

# **Лабораторная работа №6. Управление процессами**

**Дисциплина: Администрирование операционных систем**

Жукова Арина Александровна

# 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>
3.1	Управление заданиями . . . . .	7
3.2	Управление процессами . . . . .	9
<b>4</b>	<b>Самостоятельная работа</b>	<b>12</b>
4.1	Задание 1 . . . . .	12
4.2	Задание 2 . . . . .	13
<b>5</b>	<b>Контрольные вопросы</b>	<b>17</b>
<b>6</b>	<b>Выводы</b>	<b>19</b>
	<b>Список литературы</b>	<b>20</b>

# Список иллюстраций

3.1	Запускаю задания . . . . .	7
3.2	Работа команды jobs, bg 3 . . . . .	8
3.3	Перенос задания 1 на передний план . . . . .	8
3.4	Перенос на передний план . . . . .	8
3.5	Работа команды . . . . .	9
3.6	Утилита top . . . . .	9
3.7	Запуск задания . . . . .	9
3.8	Просмотр строк с dd . . . . .	10
3.9	Установка приоритета . . . . .	10
3.10	Просмотр иерархии процессов . . . . .	10
3.11	Удаление всех процессов . . . . .	11
4.1	Выполнение задания 1 . . . . .	12
4.2	Выполнение части задания 2 . . . . .	13
4.3	Запуск процесса в фоне . . . . .	13
4.4	Работа команды jobs . . . . .	14
4.5	Утилита top . . . . .	14
4.6	Работа команд kill -9, kill -1 . . . . .	15
4.7	Удаление программ одновременно . . . . .	15
4.8	Установка приоритета . . . . .	16

## **Список таблиц**

# **1 Цель работы**

Получить навыки управления процессами операционной системы.

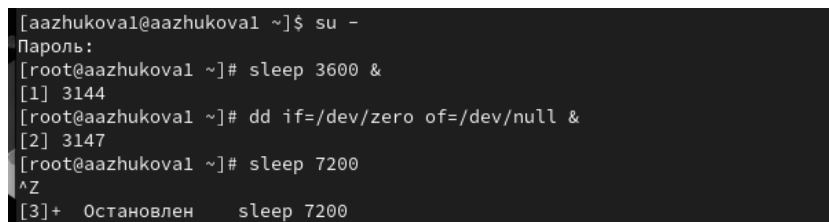
## 2 Задание

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

### 3.1 Управление заданиями

1. Получаю полномочия администратора. Ввожу следующие команды: `sleep 3600 & dd if=/dev/zero of=/dev/null & sleep 7200` (рис. 3.1).



```
[aazhukoval@aazhukoval ~]$ su -
Пароль:
[root@aazhukoval ~]# sleep 3600 &
[1] 3144
[root@aazhukoval ~]# dd if=/dev/zero of=/dev/null &
[2] 3147
[root@aazhukoval ~]# sleep 7200
^Z
[3]+  Остановлен      sleep 7200
```

Рис. 3.1: Запускаю задания

Поскольку я запустила последнюю команду без `&` после неё, у меня есть 2 часа, прежде чем я снова получу контроль над оболочкой. Ввожу `Ctrl + z`, чтобы остановить процесс.

2. Ввожу `jobs`. Вижу три задания, которые я только что запустила. Первые два имеют состояние `Running`, а последнее задание в настоящее время находится в состоянии `Stopped`. Для продолжения выполнения задания 3 в фоновом режиме ввожу `bg` 3. С помощью команды `jobs` смотрю изменения в статусе заданий (рис. 3.2).

```

[3]- Остановлен      sleep 7200
[root@aazhukoval ~]# jobs
[1]  Запущен          sleep 3600 &
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Остановлен      sleep 7200
[root@aazhukoval ~]# bg 3
[3]+ sleep 7200 &
[root@aazhukoval ~]# jobs
[1]  Запущен          sleep 3600 &
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Запущен          sleep 7200 &
[root@aazhukoval ~]#

```

Рис. 3.2: Работа команды jobs, bg 3

3. Для перемещения задания 1 на передний план ввожу fg 1. Ввожу Ctrl + c, чтобы отменить задание 1. С помощью команды jobs смотрю изменения в статусе заданий (рис. 3.3).

```

[3]- Остановлен      sleep 7200 &
[root@aazhukoval ~]# fg 1
sleep 3600
^C
[root@aazhukoval ~]# jobs
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Запущен          sleep 7200 &
[root@aazhukoval ~]#

```

Рис. 3.3: Перенос задания 1 на передней план

4. Проделываю то же самое для отмены заданий 2 и 3 (рис. 3.4).

```

[root@aazhukoval ~]# fg 2
dd if=/dev/zero of=/dev/null
^C504363974+0 записей получено
504363974+0 записей отправлено
258234354688 байт (258 GB, 240 GiB) скопирован, 241,271 s, 1,1 GB/s

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

```

Рис. 3.4: Перенос на передний план

5. Открываю второй терминал и под учётной записью своего пользователя ввожу в нём: dd if=/dev/zero of=/dev/null &. Ввожу exit, чтобы закрыть второй терминал (рис. 3.5).



```
[aazhukoal@aazhukoal ~]$ dd if=/dev/zero of=/dev/null &
[1] 3228
[aazhukoal@aazhukoal ~]$ exit
```

Рис. 3.5: Работа команды

- На другом терминале под учётной записью своего пользователя запускаю `top`. Вижу, что задание `dd` всё ещё запущено. Для выхода из `top` использую `q` (рис. 3.6).

```
top - 19:46:32 up 16 min, 2 users, load average: 0,86, 0,63, 0,37
Tasks: 209 total, 2 running, 207 sleeping, 0 stopped, 0 zombie
%Cpu(s): 13,7 us, 27,7 sy, 0,0 ni, 56,8 id, 0,0 wa, 1,8 hi, 0,0 si, 0,0 st
MiB Mem : 3659,7 total, 1203,0 free, 1253,8 used, 1457,8 buff/cache
MiB Swap: 4044,0 total, 4044,0 free, 0,0 used, 2405,9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3228	aazhuko+	20	0	220988	1792	1792	R	97,0	0,0	1:38.17	dd
2179	aazhuko+	20	0	4112204	382508	125768	S	4,7	10,2	0:12.21	gnome-s+
2273	aazhuko+	9	-11	327476	13620	9216	S	0,3	0,4	0:00.10	pipewire
2275	aazhuko+	9	-11	326956	12136	8264	S	0,3	0,3	0:00.03	pipewir+
2732	aazhuko+	20	0	431308	3076	2688	S	0,3	0,1	0:01.51	VBoxCli+
3320	aazhuko+	20	0	225884	4096	3328	R	0,3	0,1	0:00.05	top
1	root	20	0	173132	16548	10800	S	0,0	0,4	0:01.03	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthreadd
3	root	0	-20	0	0	0	T	0,0	0,0	0:00.00	rcu_gp

Рис. 3.6: Утилита `top`

Вновь запускаю `top` и в нём использую `k`, чтобы убить задание `dd`. После этого выхожу из `top`.

## 3.2 Управление процессами

- Получаю полномочия администратора. Ввожу следующую команду 3 раза `dd if=/dev/zero of=/dev/null &` (рис. 3.7).

```
[aazhukoal@aazhukoal ~]$ su -
Пароль:
[root@aazhukoal ~]# dd if=/dev/zero of=/dev/null &
[1] 3443
[root@aazhukoal ~]# dd if=/dev/zero of=/dev/null &
[2] 3444
[root@aazhukoal ~]# dd if=/dev/zero of=/dev/null &
[3] 3445
```

Рис. 3.7: Запуск задания

2. Ввожу `ps aux | grep dd`. Это команда показывает все строки, в которых есть буквы `dd`. Запущенные процессы `dd` идут последними (рис. 3.8).

```
[root@aazhukoval ~]# ps aux | grep dd
root      2  0.0  0.0   0   0 ?        S   19:30   0:00 [kthreadd]
root     66  0.0  0.0   0   0 ?        I<  19:30   0:00 [ipv6_addrconf]
root    1049  0.0  0.1 508552 3840 ?        Ssl 19:30   0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-
service.sh
aazhuko+ 2325  0.0  0.8 880724 30364 ?        Ssl 19:31   0:00 /usr/libexec/evolution-addressbook-factory
aazhuko+ 2731  0.0  0.0 232496 1412 ?        S   19:31   0:00 /usr/bin/VBoxClient --dragnaddrop
aazhuko+ 2732  0.1  0.0 431308 3076 ?        Ssl 19:31   0:01 /usr/bin/VBoxClient --dragnaddrop
root    3443 70.9  0.0 220988 1792 pts/0    R   19:48   0:13 dd if=/dev/zero of=/dev/null
root    3444 68.4  0.0 220988 1792 pts/0    R   19:48   0:11 dd if=/dev/zero of=/dev/null
root    3445 63.6  0.0 220988 1792 pts/0    R   19:48   0:10 dd if=/dev/zero of=/dev/null
root    3453  0.0  0.0 221820 2432 pts/0    S+  19:48   0:00 grep --color=auto dd
[root@aazhukoval ~]#
```

Рис. 3.8: Просмотр строк с `dd`

3. Использую PID одного из процессов `dd`, чтобы изменить приоритет. Использую `renice -n 5` (рис. 3.9).

```
[root@aazhukoval ~]# renice -n 5 3445
3445 (process ID) old priority 0, new priority 5
[root@aazhukoval ~]#
```

Рис. 3.9: Установление приоритета

4. Ввожу `ps fax | grep -B5 dd`. Параметр `-B5` показывает соответствующие запросу строки, включая пять строк до этого. Поскольку `ps fax` показывает иерархию отношений между процессами, я также увижу оболочку, из которой были запущены все процессы `dd`, и её PID (рис. 3.10).

```
[root@aazhukoval ~]# ps fax | grep -B5 dd
PID TTY STAT TIME COMMAND
--
55 ? I< 0:00 \_ [kthrotld]
60 ? I< 0:00 \_ [acpi_thermal_pm]
61 ? I< 0:00 \_ [kmpath_rdacd]
62 ? I< 0:00 \_ [kaluad]
65 ? I< 0:00 \_ [mld]
66 ? I< 0:00 \_ [ipv6_addrconf]
--
783 ? S 0:00 /usr/sbin/chronyd -F 2
787 ? Sns 0:00 /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --initfile=/l
b/alsa/init/00main rdaemon
800 ? Ssl 0:00 /usr/sbin/ModemManager
803 ? Ssl 0:00 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
1047 ? Sl 0:00 /usr/bin/VBoxDRMClient
1049 ? Sl 0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
--
2274 ? S<sl 0:00 \_ /usr/bin/wireplumber
2275 ? S<sl 0:00 \_ /usr/bin/pipewire-pulse
```

Рис. 3.10: Просмотр иерархии процессов

5. Нахожу PID корневой оболочки, из которой были запущены процессы `dd`, и ввожу `kill -9` (заменив на значение PID оболочки). Вижу, что моя корневая оболочка закрылась, а вместе с ней и все процессы `dd`. (рис. 3.11).

```
kill -9 3413  
[root@aazhukoval ~]# kill -9 3413  
Убито
```

Рис. 3.11: Удаление всех процессов

## 4 Самостоятельная работа

### 4.1 Задание 1

1. Запускаю команду `dd if=/dev/zero of=/dev/null` трижды как фоновое задание. Увеличиваю приоритет одной из этих команд, используя значение приоритета `-5`. Изменяю приоритет того же процесса ещё раз, но использую на этот раз значение `-15`. Завершаю все процессы `dd`, которые я запустила (рис. 4.1).

```
[aazhukoval@aazhukoval ~]$ su -  
Пароль:  
[root@aazhukoval ~]# dd if=/dev/zero of=/dev/null &  
[1] 44795  
[root@aazhukoval ~]# dd if=/dev/zero of=/dev/null &  
[2] 44796  
[root@aazhukoval ~]# dd if=/dev/zero of=/dev/null &  
[3] 44797  
[root@aazhukoval ~]# renice -n 5 dd  
renice: bad process ID value: dd  
[root@aazhukoval ~]# renice -n 5 44797  
44797 (process ID) old priority 0, new priority 5  
[root@aazhukoval ~]# renice -n 15 44797  
44797 (process ID) old priority 5, new priority 15  
[root@aazhukoval ~]# killall dd  
[1]  Завершено      dd if=/dev/zero of=/dev/null  
[2]-  Завершено      dd if=/dev/zero of=/dev/null  
[3]+  Завершено      dd if=/dev/zero of=/dev/null  
[root@aazhukoval ~]#
```

Рис. 4.1: Выполнение задания 1

## 4.2 Задание 2

1. Запускаю программу `yes` в фоновом режиме с подавлением потока вывода `yes > /dev/null &`. Запускаю программу `yes` на переднем плане с подавлением потока вывода, приостанавливаю выполнение программы. Заново запускаю программу `yes` с теми же параметрами, затем завершаю её выполнение. Запускаю программу `yes` на переднем плане без подавления потока вывода `yes > /dev/null`. Приостанавливаю выполнение программы. Заново запускаю программу `yes` с теми же параметрами, затем завершаю её выполнение. Проверяю состояния заданий, воспользовавшись командой `jobs`. Перевожу процесс, который у меня выполняется в фоновом режиме, на передний план, затем останавливаю его (`fg 1`, после чего `Ctrl+C`). Перевожу третий процесс с подавлением потока вывода в фоновый режим. Проверяю состояния заданий, воспользовавшись командой `jobs`. Обращаю внимание, что процесс стал выполняющимся (Running) в фоновом режиме (рис. 4.2).

```
[root@aazhukoval ~]# yes > /dev/null &
[1] 44823
[root@aazhukoval ~]# yes > /dev/null
^Z
[2]+  Остановлен   yes > /dev/null
[root@aazhukoval ~]# jobs
[1]-  Запущен     yes > /dev/null &
[2]+  Остановлен   yes > /dev/null
[root@aazhukoval ~]# fg 1
yes > /dev/null
^C
[root@aazhukoval ~]# yes > /dev/null
^Z
[3]+  Остановлен   yes > /dev/null
[root@aazhukoval ~]# bg 3
[3]+  yes > /dev/null &
[root@aazhukoval ~]# jobs
[2]+  Остановлен   yes > /dev/null
[3]-  Запущен     yes > /dev/null &
```

Рис. 4.2: Выполнение части задания 2

2. Запускаю процесс в фоновом режиме таким образом, чтобы он продолжил свою работу даже после отключения от терминала (рис. 4.3).

```
[root@aazhukoval ~]# nohup yes > /dev/null &
[4] 44831
[root@aazhukoval ~]# nohup: ввод игнорируется и поток ошибок перенаправляется на стандартный вывод
```

Рис. 4.3: Запуск процесса в фоне

3. Закрываю окно и заново запускаю консоль. Убеждаюсь, что процесс продолжил свою работу (рис. 4.4).

```
[root@aazhukoval ~]# jobs
[2]+  Остановлен      yes > /dev/null
[3]   Запущен        yes > /dev/null &
[4]-  Запущен        nohup yes > /dev/null &
```

Рис. 4.4: Работа команды jobs

4. Получаю информацию о запущенных в операционной системе процессах с помощью утилиты top (рис. 4.5).

```
top - 20:25:39 up 18 min,  2 users,  load average: 1,45, 1,36, 0,95
Tasks: 204 total,  2 running, 202 sleeping,  0 stopped,  0 zombie
%Cpu(s): 10,0 us, 21,9 sy,  0,0 ni, 67,2 id,  0,0 wa,  0,9 hi,  0,0 si,  0,0 st
MiB Mem : 3659,7 total, 1802,6 free, 1279,9 used,  830,4 buff/cache
MiB Swap: 4044,0 total, 4044,0 free,  0,0 used. 2379,8 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 44831 root       20   0   220948   1664   1664 R   98,7   0,0   3:03.45 yes
43962 aazhuko+  20   0 4130304 401560 126528 S    0,7  10,7   0:10.98 gnome-s+
   1 root       20   0   174488   17716  10700 S    0,0   0,5   0:01.20 systemd
   2 root       20   0         0         0         0 S    0,0   0,0   0:00.00 kthreadd
   3 root       0 -20         0         0         0 I    0,0   0,0   0:00.00 rcu_gp
   4 root       0 -20         0         0         0 I    0,0   0,0   0:00.00 rcu_par+
   5 root       0 -20         0         0         0 I    0,0   0,0   0:00.00 slub_fl+
   6 root       0 -20         0         0         0 I    0,0   0,0   0:00.00 netns
   8 root       0 -20         0         0         0 I    0,0   0,0   0:00.00 kworker+
  10 root       0 -20         0         0         0 I    0,0   0,0   0:00.00 mm_perc+
  11 root       20   0         0         0         0 I    0,0   0,0   0:00.29 kworker+
  12 root       20   0         0         0         0 I    0,0   0,0   0:00.00 rcu_tas+
  13 root       20   0         0         0         0 I    0,0   0,0   0:00.00 rcu_tas+
  14 root       20   0         0         0         0 I    0,0   0,0   0:00.00 rcu_tas+
  15 root       20   0         0         0         0 S    0,0   0,0   0:00.17 ksoftirq+
```

Рис. 4.5: Утилита top

5. Запускаю ещё три программы yes в фоновом режиме с подавлением потока вывода. Убиваю два процесса: для одного использую его PID, а для другого — его идентификатор конкретного задания. Пробую послать сигнал 1 (SIGHUP) процессу, запущенному с помощью nohup, и обычному процессу (рис. 4.6).

```

[root@aazhukoval ~]# yes > /dev/null &
[1] 44998
[root@aazhukoval ~]# yes > /dev/null &
[2] 44999
[root@aazhukoval ~]# yes > /dev/null &
[3] 45001
[root@aazhukoval ~]# kill -9 44999
[root@aazhukoval ~]# fg 3
yes > /dev/null
^C
[2] Убито yes > /dev/null
[root@aazhukoval ~]# kill -1 44998
[root@aazhukoval ~]# kill -1 45001
-bash: kill: (45001) - Нет такого процесса
[1]+ Обрыв терминальной линии yes > /dev/null

```

Рис. 4.6: Работа команд kill -9, kill -1

6. Запускаю ещё несколько программ yes в фоновом режиме с подавлением потока вывода. Завершаю их работу одновременно, используя команду killall (рис. 4.7).

```

[root@aazhukoval ~]# yes > /dev/null &
[1] 45002
[root@aazhukoval ~]# yes > /dev/null &
[2] 45004
[root@aazhukoval ~]# yes > /dev/null &
[3] 45005
[root@aazhukoval ~]# killall yes
[1] Завершено yes > /dev/null
[2]- Завершено yes > /dev/null
[3]+ Завершено yes > /dev/null

```

Рис. 4.7: Удаление программ одновременно

7. Запускаю программу yes в фоновом режиме с подавлением потока вывода. Используя утилиту nice, запускаю программу yes с теми же параметрами и с приоритетом, большим на 5. Сравниваю абсолютные и относительные приоритеты у этих двух процессов. Используя утилиту renice, изменяю приоритет у одного из потоков yes таким образом, чтобы у обоих потоков приоритеты были равны (рис. 4.8).

```

[root@aazhukoval ~]# yes > /dev/null &
[1] 45008
[root@aazhukoval ~]# nice -5 15 yes > /dev/null &
[2] 45009
[root@aazhukoval ~]# nice: «15»: Нет такого файла или каталога
^C
[2]+ Выход 127          nice -5 15 yes > /dev/null
[root@aazhukoval ~]# nice -n 15 yes > /dev/null &
[2] 45010
[root@aazhukoval ~]# ps -l | grep yes
0 R      0   45008   44949 98   80   0 - 55237 -      pts/0    00:01:22 yes
0 R      0   45010   44949 96   95   15 - 55237 -      pts/0    00:00:19 yes
[root@aazhukoval ~]# renice -n 15 45008
45008 (process ID) old priority 0, new priority 15
[root@aazhukoval ~]# ps -l | grep yes
0 R      0   45008   44949 98   95   15 - 55237 -      pts/0    00:02:21 yes
0 R      0   45010   44949 97   95   15 - 55237 -      pts/0    00:01:18 yes
[root@aazhukoval ~]#

```

Рис. 4.8: Установление приоритета



## 5 Контрольные вопросы

1. Команда `jobs` выводит список всех текущих заданий оболочки. Она показывает статус каждого задания (работает, остановлен, в фоновом режиме).
2. Чтобы остановить текущее задание оболочки и продолжить его выполнение в фоновом режиме, можно использовать комбинацию клавиш `Ctrl + Z` (в большинстве оболочек). После этого введите команду `bg` и нажмите `Enter`, чтобы перевести задание в фоновый режим.
3. Для отмены текущего задания оболочки используйте комбинацию клавиш `Ctrl + C`.
4. В этом случае нужно воспользоваться командой `kill`. Для этого необходимо знать PID (идентификатор процесса) отменяемого задания.
  - Чтобы найти PID, используйте команду `ps aux` (или `ps -ef`).
  - Затем выполните команду `kill -9`, чтобы немедленно остановить процесс.
  - Важно! Используйте `kill -9` только в крайнем случае, так как этот сигнал не дает процессу возможности завершиться корректно и может привести к потере данных.
5. Команда `ps tree` отображает дерево процессов, показывая связи между родительскими и дочерними процессами.
6. Команда `renice` используется для изменения приоритета процесса. Чтобы повысить приоритет процесса с PID 1234, выполните команду `renice -n -5 1234`

7. Проще всего остановить все процессы `dd`, используя команду `kill` с соответствующим шаблоном. Например: `kill dd`
8. Чтобы остановить команду с именем `mycommand`, используйте команду `kill`: `kill mycommand`
9. В `top` для убийства процесса используйте сигнал (`signal`). Чтобы убить процесс, введите номер PID процесса, который хотите убить, и нажмите `k`. Затем введите 9 (сигнал KILL) и нажмите `Enter`.
10. Запуск команды с высоким приоритетом может негативно повлиять на производительность других процессов.
  - Для планирования задач используйте `cron`. `Cron` позволяет выполнять команды в заданное время и с определенной периодичностью, что позволяет избежать перегрузки системы.
  - Использование команды `nice` позволит понизить приоритет запускаемого процесса. Например: `nice -n 10 mycommand`

Это уменьшит нагрузку на систему, освободив ресурсы для других процессов.

- Важно! Используйте высокий приоритет только в случае крайней необходимости.

## **6 Выводы**

Мы получили навыки управления процессами операционной системы.

## Список литературы

1. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ-Петербург, 2010.
2. Колисниченко Д. Н. Самоучитель системного администратора Linux. — СПб. : БХВПетербург, 2011. — (Системный администратор).
3. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
4. — (Классика Computer Science).
5. Neil N. J. Learning CentOS: A Beginners Guide to Learning Linux. — CreateSpace Independent Publishing Platform, 2016.
6. Unix и Linux: руководство системного администратора / Э. Немец, Г. Снайдер, Т. Хейн, Б. Уэйли, Д. Макни. — 5-е изд. — СПб. : ООО «Диалектика», 2020.