

# **Лабораторная работа №6**

**Управление процессами**

Юсупова Ксения Равиловна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Ответы на контрольные вопросы</b>	<b>14</b>
<b>5</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

3.1	Выполнили пункты 1-8 из раздела 6.4.1. (Управление заданиями) .	7
3.2	Выполнили пункты 9 и 10 из раздела 6.4.1. (Управление заданиями)	8
3.3	Выполнили пункты 11 и 12 из раздела 5.4.1 (Управление сервисами)	8
3.4	Выполнили пункты 1-4 из раздела 6.4.2. (Управление процессами)	9
3.5	Выполнили пункты 5 и 6 из раздела 6.4.2. (Управление процессами)	9
3.6	Выполнили пункты 1-4 из раздела 6.5 (Самостоятельная работа) и 6.5.1. (Задание 1) . . . . .	10
3.7	Выполнили пункты 1-3 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2) . . . . .	10
3.8	Выполнили пункты 4-9 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2) . . . . .	11
3.9	Выполнили пункты 10 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2) . . . . .	12
3.10	Выполнили пункты 11-13 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2) . . . . .	12
3.11	Выполнили пункты 14-17 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2) . . . . .	13

## **Список таблиц**

# **1 Цель работы**

Получить навыки управления процессами операционной системы.

## 2 Задание

1. Продемонстрируйте навыки управления заданиями операционной системы (см. раздел 6.4.1).
2. Продемонстрируйте навыки управления процессами операционной системы (см. раздел 6.4.2).
3. Выполните задания для самостоятельной работы (см. раздел 6.5)

### 3 Выполнение лабораторной работы

Получили полномочия администратора, ввели команды и поскольку запустили последнюю команду без & после неё, у вас есть 2 часа, прежде чем вы снова получите контроль над оболочкой. Ввели Ctrl + z , чтобы остановить процесс и jobs, для продолжения выполнения задания 3 в фоновом режиме введите bg 3. С помощью команды jobs посмотрели изменения в статусе заданий. Для перемещения задания 1 на передний план ввели fg 1 и проделали то же самое для отмены заданий 2 и 3.(рис. 3.1).

```
[ksyusha@yu ~]$ su -
Пароль:
[root@yu ~]# sleep 3600 &
[1] 3064
[root@yu ~]# dd if=/dev/zero of=/dev/null &
[2] 3066
[root@yu ~]# sleep 7200
^Z
[3]+  Остановлен      sleep 7200
[root@yu ~]# jobs
[1]  Запущен          sleep 3600 &
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Остановлен      sleep 7200
[root@yu ~]# fg 1
sleep 3600
^C
[root@yu ~]# jobs
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Остановлен      sleep 7200
[root@yu ~]# fg 2
dd if=/dev/zero of=/dev/null
^C282494311+0 записей получено
282494311+0 записей отправлено
144637087232 байт (145 GB, 135 GiB) скопирован, 248,46 s, 582 MB/s

[root@yu ~]# fg 3
sleep 7200
^C
[root@yu ~]# jobs
```

Рис. 3.1: Выполнили пункты 1-8 из раздела 6.4.1. (Управление заданиями)

Открыли второй терминал и под учётной записью своего пользователя и ввели

exit, чтобы закрыть второй терминал.(рис. 3.2).

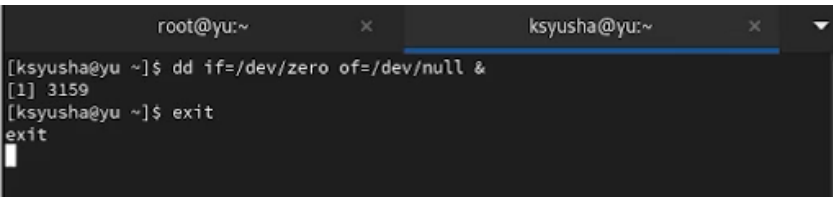


Рис. 3.2: Выполнили пункты 9 и 10 из раздела 6.4.1. (Управление заданиями)

На другом терминале под учётной записью своего пользователя запустили top и увидели, что задание dd всё ещё запущено. Вновь запустили top и в нём использовали k, чтобы убить задание dd. После этого вышли из top.(рис. 3.3).

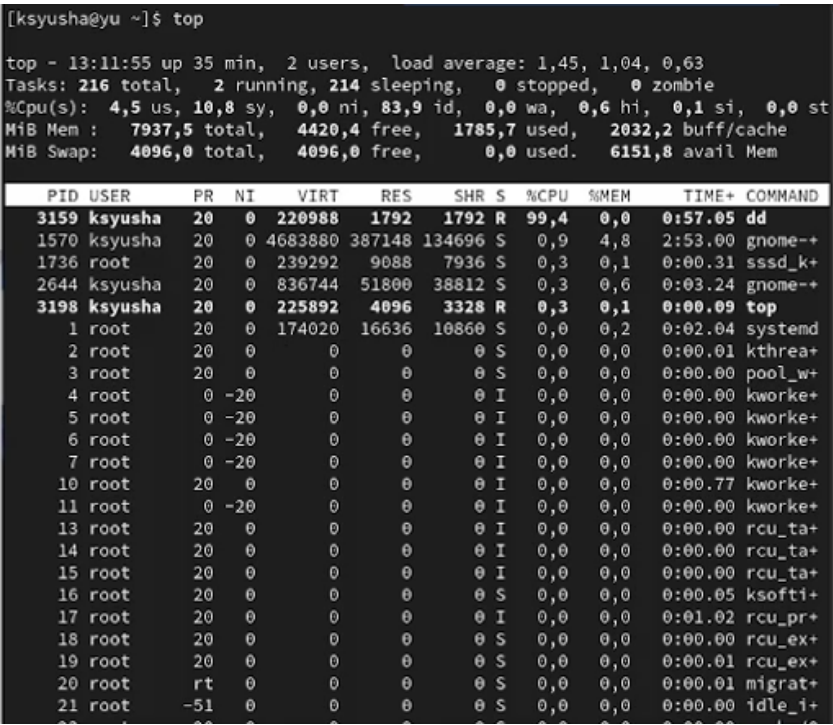


Рис. 3.3: Выполнили пункты 11 и 12 из раздела 5.4.1 (Управление сервисами)

Получили полномочия администратора и ввели команды; увидели, что запущенные процессы dd идут последними. Использовали PID одного из процессов dd, чтобы изменить приоритет.(рис. 3.4).



```

[root@yu ~]# dd if=/dev/zero of=/dev/null &
[1] 3258
[root@yu ~]# dd if=/dev/zero of=/dev/null &
[2] 3259
[root@yu ~]# dd if=/dev/zero of=/dev/null &
[3] 3260
[root@yu ~]# ps aux | grep dd
root      2  0.0  0.0      0   0 ?        S   12:36   0:00 [kthreadd]
ksyusha   1724  0.0  0.3 955596 30536 ?        Ssl  12:37   0:00 /usr/libexec/evolution-addressbook-factory
ksyusha   3159 98.4  0.0 220988 1792 ?        R   13:10   3:09 dd if=/dev/zero of=/dev/null
root      3258 68.6  0.0 220988 1792 pts/0    R   13:13   0:08 dd if=/dev/zero of=/dev/null
root      3259 60.0  0.0 220988 1792 pts/0    R   13:13   0:06 dd if=/dev/zero of=/dev/null
root      3260 57.8  0.0 220988 1792 pts/0    R   13:14   0:05 dd if=/dev/zero of=/dev/null
root      3265  0.0  0.0 221820 2432 pts/0    S+  13:14   0:00 grep --color=auto dd
[root@yu ~]# renice -n 5 3258
3258 (process ID) old priority 0, new priority 5

```

Рис. 3.4: Выполнили пункты 1-4 из раздела 6.4.2. (Управление процессами)

Введите команду, которая показала, что параметр -B5 показывает соответствующие запросу строки, включая пять строк до этого. Поскольку ps fax показывает иерархию отношений между процессами, также увидели оболочку, из которой были запущены все процессы dd, и её PID. Нашли PID корневой оболочки, из которой были запущены процессы dd, увидели, что корневая оболочка закрылась, а вместе с ней и все процессы dd.(рис. 3.5).

```

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

1688 ?          Ssl    0:00 \_ /usr/libexec/gvfs-gphoto2-volume-monitor
1696 ?          Ssl    0:00 \_ /usr/libexec/goa-identity-service
1703 ?          Ssl    0:00 \_ /usr/libexec/gvfs-goa-volume-monitor
1706 ?          Ssl    0:00 \_ /usr/libexec/evolution-calendar-factory
1723 ?          Ssl    0:00 \_ /usr/libexec/dconf-service
1724 ?          Ssl    0:00 \_ /usr/libexec/evolution-addressbook-factory

2155 ?          Ssl    0:00 \_ /usr/libexec/gvfsd-metadata
2644 ?          Ssl    0:05 \_ /usr/libexec/gnome-terminal-server
2676 pts/0      Ss     0:00 | \_ bash
3017 pts/0      S       0:00 | | \_ su -
3028 pts/0      S       0:00 | | | \_ -bash
3258 pts/0      RN     0:58 | | | \_ dd if=/dev/zero of=/dev/null
3259 pts/0      R      1:08 | | | \_ dd if=/dev/zero of=/dev/null
3260 pts/0      R      1:06 | | | \_ dd if=/dev/zero of=/dev/null
3285 pts/0      R+     0:00 | | | \_ ps fax
3286 pts/0      S+     0:00 | | | \_ grep --color=auto -B5 dd
3170 pts/2      Ss+    0:00 | | \_ bash
3203 pts/3      Ss+    0:00 | | \_ bash
3159 ?          R      4:41 | \_ dd if=/dev/zero of=/dev/null

```

Рис. 3.5: Выполнили пункты 5 и 6 из раздела 6.4.2. (Управление процессами)

Запустите команду `dd if=/dev/zero of=/dev/null` трижды как фоновое задание. Изменили приоритет одной из этих команд, используя значение приоритета -5. Изменили приоритет того же процесса ещё раз, но используйте на этот раз значение -15. Завершили все процессы `dd`, которые запустили.(рис. 3.6).

```
[root@yu ~]# dd if=/dev/zero of=/dev/null &
[1] 3480
[root@yu ~]# dd if=/dev/zero of=/dev/null &
[2] 3483
[root@yu ~]# dd if=/dev/zero of=/dev/null &
[3] 3484
[root@yu ~]# renice -n -5 3480
3480 (process ID) old priority 0, new priority -5
[root@yu ~]# renice -n -15 3480
3480 (process ID) old priority -5, new priority -15
[root@yu ~]# fg 1
dd if=/dev/zero of=/dev/null
^C36190277+0 записей получено
36190276+0 записей отправлено
18529421312 байт (19 GB, 17 GiB) скопирован, 107,095 s, 173 MB/s
```

Рис. 3.6: Выполнили пункты 1-4 из раздела 6.5 (Самостоятельная работа) и 6.5.1. (Задание 1)

Запустили программу `yes` в фоновом режиме с подавлением потока вывода и программу `yes` на переднем плане с подавлением потока вывода. Приостановили выполнение программы. Заново запустили программу `yes` с теми же параметрами, затем завершили её выполнение. Затем запустили программу `yes` на переднем плане без подавления потока вывода. Приостановили выполнение программы. Заново запустили программу `yes` с теми же параметрами, затем завершили её выполнение.(рис. 3.7).

```
[root@yu ~]# yes > /dev/null &
[1] 3557
[root@yu ~]# yes > /dev/null
^Z
[2]+  Остановлен    yes > /dev/null
[root@yu ~]# yes > /dev/null
^C
[root@yu ~]# yes
```

Рис. 3.7: Выполнили пункты 1-3 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2)

Проверили состояния заданий, воспользовавшись командой `jobs`, перевели

процесс, который у вас выполняется в фоновом режиме, на передний план, затем остановите его. Перевели процесс с подавлением потока вывода в фоновый режим и проверили состояния заданий, воспользовавшись командой `jobs`. Обратили внимание, что процесс стал выполняющимся (Running) в фоновом режиме. Запустили процесс в фоновом режиме таким образом, чтобы он продолжил свою работу даже после отключения от терминала. Закрыли окно и заново запустили консоль. Убелись, что процесс продолжил свою работу.(рис. 3.8).

```
[root@yu ~]# jobs
[1]  Запущен      yes > /dev/null &
[2]- Остановлен  yes > /dev/null
[3]+ Остановлен  yes
[root@yu ~]# fg 1
yes > /dev/null
^Z
[1]+  Остановлен  yes > /dev/null
[root@yu ~]# bg 2
[2] yes > /dev/null &
[root@yu ~]# jobs
[1]+  Остановлен  yes > /dev/null
[2]  Запущен      yes > /dev/null &
[3]- Остановлен  yes
[root@yu ~]# nohup yes > /dev/null &
[4] 3637
[root@yu ~]# nohup: ввод игнорируется и поток ошибок перенаправляется на стандартный вывод
exit
выход
```

Рис. 3.8: Выполнили пункты 4-9 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2)

Получили информацию о запущенных в операционной системе процессах с помощью утилиты `top`.(рис. 3.9).

```
top - 13:40:11 up 1:04, 2 users, load average: 8,77, 8,11, 6,40
Tasks: 221 total, 10 running, 210 sleeping, 1 stopped, 0 zombie
%Cpu(s): 28,3 us, 69,0 sy, 0,5 ni, 0,0 id, 0,0 wa, 2,2 hi, 0,0 si, 0,0 st
MiB Mem : 7937,5 total, 4453,1 free, 1752,4 used, 2030,7 buff/cache
MiB Swap: 4096,0 total, 4096,0 free, 0,0 used. 6185,1 avail Mem
```

Unknown command - try 'h' for help

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3159	ksyusha	20	0	220988	1792	1792	R	98,3	0,0	28:42.16	dd
3348	ksyusha	5	-15	220988	1792	1792	R	98,3	0,0	18:11.40	dd
3260	root	20	0	220988	1792	1792	R	15,6	0,0	8:21.21	dd
3637	root	20	0	220948	1664	1664	R	15,6	0,0	0:09.97	yes
3259	root	20	0	220988	1792	1792	R	15,2	0,0	8:24.25	dd
3353	ksyusha	20	0	220988	1792	1792	R	15,2	0,0	4:06.10	dd
3358	ksyusha	20	0	220988	1792	1792	R	15,2	0,0	4:04.40	dd
3564	root	20	0	220948	1664	1664	R	15,2	0,0	0:43.52	yes
3258	root	25	5	220988	1792	1792	R	5,0	0,0	3:24.42	dd
1570	ksyusha	20	0	4683648	391356	134588	S	1,0	4,8	4:03.20	gnome-+
2644	ksyusha	20	0	837684	54956	38640	S	0,3	0,7	0:10.59	gnome-+
1	root	20	0	174020	16636	10860	S	0,0	0,2	0:02.07	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthrea+
3	root	20	0	0	0	0	S	0,0	0,0	0:00.00	pool_w+
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworke+
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworke+
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworke+
7	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworke+
10	root	20	0	0	0	0	I	0,0	0,0	0:00.96	kworke+
11	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworke+
13	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_ta+
14	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_ta+
15	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_ta+

Рис. 3.9: Выполнили пункты 10 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2)

Запустили ещё три программы `yes` в фоновом режиме с подавлением потока вывода. Убили два процесса: для одного с помощью его PID, а для другого — его идентификатором конкретного задания. Послали сигнал 1 (SIGHUP) процессу, запущенному с помощью `nohup`, и обычному процессу.(рис. 3.10).

```
[ksyusha@yu ~]$ yes > /dev/null &
[1] 3782
[ksyusha@yu ~]$ yes > /dev/null &
[2] 3787
[ksyusha@yu ~]$ yes > /dev/null &
[3] 3792
[ksyusha@yu ~]$ fg 1
yes > /dev/null
^C
[ksyusha@yu ~]$ kill -9 3787
[2]- Убито yes > /dev/null
[ksyusha@yu ~]$ ps aux | grep yes
root      3564 23.4  0.0 220948 1664 pts/0    R   13:33   5:20 yes
root      3637 27.6  0.0 220948 1664 pts/0    R   13:39   4:45 yes
ksyusha   3792 28.9  0.0 220948 1664 pts/0    R   13:46   2:55 yes
ksyusha   3924  0.0  0.0 221688 2304 pts/0    S+  13:56   0:00 grep --color
=auto yes
[ksyusha@yu ~]$ sudo kill -1 3637
[ksyusha@yu ~]$ sudo kill -1 3792
[3]+ Обрыв терминальной линии yes > /dev/null
```

Рис. 3.10: Выполнили пункты 11-13 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2)

Запустили ещё несколько программ `yes` в фоновом режиме с подавлением потока вывода. Завершили их работу одновременно, используя команду `killall`. Запустили программу `yes` в фоновом режиме с подавлением потока вывода. Используя утилиту `nice`, запустили программу `yes` с теми же параметрами и с приоритетом, большим на 5. Сравнили абсолютные и относительные приоритеты у этих двух процессов. Используя утилиту `renice`, изменили приоритет у одного из потоков `yes` таким образом, чтобы у обоих потоков приоритеты были равны. (рис. 3.11).

```
[ksyusha@yu ~]$ yes > /dev/null &
[1] 3969
[ksyusha@yu ~]$ yes > /dev/null &
[2] 3974
[ksyusha@yu ~]$ yes > /dev/null &
[3] 3979
[ksyusha@yu ~]$ killall yes
yes(3564): Операция не позволена
yes(3637): Операция не позволена
[3]+ Завершено      yes > /dev/null
[1]- Завершено      yes > /dev/null
[2]+ Завершено      yes > /dev/null
[ksyusha@yu ~]$ yes > /dev/null &
[1] 4007
[ksyusha@yu ~]$ nice -n 5 yes > /dev/null &
[2] 4020
[ksyusha@yu ~]$ ps -l | grep yes
0 R 1000  4007  2676 35 80  0 - 55237 - pts/0  00:01:10 yes
0 R 1000  4020  2676 11 85  5 - 55237 - pts/0  00:00:16 yes
[ksyusha@yu ~]$ renice -n 5 4007
4007 (process ID) old priority 0, new priority 5
[ksyusha@yu ~]$ ps -l | grep yes
0 R 1000  4007  2676 32 85  5 - 55237 - pts/0  00:01:30 yes
0 R 1000  4020  2676 11 85  5 - 55237 - pts/0  00:00:25 yes
[ksyusha@yu ~]$
```

Рис. 3.11: Выполнили пункты 14-17 из раздела 6.5 (Самостоятельная работа) и 6.5.2 (Задание 2)

## 4 Ответы на контрольные вопросы

1. Команда `jobs` показывает все текущие задания оболочки.
2. Остановить задание для фонового выполнения можно комбинацией `Ctrl+Z`.
3. Для отмены текущего задания оболочки используется комбинация `Ctrl+C`.
4. Если нет доступа к оболочке, задание можно отменить командой `kill` с идентификатором процесса.
5. Команда `ps tree` отображает отношения между родительскими и дочерними процессами.
6. Изменить приоритет процесса 1234 на более высокий можно командой `renice -n -5 1234`.
7. Чтобы остановить все 20 процессов `dd`, проще всего использовать `killall dd`.
8. Остановить команду `mysommand` можно командой `kill mysommand`.
9. В утилите `top` для завершения процесса используется клавиша 'к'.
10. Для запуска команды с высоким приоритетом без риска для системы используется `nice -n -10 command`.

## **5 Выводы**

В ходе лабораторной работы мы получили навыки управления процессами операционной системы.