
2661 ?        Ssl        0:00 \. /usr/libexec/libx-portal
2726 ?        Ssl        0:00 \. /usr/libexec/evolution-calendar-factory
2743 ?        Ssl        0:00 \. /usr/libexec/goa-identity-service
2753 ?        Ssl        0:00 \. /usr/libexec/gvfs-mtp-volume-monitor
2771 ?        Ssl        0:00 \. /usr/libexec/evolution-addressbook-factory

3002 ?        Ssl        0:00 \. /usr/libexec/xdg-desktop-portal-gnome
3016 ?        Ssl        0:00 \. /usr/libexec/xdg-desktop-portal-gtk
3927 ?        Ssl        0:00 \. /usr/libexec/gvfsd-metadata
```

Рис. 6: Просмотр иерархии отношений между процессами.

Управление процессами

Теперь найдём PID корневой оболочки, из которой были запущены процессы dd, и введём `kill -9` (указав PID оболочки). Мы увидим, что наша корневая оболочка закрылась, а вместе с ней и все процессы dd (остановка родительского процесса — простой и удобный способ остановить все его дочерние процессы)



```
root@akpperfilov:~# kill -9 4419
```


Рис. 7: Закрытие корневой оболочки.

Раздел 3

Выполнение заданий для самостоятельной работы

Самостоятельная работа (задание 1)

Получим полномочия администратора `su -` и запустим команду `dd if=/dev/zero of=/dev/null &` трижды как фоновое задание. Затем увеличим приоритет первой команды, используя значение приоритета `-5`, после чего изменим приоритет того же процесса ещё раз, но используем на этот раз значение `-15` (мы можем менять приоритет команды от `-20` (самый высокий приоритет) до `19` (самый низкий приоритет)). Завершим все процессы `dd`, которые мы запустили командой: `killall dd`



```
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &  
[1] 5265  
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &  
[2] 5266  
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &  
[3] 5269
```

Рис. 8: Получение полномочий администратора, запуск команды трижды как фоновое задание.

Самостоятельная работа (задание 1)

```
root@akpperfilov:~# renice -n -5 5266  
5266 (process ID) old priority 0, new priority -5
```

Рис. 9: Увеличение приоритета первой команды.

Самостоятельная работа (задание 1)

```
root@akpperfilov:~# renice -n -15 5266  
5266 (process ID) old priority -5, new priority -15
```

Рис. 10: Увеличение приоритета первой команды.

Самостоятельная работа (задание 1)

```
root@akpperfilov:~# fg 1
dd if=/dev/zero of=/dev/null
^C296294751+0 records in
296294750+0 records out
151702912000 bytes (152 GB, 141 GiB) copied, 103,217 s, 1,5 GB/s

root@akpperfilov:~# fg 2
dd if=/dev/zero of=/dev/null
^C300621645+0 records in
300621645+0 records out
153918282240 bytes (154 GB, 143 GiB) copied, 107,674 s, 1,4 GB/s

root@akpperfilov:~# fg 3^C
root@akpperfilov:~# fg 3
dd if=/dev/zero of=/dev/null
^C316103869+0 records in
316103868+0 records out
161845180416 bytes (162 GB, 151 GiB) copied, 115,315 s, 1,4 GB/s

root@akpperfilov:~# jobs
root@akpperfilov:~#
```

Рис. 11: Завершение всех процессов.

Самостоятельная работа (задание 2)

Получим полномочия администратора su – и запустим программу yes в фоновом режиме с подавлением потока вывода, далее запустим программу yes на переднем плане с подавлением потока вывода и приостановим выполнение программы. Заново запустим программу yes с теми же параметрами, затем завершим её выполнение. Повторим действия, но уже запустим программу yes на переднем плане без подавления потока вывода. Также приостановим выполнение программы и заново запустим программу yes с теми же параметрами, затем завершим её выполнение. Проверим состояния заданий, воспользовавшись командой jobs. Далее переведём процесс, который у нас выполняется в фоновом режиме, на передний план, затем остановим его. Переведём 3 процесс с подавлением потока вывода в фоновый режим (bg 3) и проверим состояния заданий, воспользовавшись командой jobs. Запустим процесс в фоновом режиме таким образом, чтобы он продолжил свою работу даже после отключения от терминала. Закроем окно и заново запустим консоль. Убедимся, что процесс продолжил свою работу

Получение полномочий администратора. Запуск программы yes в фоновом режиме с

Самостоятельная работа (задание 2)

Сейчас получим информацию о запущенных в операционной системе процессах с помощью утилиты `top`

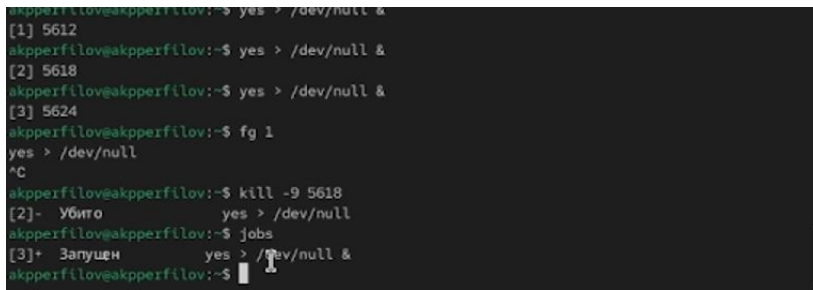
```
top - 22:01:54 up 34 min, 5 users, load average: 4.32, 4.24, 3.99
Tasks: 287 total, 0 running, 281 sleeping, 0 stopped, 0 zombie
MiB Mem: 20,9 us, 61,2 sy, 6,0 hi, 1,5 id, 0,0 ma, 1,5 hi, 0,0 si, 0,0 st
MiB Mem: 7670,8 total, 4385,9 free, 2174,5 used, 1432,7 buff/cache
MiB Swap: 3312,0 total, 3312,0 free, 0,0 used, 5583,9 avail Mem
```

PID	USER	PR	%C	%M	VSZ	RES	TIME+	TIME-	COMMAND		
4970	root	20	0	227308	3088	2088	R	100,0	0,0	20:29,25	dd
4972	root	20	0	227308	3088	2088	R	100,0	0,0	20:28,34	dd
4966	root	25	5	227308	2196	2196	R	90,9	0,0	20:24,90	dd
5295	root	20	0	227280	2128	2128	R	90,9	0,0	1:44,24	yes
5368	root	20	0	227280	2044	2044	R	90,0	0,0	0:22,43	yes
1	root	20	0	40192	41812	18140	S	0,0	0,5	0:01,09	synted
2	root	20	0	0	0	0	S	0,0	0,0	0:00,01	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00,00	pool_workqueue_releas
4	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	knockd/H-rcu_gp
5	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	knockd/H-sync_wq
6	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	knockd/H-slab_flush
7	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	knockd/H-netns
10	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	knockd/0-0H-events_*
11	root	20	0	0	0	0	I	0,0	0,0	0:00,00	knockd/u24:0-events_*
12	root	20	0	0	0	0	I	0,0	0,0	0:00,01	knockd/u24:1-ipv6_*
13	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	knockd/H-net_pcpu_*
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_kthre
16	root	20	0	0	0	0	I	0,0	0,0	0:00,00	rcu_tasks_trace_kthre
17	root	20	0	0	0	0	S	0,0	0,0	0:00,00	ksftirqd/0
18	root	20	0	0	0	0	I	0,0	0,0	0:00,03	rcu_preempt
19	root	20	0	0	0	0	S	0,0	0,0	0:00,00	rcu_exp_bar_gp_kthre
20	root	20	0	0	0	0	S	0,0	0,0	0:00,01	rcu_exp_gp_kthread_w
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
25	root	-51	0	0	0	0	S	0,0	0,0	0:00,00	idle_inject/1

Рис. 13: Получение информации о запущенных в операционной системе процессах.

Самостоятельная работа (задание 2)

Запустим ещё три программы `yes` в фоновом режиме с подавлением потока вывода (`yes > /dev/null &`). Убьём два процесса: для одного используем его PID (`kill -9`), а для другого — его идентификатор конкретного задания (`fg 2` и `Ctrl+c`). Попробуем послать сигнал 1 (`SIGHUP`) процессу, запущенному с помощью `nohup` (`kill -1`), и обычному процессу (`kill -1`)

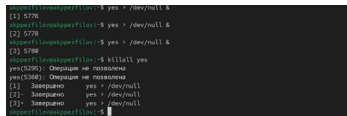


```
akpperfilov@akpperfilov:~$ yes > /dev/null &
[1] 5612
akpperfilov@akpperfilov:~$ yes > /dev/null &
[2] 5618
akpperfilov@akpperfilov:~$ yes > /dev/null &
[3] 5624
akpperfilov@akpperfilov:~$ fg 1
yes > /dev/null
^C
akpperfilov@akpperfilov:~$ kill -9 5618
[2]-  Убито                  yes > /dev/null
akpperfilov@akpperfilov:~$ jobs
[3]+  Запущен                yes > /dev/null &
akpperfilov@akpperfilov:~$
```

Рис. 14: Запуск трёх программ `yes` в фоновом режиме с подавлением потока вывода, убийство двух процессов, попытка послать сигнал 1 (`SIGHUP`).

Самостоятельная работа (задание 2)

Запустим ещё несколько программ `yes` в фоновом режиме с подавлением потока вывода (`yes > /dev/null &`) и завершим их работу одновременно, используя команду `killall yes`



```
alex@alex:~$ yes > /dev/null &
[1] 5776
alex@alex:~$ yes > /dev/null &
[2] 5778
alex@alex:~$ yes > /dev/null &
[3] 5780
alex@alex:~$ killall yes
yes(5295): Операция не позволена
yes(5368): Операция не позволена
[1]- Завершено   yes > /dev/null
[2]- Завершено   yes > /dev/null
[3]+ Завершено   yes > /dev/null
alex@alex:~$
```

Рис. 15: Запуск программ `yes` в фоновом режиме с подавлением потока вывода и одновременное завершение их работы.

Самостоятельная работа (задание 2)

После чего запустим программу `yes` в фоновом режиме с подавлением потока вывода (`yes > /dev/null &`). Используя утилиту `nice` (`nice -n 15 yes`), запустим программу `yes` с теми же параметрами и с приоритетом, большим на 5. Сравним абсолютные и относительные приоритеты у этих двух процессов (`ps -l | grep yes`). Используя утилиту `renice`, изменим приоритет у одного из потоков `yes` таким образом, чтобы у обоих потоков приоритеты были равны (`renice -n 15`)

```

skperfilov@skperfilov:~$ yes > /dev/null &
[1] 5792
skperfilov@skperfilov:~$ nice -n 5 yes > /dev/null &
[2] 5796
skperfilov@skperfilov:~$ ps -l | grep yes
#  # 1000  5792  4296  98  00  0 - 56528 - pts/0 00:01:22 yes
#  # 1000  5796  4296  95  00  0 - 56528 - pts/0 00:00:25 yes
skperfilov@skperfilov:~$ renice -n 5 4296
4296 (process ID) old priority 0, new priority 5
skperfilov@skperfilov:~$ ps -l | grep yes
#  # 1000  5792  4296  98  00  0 - 56528 - pts/0 00:02:13 yes
#  # 1000  5796  4296  95  00  0 - 56528 - pts/0 00:01:13 yes
skperfilov@skperfilov:~$ renice -n 5 5792
5792 (process ID) old priority 0, new priority 5
skperfilov@skperfilov:~$

```

Рис. 16: Запуск программы `yes` в фоновом режиме с подавлением потока вывода. Запуск программы `yes` с теми же параметрами и с приоритетом, большим на 5. Сравнение абсолютных и относительных приоритетов, изменение приоритета.

Выводы

В ходе выполнения лабораторной работы были получены навыки управления процессами операционной системы.