

# **Отчет о лабораторной работе**

**Лабораторная работа №6**

Казначеев Сергей Ильич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Контрольные вопросы</b>	<b>15</b>
<b>4</b>	<b>Выводы</b>	<b>17</b>

# Список иллюстраций

2.1	1	6
2.2	2	6
2.3	3	7
2.4	4	7
2.5	5	7
2.6	6	8
2.7	7	8
2.8	8	8
2.9	9	9
2.10	10	9
2.11	12	9
2.12	13	10
2.13	14	10
2.14	15	10
2.15	16	10
2.16	17	11
2.17	18	11
2.18	19	11
2.19	20	12
2.20	21	12
2.21	22	12
2.22	23	13
2.23	24	13
2.24	25	14

## **Список таблиц**

# **1 Цель работы**

Получить навыки управления процессами операционной системы.

## 2 Выполнение лабораторной работы

Для начала откроем терминал и перейдем в супер пользователя root после чего введем три команды 1. `sleep 3600 &` 2. `dd if=/dev/zero of=/dev/null &` 3. `sleep 7200` После остановим процесс `sleep 7200` (рис. 2.1).

```
[sikaznacheev@localhost ~]$ su -  
Пароль:  
[root@localhost ~]# sleep 3600 &  
[1] 4562  
[root@localhost ~]# dd if=/dev/zero of=/dev/null &  
[2] 4566  
[root@localhost ~]# sleep 7200  
^Z  
[3]+  Остановлен    sleep 7200
```

Рис. 2.1: 1

Далее проверяем наши задачи с помощью команды `jobs` и продолжим 3 задание в фоновом режиме с помощью команды `bg 3` (рис. 2.2).

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

Рис. 2.2: 2

После чего переводим 1 задание на передний план и отменим ее. Затем проверим статус заданий с помощью команды `jobs` после проверки сделаем тоже самое с 2 и 3 (рис. 2.3).

```

[root@localhost ~]# fg 1
sleep 3600
^C
[root@localhost ~]# jobs
[2]-  Запущен      dd if=/dev/zero of=/dev/null &
[3]+  Запущен      sleep 7200 &
[root@localhost ~]# fg 2
dd if=/dev/zero of=/dev/null
^C329850190+0 записей получено
329850189+0 записей отправлено
168883296768 байт (169 GB, 157 GiB) скопирован, 92,4393 s, 1,8 GB/s
[root@localhost ~]# fg 3
sleep 7200
^C
[root@localhost ~]#

```

Рис. 2.3: 3

Далее открываем второй терминал и под своей учетной записью вводим команду `dd if=/dev/zero of=/dev/null &` и введем `exit` чтобы закрыть второй терминал(рис. 2.4).

```

[sikaznacheev@localhost ~]$ dd if=/dev/zero of=/dev/null &
[1] 4649
[sikaznacheev@localhost ~]$ exit

```

Рис. 2.4: 4

Затем введем команду `top` чтобы проверить запущена задание `dd` или нет. После проверки выйдем используя `q`(рис. 2.5).

```

top - 12:53:37 up 28 min,  2 users,  load average: 2,16, 1,76, 1,07
Tasks: 304 total,  2 running, 302 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2,5 us,  3,8 sy,  0,0 ni, 86,8 id,  0,0 wa,  6,9 hi,  0,0 si,  0,0 st
MiB Mem : 3654,2 total,  696,4 free, 1984,3 used, 1286,8 buff/cache
MiB Swap: 4040,0 total, 4033,2 free,   6,8 used. 1669,9 avail Mem
PID to signal/kill [default pid = 4649]

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4649	sikazna+	20	0	220988	1792	1792	R	90,9	0,0	1:10.84	dd
2391	sikazna+	20	0	6467368	443080	146652	S	45,5	11,8	3:21.38	gnome-shell
88	root	39	19	0	0	0	S	4,5	0,0	0:00.61	khugepaged
4234	sikazna+	20	0	3030396	244148	159292	S	4,5	6,5	0:39.15	Isolated Web Co
1	root	20	0	174432	16788	10676	S	0,0	0,4	0:03.12	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.14	kthreadd

Рис. 2.5: 5

Теперь перейдем в супер пользователя и введем следующие три команды  
 1. `dd if=/dev/zero of=/dev/null &` 2. `dd if=/dev/zero of=/dev/null &` 3. `dd if=/dev/zero of=/dev/null &` (рис. 2.6).

```
[root@localhost ~]# su -
[root@localhost ~]# dd if=/dev/zero of=/dev/null &
[1] 4724
[root@localhost ~]# dd if=/dev/zero of=/dev/null &
[2] 4729
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# dd if=/dev/zero of=/dev/null &
[3] 4736
```

Рис. 2.6: 6

Затем введем команду `ps aux | grep dd` данная команда нам покажет строки в которых есть буква `dd` (рис. 2.7).

```
[root@localhost ~]# ps aux | grep dd
root      2  0.0  0.0   0   0 ?        S    12:25   0:00 [kthreadd]
root     1182  0.0  0.0 508432 3584 ?        Sl   12:25   0:00 /usr/sbin/VBoxService --pid
file /var/run/vboxadd-service.sh
sikazna+ 2561  0.0  0.8 881864 30328 ?        Ssl  12:39   0:00 /usr/libexec/evolution-addr
essbook-factory
sikazna+ 3071  0.0  0.0 232496 1540 ?        S    12:39   0:00 /usr/bin/VBoxClient --draga
nddrop
sikazna+ 3072  0.1  0.0 431300 3076 ?        Sl   12:39   0:01 /usr/bin/VBoxClient --draga
nddrop
sikazna+ 4227  0.0  1.2 269220 46984 ?        Sl   12:45   0:00 /usr/lib64/firefox/firefox
-contentproc -parentBuildID 20250725060637 -sandboxingKind 0 -prefsLen 34161 -prefMapSize 2496
91 -appDir /usr/lib64/firefox/browser {f6fae17a-3016-4822-88c7-fc1dde5b714} 3961 utility
root      4724 87.6  0.0 220988 1792 pts/0    R    12:54   0:28 dd if=/dev/zero of=/dev/nul
l
root      4729 87.5  0.0 220988 1792 pts/0    R    12:55   0:21 dd if=/dev/zero of=/dev/nul
l
root      4736 88.7  0.0 220988 1792 pts/0    R    12:55   0:12 dd if=/dev/zero of=/dev/nul
l
root      4739  0.0  0.0 221820 2560 pts/0    R+   12:55   0:00 grep --color=auto dd
```

Рис. 2.7: 7

После испльзуем PID ондого из процессов `dd` чтобы изменить приоритет (рис. 2.8).

```
[root@localhost ~]# renice -n 5 4724
4724 (process ID) old priority 0, new priority 5
```

Рис. 2.8: 8

Далее введем `ps fax | grep -B5 dd` и увидим иерархию отношений между процес-сами и оболочку из которой были запущены процессы `dd` и ее PID (рис. 2.9).



```

2516 ?      Ssl  0:00  \_ /usr/libexec/goa-identity-service
2525 ?      Ssl  0:00  \_ /usr/libexec/evolution-calendar-factory
2528 ?      Ssl  0:00  \_ /usr/libexec/gvfs-udisks2-volume-monitor
2546 ?      Ssl  0:00  \_ /usr/libexec/gvfs-mtp-volume-monitor
2560 ?      Ssl  0:00  \_ /usr/libexec/dconf-service
2561 ?      Ssl  0:00  \_ /usr/libexec/evolution-addressbook-factory

2941 ?      Sl   0:00  \_ /usr/libexec/ibus-x11 --kill-daemon
2948 ?      Ssl  0:00  \_ /usr/libexec/ibus-portal
3041 ?      Ssl  0:00  \_ /usr/libexec/xdg-desktop-portal-gtk
3063 ?      S    0:00  \_ /usr/bin/VBoxClient --seamless
3065 ?      Sl   0:00  | \_ /usr/bin/VBoxClient --seamless
3071 ?      S    0:00  \_ /usr/bin/VBoxClient --dragandddrop
3072 ?      Sl   0:01  | \_ /usr/bin/VBoxClient --dragandddrop

3907 pts/0  Ss    0:00  \_ bash
4507 pts/0  S      0:00      \_ su -
4518 pts/0  S      0:00          \_ -bash
4679 pts/0  S      0:00              \_ su -
4682 pts/0  S      0:00                  \_ -bash
4724 pts/0  RN    3:20                      \_ dd if=/dev/zero of=/dev/null
4729 pts/0  R      3:13                      \_ dd if=/dev/zero of=/dev/null
4736 pts/0  R      3:05                      \_ dd if=/dev/zero of=/dev/null
4780 pts/0  R+    0:00                      \_ ps fax
4781 pts/0  S+    0:00                      \_ grep --color=auto -B5 dd

```

Рис. 2.9: 9

После чего находим PID корневой оболочки из которой были запущены процессы dd и введем kill -9 4682 (рис. 2.10).

```

[root@localhost ~]# kill -9 4682
Убито

```

Рис. 2.10: 10

Далее запустим команду dd if=/dev/zero of=/dev/null трижды (рис. 2.11).

```

[root@localhost ~]# dd if=/dev/zero of=/dev/null &
[1] 4836
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# dd if=/dev/zero of=/dev/null &
[2] 4837
[root@localhost ~]# dd if=/dev/zero of=/dev/null &
[3] 4838

```

Рис. 2.11: 12

После чего увеличим приоритет одного из процессов на 5, а потом изменим на 15 и завершим все процессы (рис. 2.12).

```
[root@localhost ~]# renice -n -5 4836
4836 (process ID) old priority 0, new priority -5
[root@localhost ~]# renice -n -15 4836
4836 (process ID) old priority -5, new priority -15
```

Рис. 2.12: 13

После запуска программы `yes` в фоновом режиме с подавлением потока вывода, далее запустим программу `yes` на переднем плане с подавлением потока вывода. После чего приостановим процесс и заново запустим с теми же параметрами и завершим ее выполнение (рис. 2.13).

```
[sikaznacheev@localhost ~]$ su -
Пароль:
[root@localhost ~]# yes > /dev/null &
[1] 5056
[root@localhost ~]# yes > /dev/null
^Z
[2]+  Остановлен    yes > /dev/null
[root@localhost ~]# yes > /dev/null
^C
```

Рис. 2.13: 14

После чего проверим состояния заданий (рис. 2.14).

```
root@localhost ~]# jobs
1)  Запущен        yes > /dev/null &
2)-  Остановлен    yes > /dev/null
3)+  Остановлен    yes
```

Рис. 2.14: 15

Далее переводим процесс который выполняется в фоновом режиме на передний план и останавливаем его (рис. 2.15).

```
[root@localhost ~]# fg 1
yes > /dev/null
^Z
[1]+  Остановлен    yes > /dev/null
```

Рис. 2.15: 16

Далее переводим любой процесс с подавлением потока вывода в фоновом режиме, затем проверяем состояние заданий командой jobs и запускаем процесс в фоновом режиме таким образом чтобы он продолжил свою работу даже после отключения

```
[root@localhost ~]# bg 2
[2] yes > /dev/null &
[root@localhost ~]# jobs
[1]+  Остановлен      yes > /dev/null
[2]   Запущен         yes > /dev/null &
[3]-  Остановлен      yes
```

Рис. 2.16: 17

После закрываем окно и заново запускаем консоль(рис. 2.17).

```
[root@localhost ~]# nohup yes > /dev/null &
[4] 5125
[root@localhost ~]# nohup: ввод игнорируется и поток ошибок перенаправляется на стандартный вывод
[root@localhost ~]#
```

Рис. 2.17: 18

И проверяем процессы.(рис. 2.18).

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	MEM	TIME+	COMMAND
4724	root	25	5	220988	1792	1792	R	95,7	0,0	22:05.44	dd
4736	root	20	0	220988	1792	1792	R	95,0	0,0	22:12.15	dd
5125	root	20	0	220948	1792	1792	R	95,0	0,0	2:40.73	yes
4729	root	20	0	220988	1792	1792	R	94,7	0,0	22:30.11	dd
5065	root	20	0	220948	1792	1792	R	94,7	0,0	4:56.28	yes
2318	root	20	0	0	0	0	I	0,3	0,0	0:00.45	kworker/u33:0-events_unbound
3072	sikazna+	20	0	431390	3076	2816	S	0,3	0,1	0:01.41	VBoxClient
5064	root	20	0	0	0	0	I	0,3	0,0	0:00.85	kworker/1:2-events
1	root	20	0	174432	18708	10676	S	0,0	0,5	0:03.19	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.15	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.00	pool_workqueue_
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-sync_
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-slub_
7	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-netns
10	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker/u32:0-events_unbound
11	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-mm_pe
12	root	20	0	0	0	0	I	0,0	0,0	0:00.13	kworker/u32:1-netns
13	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_kthre
14	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_rude_
15	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_trace
16	root	20	0	0	0	0	S	0,0	0,0	0:00.14	ksoftirqd/0
17	root	20	0	0	0	0	I	0,0	0,0	0:31.14	rcu_preempt
18	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_exp_par_gp
19	root	20	0	0	0	0	S	0,0	0,0	0:00.95	rcu_exp_gp_kthr
20	root	rt	0	0	0	0	S	0,0	0,0	0:00.31	migration/0
21	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/0
22	root	20	0	0	0	0	I	0,0	0,0	0:00.43	kworker/0:1-mm_percpu_wq
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
26	root	rt	0	0	0	0	S	0,0	0,0	0:00.75	migration/1
27	root	20	0	0	0	0	I	0,0	0,0	0:00.10	ksoftirqd/1
29	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/1:0H-events_highpri
30	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/2
31	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/2
32	root	rt	0	0	0	0	S	0,0	0,0	0:00.72	migration/2
33	root	20	0	0	0	0	S	0,0	0,0	0:00.20	ksoftirqd/2
35	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/2:0H-events_highpri
36	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/3
37	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/3

Рис. 2.18: 19

Затем запустим три программы yes в фоновом режиме с подавлением потока вывода(рис. 2.19).

```
[root@localhost ~]# yes > /dev/null &
[5] 5164
[root@localhost ~]# yes > /dev/null &
[6] 5165
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# yes > /dev/null &
[7] 5202
```

Рис. 2.19: 20

После чего убиваем два процесса для одного используем PID а для другого его индификатор конкретного задания

```
[root@localhost ~]# fg 1
yes > /dev/null
^C
[root@localhost ~]# kill -9 5165
[root@localhost ~]# kill -9 5165
-bash: kill: (5165) - Нет такого процесса
[6] убито
yes > /dev/null
```

Рис. 2.20: 21

Пробуем послать сигнал 1 процессу запущенному с помощью nohup и обычному процессу

```
[root@localhost ~]# ps aux | grep yes
root      5065  85.5  0.0 220948 1792 pts/1    R   13:09  25:18 yes
root      5077   0.0  0.0 220948 1792 pts/1    T   13:10   0:00 yes
root      5125  96.5  0.0 220948 1792 pts/1    R   13:15  23:07 yes
root      5202  95.3  0.0 220948 1792 pts/1    R   13:22  16:25 yes
sikazna+  5437  93.8  0.0 220948 1792 pts/2    R   13:37   1:36 yes
sikazna+  5450  95.3  0.0 220948 1792 pts/2    R   13:38   1:26 yes
root      5490   0.0  0.0 221688 2432 pts/1    R+  13:39   0:00 grep --color=auto yes
[5] Обрыв терминальной линии
yes > /dev/null
[root@localhost ~]# kill -1 5437
[root@localhost ~]# kill -1 5450
[root@localhost ~]# ps aux | grep yes
root      5065  85.8  0.0 220948 1792 pts/1    R   13:09  26:26 yes
root      5077   0.0  0.0 220948 1792 pts/1    T   13:10   0:00 yes
root      5125  96.5  0.0 220948 1792 pts/1    R   13:15  24:15 yes
root      5202  95.4  0.0 220948 1792 pts/1    R   13:22  17:33 yes
root      5505   0.0  0.0 221688 2432 pts/1    S+  13:40   0:00 grep --color=auto yes
```

Рис. 2.21: 22

После чего запускаем еще несколько программ `yes` в фоновом режиме с подавлением потока вывода

```
[sikaznacheev@localhost ~]$ yes > /dev/null &
[3] 5522
[1] Обрыв терминальной линии          yes > /dev/null
[2] Обрыв терминальной линии          yes > /dev/null
[sikaznacheev@localhost ~]$ yes > /dev/null &
[4] 5536

[sikaznacheev@localhost ~]$

[sikaznacheev@localhost ~]$

[sikaznacheev@localhost ~]$
[sikaznacheev@localhost ~]$
[sikaznacheev@localhost ~]$ yes > /dev/null &
[5] 5557
```

Рис. 2.22: 23

И убиваем все процессы `yes` с помощью команды `killall yes`

```
[sikaznacheev@localhost ~]$ killall yes
yes(5065): Операция не позволена
yes(5077): Операция не позволена
yes(5125): Операция не позволена
yes(5202): Операция не позволена
[3] Завершено      yes > /dev/null
[4]- Завершено      yes > /dev/null
[5]+ Завершено      yes > /dev/null
```

Рис. 2.23: 24

Запустим программу `yes` в фоновом режиме с подавлением потока вывода. После чего используя утилиту `nice` запустим программу `yes` с теми же параметрами и с приоритетом большим на 5. И используя утилиту `renice` изменим приоритет у одного из потоков `yes` таким образом чтобы у обоих потоков приоритеты были равны

```

[sikaznacheev@localhost ~]$ yes > /dev/null &
[1] 5920
[sikaznacheev@localhost ~]$ nice -n 5 yes > /dev/null &
[2] 5929
[sikaznacheev@localhost ~]$ ps -l | grep yes
0 R 1000 5920 5807 93 80 0 - 55237 - pts/3 00:00:41 yes
0 R 1000 5929 5807 90 85 5 - 55237 - pts/3 00:00:15 yes
[sikaznacheev@localhost ~]$ renice -n 5 5920
5920 (process ID) old priority 0, new priority 5
[sikaznacheev@localhost ~]$ ps -l | grep yes
0 R 1000 5920 5807 94 85 5 - 55237 - pts/3 00:01:43 .3
0 R 1000 5929 5807 94 85 5 - 55237 - pts/3 00:01:16 yes
[sikaznacheev@localhost ~]$

```

Рис. 2.24: 25

### 3 Контрольные вопросы

1. Какая команда даёт обзор всех текущих заданий оболочки?

Ответ - команда `jobs` показывает список всех заданий запущенных из текущей оболочки

2. Как остановить текущее задание оболочки, чтобы продолжить его выполнение в фоновом режиме?

Ответ - сначала приостанавливаем задание нажав на комбинацию клавиш `Ctrl+Z` затем отправляем его фон `bg`

3. Какую комбинацию клавиш можно использовать для отмены текущего задания оболочки?

Ответ - комбинацию клавиш `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?

Ответ - используя команду `kill` указав `pid` процесса

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

Ответ - команда `ps tree`

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

Ответ - более высокий приоритет = меньшее значение nice sudo. Пример renice -n -5 -p 1234

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

Ответ - команда killall dd

8. Какая команда позволяет остановить команду с именем mycommand?

Ответ - команда pkill mycommand

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

Ответ - во время работы top нажать на клавишу k

10. Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов?

Ответ - запустить команду с пониженным приоритетом. Команда -nice -n 10



## **4 Выводы**

В результате выполнения лабораторной работы я получил навыки управления процессами операционной системы