

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

дисциплина: Основы администрирования операционных систем

Студент: Накова Амина Михайловна

Студ. билет № 1132232887

Группа: НПИбд-02-23

МОСКВА

2025 г.

Цель работы:

Целью данной работы является получение навыков управления процессами операционной системы.

Выполнение 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 (Рис. 1.1):

Рис. 1.1. Получение полномочий администратора, ввод трёх команд, остановка процесса, установка выполнения задания 3 в фоновом режиме, просмотр изменений в статусе заданий.

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 (Рис. 1.2):

```
[nakova@localhost ~]$ su nakova
Пароль:
[nakova@localhost ~]$ sleep 3600 &
[1] 37580
[nakova@localhost ~]$ dd if=/dev/zero of=/dev/null &
[2] 37592
[nakova@localhost ~]$ sleep 7200
```

```
^Z
[3]+  Остановлен    sleep 7200
[nakova@localhost ~]$ jobs
[1]  Запущен        sleep 3600 &
[2]-  Запущен        dd if=/dev/zero of=/dev/null &
[3]+  Остановлен    sleep 7200
[nakova@localhost ~]$ bg 3
[3]+ sleep 7200 &
[1]  Завершён        sleep 3600
```

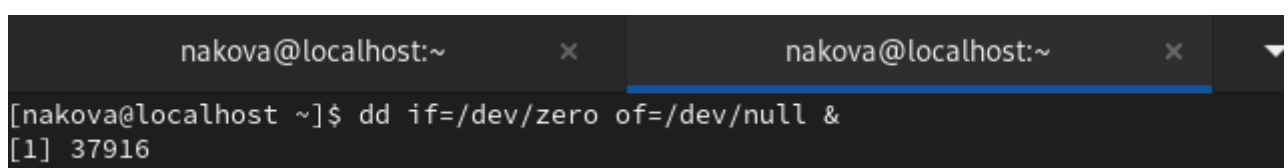
```
[nakova@localhost ~]$ ^C
[nakova@localhost ~]$ jobs
[2]-  Запущен        dd if=/dev/zero of=/dev/null &
[3]+  Запущен        sleep 7200 &
```

```
[nakova@localhost ~]$ fg 2
dd if=/dev/zero of=/dev/null
^C2654830860+0 записей получено
2654830860+0 записей отправлено
1359273400320 байт (1,4 TB, 1,2 TiB) скопирован, 3707,25 s, 367 MB/s

[nakova@localhost ~]$ jobs
[3]+  Запущен        sleep 7200 &
[nakova@localhost ~]$ fg 3
sleep 7200
^C
[nakova@localhost ~]$ jobs
[nakova@localhost ~]$
```

Рис. 1.2. Перемещение заданий на передний план и их последующая отмена.

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 (Рис. 1.3):



```
nakova@localhost:~ x nakova@localhost:~ x
[nakova@localhost ~]$ dd if=/dev/zero of=/dev/null &
[1] 37916
```

Рис. 1.3. Ввод команды и закрытие терминала.

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** (Рис. 1.4):

```
top - 22:01:22 up 20:07,  2 users,  load average: 0,64, 0,78, 0,78
Tasks: 217 total,   2 running, 215 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,4 id,  0,0 wa,  0,0 hi,  0,3 si,  0,0 st
MiB Mem :  3659,7 total,   264,1 free,  1707,8 used,  1980,9 buff/cache
MiB Swap:  4044,0 total,  4038,2 free,    5,8 used.  1951,9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
37916	nakova	20	0	220988	1792	1792	R	92,3	0,0	0:44.89	dd
1247	nakova	20	0	4124644	402520	136212	S	0,6	10,7	21:21.60	gnome-s+
1657	root	20	0	240744	9984	8704	S	0,3	0,3	0:37.33	sssd_kcm
37710	root	20	0	0	0	0	I	0,3	0,0	0:01.76	kworker+
37852	root	20	0	0	0	0	I	0,3	0,0	0:00.09	kworker+
37932	nakova	20	0	225884	4224	3456	R	0,3	0,1	0:00.15	top
1	root	20	0	174480	18432	10932	S	0,0	0,5	0:12.81	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.39	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp

```
Tasks: 217 total,   7 running, 210 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :  3659,7 total,   263,9 free,  1708,1 used,  1980,9 buff/cache
MiB Swap:  4044,0 total,  4038,2 free,    5,8 used.  1951,6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
37916	nakova	20	0	220988	1792	1792	R	91,7	0,0	1:25.85	dd
1247	nakova	20	0	4124644	402520	136212	S	0,3	10,7	21:21.99	gnome-s+
37852	root	20	0	0	0	0	I	0,3	0,0	0:00.11	kworker+
37937	nakova	20	0	225884	4352	3584	R	0,3	0,1	0:00.04	top
1	root	20	0	174480	18432	10932	S	0,0	0,5	0:12.81	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.39	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+

```
[nakova@localhost ~]$
```

Рис. 1.4. Убийство задания **dd** в **top**.

Управление процессами:

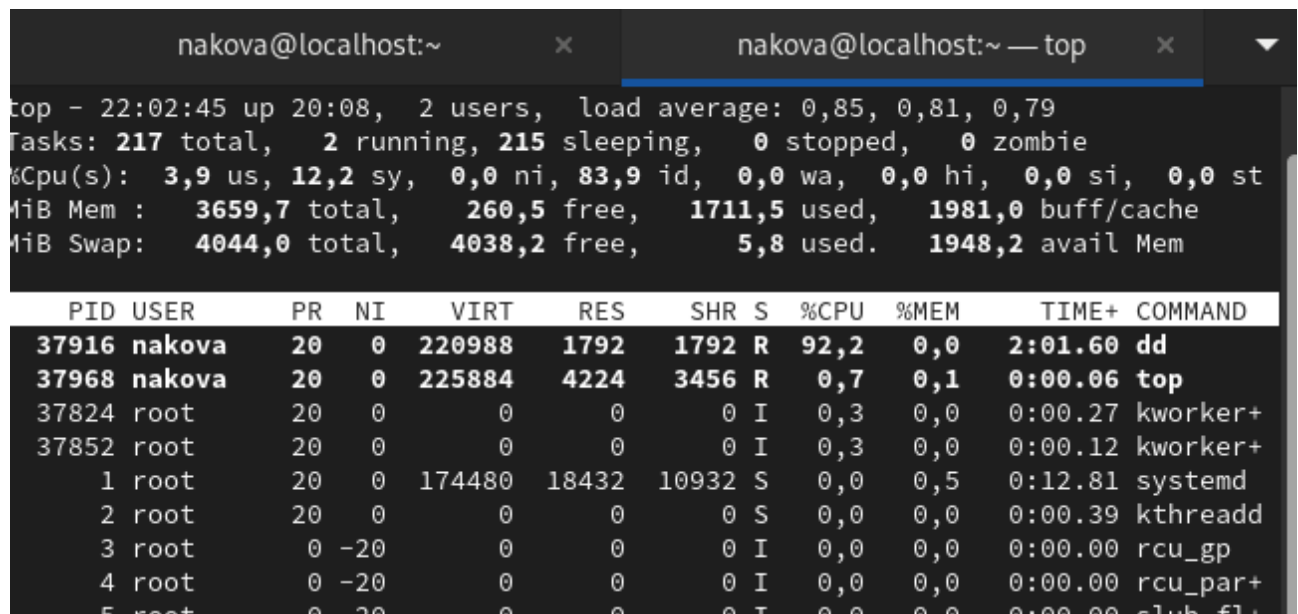
Получим полномочия администратора **su -** и введём следующие команды:

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

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

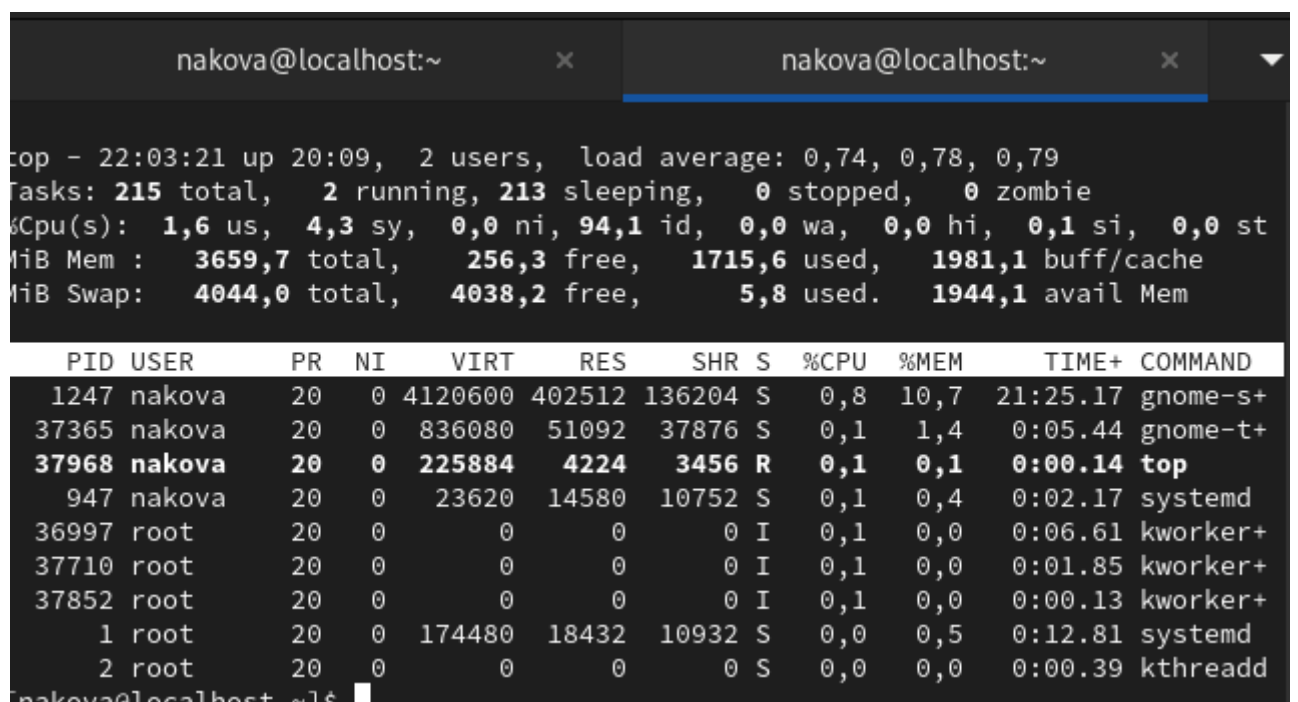
dd if=/dev/zero of=/dev/null &

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 2682**) (Рис. 2.1).



```
nakova@localhost:~ x nakova@localhost:~ — top x
top - 22:02:45 up 20:08, 2 users, load average: 0,85, 0,81, 0,79
Tasks: 217 total, 2 running, 215 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3,9 us, 12,2 sy, 0,0 ni, 83,9 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 3659,7 total, 260,5 free, 1711,5 used, 1981,0 buff/cache
MiB Swap: 4044,0 total, 4038,2 free, 5,8 used. 1948,2 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
37916	nakova	20	0	220988	1792	1792	R	92,2	0,0	2:01.60	dd
37968	nakova	20	0	225884	4224	3456	R	0,7	0,1	0:00.06	top
37824	root	20	0	0	0	0	I	0,3	0,0	0:00.27	kworker+
37852	root	20	0	0	0	0	I	0,3	0,0	0:00.12	kworker+
1	root	20	0	174480	18432	10932	S	0,0	0,5	0:12.81	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.39	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+



```
nakova@localhost:~ x nakova@localhost:~ x
top - 22:03:21 up 20:09, 2 users, load average: 0,74, 0,78, 0,79
Tasks: 215 total, 2 running, 213 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1,6 us, 4,3 sy, 0,0 ni, 94,1 id, 0,0 wa, 0,0 hi, 0,1 si, 0,0 st
MiB Mem : 3659,7 total, 256,3 free, 1715,6 used, 1981,1 buff/cache
MiB Swap: 4044,0 total, 4038,2 free, 5,8 used. 1944,1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1247	nakova	20	0	4120600	402512	136204	S	0,8	10,7	21:25.17	gnome-s+
37365	nakova	20	0	836080	51092	37876	S	0,1	1,4	0:05.44	gnome-t+
37968	nakova	20	0	225884	4224	3456	R	0,1	0,1	0:00.14	top
947	nakova	20	0	23620	14580	10752	S	0,1	0,4	0:02.17	systemd
36997	root	20	0	0	0	0	I	0,1	0,0	0:06.61	kworker+
37710	root	20	0	0	0	0	I	0,1	0,0	0:01.85	kworker+
37852	root	20	0	0	0	0	I	0,1	0,0	0:00.13	kworker+
1	root	20	0	174480	18432	10932	S	0,0	0,5	0:12.81	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.39	kthreadd

Рис. 2.1. Получение полномочий администратора, ввод команд. Просмотр всех строк, в которых есть dd. Изменение приоритета.

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 (Рис. 2.2).

```
[nakova@localhost ~]$ dd if=/dev/zero of=/dev/null &
[1] 37982
[nakova@localhost ~]$ dd if=/dev/zero of=/dev/null &
[2] 37988
[nakova@localhost ~]$ dd if=/dev/zero of=/dev/null &
[3] 37995
[nakova@localhost ~]$ ps aux | grep dd
root          2  0.0  0.0      0   0 ?        S    01:54   0:00 [kthreadd]
root         66  0.0  0.0      0   0 ?        I<   01:54   0:00 [ipv6_addrcon
f]
nakova       1667  0.0  0.8 880716 30356 ?        Ssl  01:54   0:00 /usr/libexec/
evolution-adddressbook-factory
nakova       2459  0.0  1.3 250144 48768 ?        Sl   01:56   0:00 /usr/lib64/fi
refox/firefox -contentproc -parentBuildID 20240418173117 -prefsLen 32439 -prefMa
pSize 236216 -appDir /usr/lib64/firefox/browser {86196bb7-e20e-4b6e-8cc4-dad6584
52e4c} 2115 rdd
nakova       37170 0.6  5.1 2840136 193276 ?        Sl   20:02   0:50 /usr/lib64/fi
refox/firefox -contentproc -childID 9 -isForBrowser -prefsLen 29370 -prefMapSize
236216 -jsInitLen 240916 -parentBuildID 20240418173117 -appDir /usr/lib64/firef
ox/browser {addd38931-f1a0-498f-9df5-c37a8cb285a6} 2115 tab
nakova       37982 72.6  0.0 220988 1792 pts/2    R    22:04   1:00 dd if=/dev/ze
ro of=/dev/null
nakova       37988 62.5  0.0 220988 1792 pts/2    R    22:04   0:38 dd if=/dev/ze
ro of=/dev/null
nakova       37995 57.6  0.0 220988 1792 pts/2    R    22:04   0:28 dd if=/dev/ze
ro of=/dev/null
nakova       38003 0.0  0.0 221820 2432 pts/2    S+   22:05   0:00 grep --color=
auto dd
```

Рис. 2.2. Просмотр иерархии отношений между 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) (Рис. 2.3).



Рис. 2.3. Заккрытие корневой оболочки.

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**

Самостоятельная работа (задание 2):

Получим полномочия администратора **su** – и запустим программу **yes** в фоновом режиме с подавлением потока вывода (**yes > /dev/null &**), 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 (**yes > /dev/null**). 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 (**fg 1, после чего Ctrl+c**). Переведём 3 процесс с подавлением потока вывода в фоновый режим (**bg 3**) и проверим состояния заданий, воспользовавшись командой **jobs**. Обратим внимание, что процесс стал выполняющимся (Running) в фоновом режиме. Запустим процесс в фоновом режиме таким образом, чтобы он продолжил свою работу даже после отключения от терминала (**nohup yes > /dev/null &**). Закроем окно и заново запустим консоль. Убедимся, что процесс продолжил свою работу (Рис. 4.1).



```
[2]+  Stopped                  yes > /dev/null
[3]-  Running                  yes > /dev/null &
```

Рис. 4.1. Получение полномочий администратора. Запуск программы **yes** в фоновом режиме с подавлением потока вывода. Запуск программы **yes** на переднем плане без подавления потока вывода. Перевод процесса на передний план и его остановка. Перевод процесса в фоновый режим. Проверка состояния заданий. Запуск процесса в фоновом режиме с условиями.

Сейчас получим информацию о запущенных в операционной системе процессах с помощью утилиты **top** (Рис. 4.2).

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_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-e+
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
13	root	20	0	0	0	0	S	0.0	0.0	0:00.07	ksoftirqd/0
14	root	20	0	0	0	0	I	0.0	0.0	0:00.12	rcu_preempt
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
20	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_frag_wq
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
25	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kcompactd0
26	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
27	root	39	19	0	0	0	S	0.0	0.0	0:00.12	khugepaged
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	cryptd
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
30	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
32	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
33	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
34	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
35	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	watchdogd
37	root	0	-20	0	0	0	I	0.0	0.0	0:00.03	kworker/0:1H-k+
38	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kswapd0

Рис. 4.2. Получение информации о запущенных в операционной системе процессах.

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 3098`), а для другого — его идентификатор конкретного задания (**fg 2** и **Ctrl+c**). Попробуем послать сигнал 1 (SIGHUP) процессу, запущенному с помощью `nohup` (`kill -1 3100`), и обычному процессу (`kill -1 2993`) (Рис. 4.3).

```
yes > /dev/null &  
yes > /dev/null &  
yes > /dev/null &  
  
kill -9 3098  
fg 2
```

Рис. 4.3. Запуск трёх программ `yes` в фоновом режиме с подавлением потока вывода, убийство двух процессов, попытка послать сигнал 1 (SIGHUP).

Запустим ещё несколько программ `yes` в фоновом режиме с подавлением потока вывода (`yes > /dev/null &`) и завершим их работу одновременно, используя команду `killall yes` (Рис. 4.4).

```
yes > /dev/null &  
yes > /dev/null &  
yes > /dev/null &  
  
killall yes
```

Рис. 4.4. Запуск программ `yes` в фоновом режиме с подавлением потока вывода и одновременное завершение их работы.

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 3109`) (Рис. 4.5).

```

0 R    0    3109    3071 92  95  15 - 55238 -    pts/0    00:01:37 yes
0 R    0    3113    3071  7  95  15 - 55238 -    pts/0    00:00:03 yes

```

Рис. 4.5. 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.

Ответы на контрольные вопросы:

1. Какая команда даёт обзор всех текущих заданий оболочки? **jobs.**

```

[2]+  Stopped                  yes > /dev/null
[3]-  Running                  yes > /dev/null &

```

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

```

[3]+ yes > /dev/null &

```

3. Какую комбинацию клавиш можно использовать для отмены текущего задания оболочки? **Ctrl+c.**

```

^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? **Внутри top использовать k, чтобы убить задание.**

```

top - 12:46:36 up 30 min, 1 user, load average: 0.95, 0.51, 0.22
Tasks: 198 total,  3 running, 195 sleeping,  0 stopped,  0 zombie
%Cpu(s): 48.8 us, 49.8 sy,  0.0 ni,  0.0 id,  0.0 wa,  1.3 hi,  0.0 si,  0.0 st
MiB Mem : 1967.1 total,  545.0 free,  821.4 used,  600.7 buff/cache
MiB Swap: 2096.0 total, 2096.0 free,  0.0 used.  975.3 avail Mem
PID to signal/kill [default pid = 2593] 2593

```

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

```
2531 pts/0    S      0:00      |  \_ su -
2537 pts/0    S+     0:00      |  \_ -bash
2598 pts/2    Ss     0:00      \_ bash
2643 pts/2    S      0:00      \_ su -
2654 pts/2    S      0:00      \_ -bash
2682 pts/2    RN     0:38      \_ dd if=/dev/zero of=/dev/null
2683 pts/2    R      0:41      \_ dd if=/dev/zero of=/dev/null
2684 pts/2    R      0:40      \_ dd if=/dev/zero of=/dev/null
2688 pts/2    R+     0:00      \_ ps fax
2689 pts/2    S+     0:00      \_ grep --color=auto -B5 dd
```

6. Какая команда позволит изменить приоритет процесса с идентификатором 1234 на более высокий? **renice -n приоритет_процесса <PID>**.

```
3109 (process ID) old priority 0, new priority 15
```

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

```
# killall dd
#
```

8. Какая команда позволяет остановить команду с именем mycommand? Сначала узнаем PID процесса **mycommand -ps aux | grep mycommand** далее команда **kill -9 <PID>**.

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

```
top - 12:46:36 up 30 min, 1 user, load average: 0.95, 0.51, 0.22
Tasks: 198 total, 3 running, 195 sleeping, 0 stopped, 0 zombie
%Cpu(s): 48.8 us, 49.8 sy, 0.0 ni, 0.0 id, 0.0 wa, 1.3 hi, 0.0 si, 0.0 st
MiB Mem : 1967.1 total, 545.0 free, 821.4 used, 600.7 buff/cache
MiB Swap: 2096.0 total, 2096.0 free, 0.0 used. 975.3 avail Mem
PID to signal/kill [default pid = 2593] 2593
```

10. Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов? **Запустить команду в фоновом режиме.**

Вывод:

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