

# **Лабараторная работа №6**

**Отчет**

Славинский Владислав Вадимович

# 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>Выводы</b>	<b>21</b>
<b>4</b>	<b>Ответы на контрольные вопросы</b>	<b>22</b>

# Список иллюстраций

2.1	Переход в режим суперпользователя . . . . .	6
2.2	Ввод команд . . . . .	7
2.3	Остановка процесса . . . . .	7
2.4	Комада jobs . . . . .	8
2.5	Выполнение задания 3 в фоновом режиме . . . . .	8
2.6	Отмена задания 1 . . . . .	9
2.7	Отмена заданий 2 и 3 . . . . .	9
2.8	Ввод команды во втором терминале . . . . .	10
2.9	Запуск top . . . . .	10
2.10	Завершение задания dd . . . . .	11
2.11	Ввод команды . . . . .	11
2.12	Строки, в которых есть буквы dd . . . . .	12
2.13	Смена приоритета . . . . .	12
2.14	Параметр -B5 . . . . .	13
2.15	Удаление корневой оболочки . . . . .	13
2.16	Запуск команды в фоновом значении . . . . .	14
2.17	Изменение приоритета . . . . .	14
2.18	Изменение приоритета . . . . .	15
2.19	Завершение процессов dd . . . . .	15
2.20	yes в фоновом режиме с подавлением потока вывода . . . . .	15
2.21	yes на переднем плане с подавлением потока вывода . . . . .	16
2.22	yes на переднем плане без подавления потока вывода . . . . .	16
2.23	Состояние заданий . . . . .	16
2.24	Перевод процесса на передний план и его остановка . . . . .	17
2.25	Перевод процесса в фоновый режим . . . . .	17
2.26	Проверка состояния заданий . . . . .	17
2.27	Запуск процесса, чтобы продолжал работу после закрытия терминала . . . . .	17
2.28	Информация о запущенных в операционной системе процессах . . . . .	18
2.29	Запуск программ yes в фоновом режиме с подавлением потока вывода . . . . .	18
2.30	Уничтожение процессов . . . . .	18
2.31	Послание сигнала 1 процессам . . . . .	19
2.32	Запуск программ yes в фоновом режиме с подавлением потока вывода . . . . .	19
2.33	Завершение их работы с помощью killall . . . . .	19
2.34	Запуск двух программ yes, но у одной программы приоритет больше на 5 . . . . .	20
2.35	Установка равных приоритетов . . . . .	20

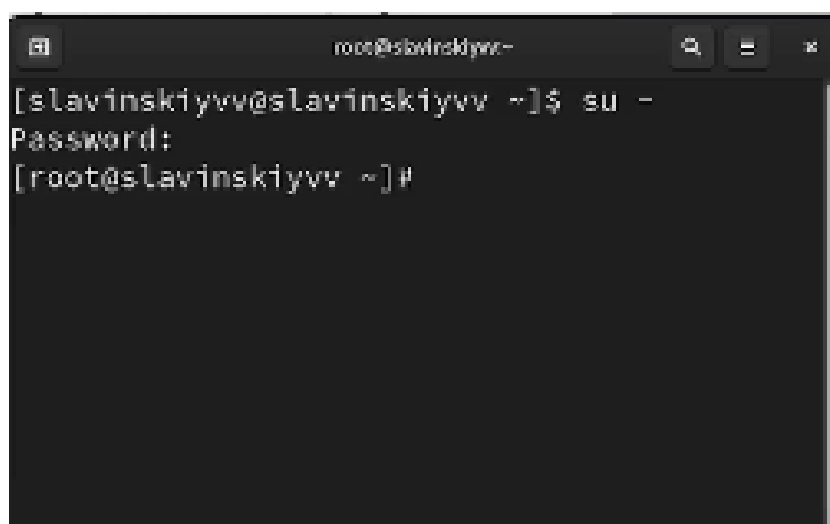
## **Список таблиц**

# **1 Цель работы**

Получить навыки управления процессами операционной системы.

## 2 Выполнение лабораторной работы

В консоли перейдем в режим работы суперпользователя, используя команду `su -`. (рис. 2.1)

A screenshot of a terminal window with a dark background. The window title is 'root@slavinskiyv~'. The prompt is '[slavinskiyv@slavinskiyv ~]\$'. The user has entered the command 'su -'. The prompt changes to 'Password:'. The user has entered a password (not visible). The prompt changes to '[root@slavinskiyv ~]#', indicating successful transition to root user.

```
root@slavinskiyv~  
[slavinskiyv@slavinskiyv ~]$ su -  
Password:  
[root@slavinskiyv ~]#
```

Рис. 2.1: Переход в режим суперпользователя

Введем следующие команды: `sleep 3600 &`, `dd if=/dev/zero of=/dev/null &` и `sleep 7200`(рис. 2.2)

```
[slavinskiyv@slavinskiyv ~]$ su -
Password:
[root@slavinskiyv ~]# sleep 3600 &
[1] 4130
[root@slavinskiyv ~]# dd if=/dev/zero of=/dev/null &
[2] 4137
[root@slavinskiyv ~]# sleep 7200
```

Рис. 2.2: Ввод команд

Поскольку мы запустили последнюю команду без & у нас есть 2 часа, прежде чем мы снова получите контроль над оболочкой. Введем ctrl+z, чтобы остановить процесс. (рис. 2.3)

```
[2] 4137
[root@slavinskiyv ~]# sleep 7200
^Z
[3]+  stopped                  sleep 7200
[root@slavinskiyv ~]#
```

Рис. 2.3: Остановка процесса

Введем команду jobs. Мы видим три процесса, которые мы запустили. Два первых процесса имеют статус running, а последний имеет статус stopped. (рис. 2.4)

```
[3]+  Stopped                  sleep 7200
[root@slavinskiyv ~]# jobs
[1]-  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Stopped                  sleep 7200
[root@slavinskiyv ~]#
```

Рис. 2.4: Комада jobs

Для продолжения выполнения задания 3 в фоновом режиме введем bg 3 и посмотрим статус через jobs. Видим, что состояние изменилось на running. (рис. 2.5)

```
[root@slavinskiyv ~]# jobs
[1]-  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev,
[3]+  Stopped                  sleep 7200
[root@slavinskiyv ~]# bg 3
[3]+  sleep 7200 &
[root@slavinskiyv ~]# jobs
[1]-  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev,
[3]+  Running                  sleep 7200 &
[root@slavinskiyv ~]#
```

Рис. 2.5: Выполнение задания 3 в фоновом режиме

Для перемещения задания 1 на передний план введем fg 1. После чего отменим задание через ctrl+c и посмотрим статус (рис. 2.6)



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

Рис. 2.6: Отмена задания 1

Проделаем то же самое с заданиями 2 и 3. (рис. 2.7)

```
[3]+  Running                  sleep 7200 &
[root@slavinskiyv ~]# fg 2
dd if=/dev/zero of=/dev/null
^C390802180+0 records in
390802179+0 records out
200890715648 bytes (200 GB, 186 GiB) copied, 127.685 s
[root@slavinskiyv ~]# fg 3
sleep 7200
^C
[root@slavinskiyv ~]#
```

Рис. 2.7: Отмена заданий 2 и 3

Откроем второй терминал и под учётной записью своего пользователя введем в нём: `dd if=/dev/zero of=/dev/null &` и выйдем из него.(рис. 2.8)

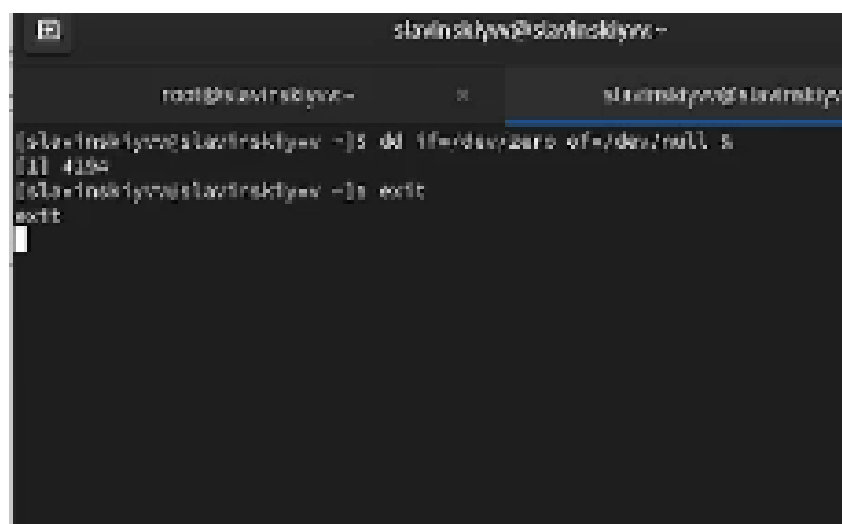


Рис. 2.8: Ввод команды во втором терминале

На другом терминале под учётной записью своего пользователя запустим `top`. Видим, что задание `dd` все еще запущено. Выйдем из `top` через `q`. (рис. 2.9)

PID	USER	PR	NI	VSZ	RES	SHR S	MEM	TIME	COMMAND
4194	slavinskijev	20	0	120988	1664	1664 R	38.7	0.0	0:11.51 dd
2277	slavinskijev	20	0	5502768	552168	177732 S	4.7	7.0	3:25.93 gnome-shell
2874	root	20	0	0	0	0 I	1.3	0.0	0:01.04 kworker/0:2-10+
3065	slavinskijev	20	0	763440	54112	16016 S	1.3	0.7	0:01.07 gnome-terminal
4238	slavinskijev	20	0	125088	4224	3224 R	0.3	0.1	0:00.01 top
1	root	20	0	172816	18160	10780 S	0.0	0.3	0:01.35 systemd
3	root	20	0	0	0	0 S	0.0	0.0	0:00.02 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 T	0.0	0.0	0:00.00 kworker/0:0-cm
5	root	0	-20	0	0	0 T	0.0	0.0	0:00.00 kworker/0:0-cm
6	root	0	-20	0	0	0 T	0.0	0.0	0:00.00 kworker/0:0-lm
7	root	0	-20	0	0	0 T	0.0	0.0	0:00.00 kworker/0:0-rcu
8	root	0	-20	0	0	0 T	0.0	0.0	0:00.00 kworker/0:0-mm
10	root	0	-20	0	0	0 T	0.0	0.0	0:00.00 kworker/0:0-cm
11	root	0	-20	0	0	0 T	0.0	0.0	0:00.00 kworker/0:0-mm
12	root	20	0	0	0	0 T	0.0	0.0	0:00.00 kworker/u32:1-
13	root	20	0	0	0	0 T	0.0	0.0	0:00.00 rcu_tasks_kthre
14	root	20	0	0	0	0 T	0.0	0.0	0:00.00 rcu_tasks_rude
15	root	20	0	0	0	0 T	0.0	0.0	0:00.00 rcu_tasks_trace
16	root	20	0	0	0	0 S	0.0	0.0	0:00.00 ksoftirqd/0
17	root	20	0	0	0	0 T	0.0	0.0	0:00.14 rcu_preempt
18	root	20	0	0	0	0 S	0.0	0.0	0:00.00 rcu_exp_gov_g
19	root	20	0	0	0	0 S	0.0	0.0	0:00.00 rcu_exp_gov_kt
20	root	0	-20	0	0	0 S	0.0	0.0	0:00.00 migration/0
21	root	0	-20	0	0	0 S	0.0	0.0	0:00.00 idle_inject/0

Рис. 2.9: Запуск `top`

Вновь запустим `top` и с помощью `k` убьем задание `dd`. Потом выйдем из `top` с помощью `q`. (рис. 2.10)

Mem Mem : 7881.9 total, 4889.8 free, 2294.3 used, 2299.2 buff/cache		Mem Swap: 2482.9 total, 2192.4 free, 0.5 used, 2489.4 avail Mem									
PID	USER	PR	NI	VIRT	RES	SHR	S	WCHAN	TIME	COMMAND	
1	root	20	0	171200	10392	10750	S	0.0	0.0	0:01.35	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	htlreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	postl.workqueue+
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/B-hcu+
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/B-hcu+
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/B-hcu+
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/B-hcu+
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/B-hcu+
10	root	20	0	0	0	0	T	0.0	0.0	0:00.00	kworker/u3000+
11	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/B-hcu+
12	root	20	0	0	0	0	I	0.0	0.0	0:00.01	kworker/u3000+
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_klbt+
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_klbt+
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_klbt+
16	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kacfti rep/0
17	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_gp_hlpt
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_exp_gp_hl+
19	root	20	0	0	0	0	S	0.0	0.0	0:00.01	rcu_exp_gp_hl+
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.01	migrator/0
21	root	-rt	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
22	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	-rt	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	0:00.00	migrator/1
27	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kacfti rep/1

lshw -n 100

Рис. 2.10: Завершение задания dd

Перейдем в режим суперпользователя и введем следующую команду 3 раза dd if=/dev/zero of=/dev/null &. (рис. 2.11)

```
sleep 7200
^C
[root@slavinskiyvv ~]# jobs
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[1] 4276
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[2] 4277
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[3] 4278
[root@slavinskiyvv ~]#
```

Рис. 2.11: Ввод команды

Введем ps aux | grep dd. Это показывает все строки, в которых есть буквы dd. Запущенные процессы dd идут последними. (рис. 2.12)

```

[2] 4277
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[3] 4278
[root@slavinskiyvv ~]# ps aux | grep dd
root      2   0.0  0.0      0   0 ?        S    19:34   0:00 [kthreadd]
root     1131   0.0  0.0 508476 3584 ?        Sl   19:34   0:00 /usr/sbin/VBoxSe
ce --pidfile /var/run/vboxadd-service.sh
slavins+ 2442   0.0  0.3 881216 29956 ?        Ssl  19:46   0:00 /usr/libexec/evon
tion-addressbook-factory
root     4276 98.0  0.0 220988 1664 pts/0    R    20:21   0:12 dd if=/dev/zero o
/dev/null
root     4277 94.2  0.0 220988 1664 pts/0    R    20:21   0:11 dd if=/dev/zero o
/dev/null
root     4278 99.0  0.0 220988 1664 pts/0    R    20:21   0:09 dd if=/dev/zero o
/dev/null
root     4282   0.0  0.0 221796 2304 pts/0    S+   20:21   0:00 grep --color=aut
d
[root@slavinskiyvv ~]#

```

Рис. 2.12: Строки, в которых есть буквы dd

Используем PID одного из процессов dd, например 4276, чтобы изменить приоритет. Используем `renice -n 5` . (рис. 2.13)

```

y
root     4276 98.0  0.0 220988 1664 pts/0    R    20:21   0:12 dd if=
root     4277 94.2  0.0 220988 1664 pts/0    R    20:21   0:11 dd if=
root     4278 99.0  0.0 220988 1664 pts/0    R    20:21   0:09 dd if=
root     4282   0.0  0.0 221796 2304 pts/0    S+   20:21   0:00 grep --
[root@slavinskiyvv ~]# renice -n 5 4276
4276 (process ID) old priority 0, new priority 5
[root@slavinskiyvv ~]#

```

Рис. 2.13: Смена приоритета

Введем `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 увидим оболочку, из которой были запущены все процессы dd, и её PID.(рис. 2.14)

```

4276 (process 10) old priority 0, new priority 5
[root@slavinskiyvv ~]# ps fax | grep -B5 dd
  PID TTY          STAT TIME COMMAND
    2 ?           S      0:00 [kthreadd]

--
 852 ?           Sns     0:00 /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf
tfiler=/lib/alsa/init/00main rdaemon
 865 ?           S        0:00 /usr/sbin/chronyd -F 2
 903 ?           Ssl     0:00 /usr/sbin/ModemManager
 917 ?           Ssl     0:00 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
1129 ?           Sl      0:00 /usr/bin/VBoxDRMClient
1131 ?           Sl      0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh

--
2393 ?           Ssl     0:00 \_ /usr/libexec/goa-identity-service
2404 ?           Ssl     0:00 \_ /usr/libexec/gvfs-udisks2-volume-monitor
2413 ?           Ssl     0:00 \_ /usr/libexec/gvfs-mtp-volume-monitor
2425 ?           Ssl     0:00 \_ /usr/libexec/gvfs-gphoto2-volume-monitor
2432 ?           Ssl     0:00 \_ /usr/libexec/dconf-service
2442 ?           Ssl     0:00 \_ /usr/libexec/evolution-addressbook-factory

--
2975 ?           Ssl     0:00 \_ /usr/libexec/gvfsd-metadata
3365 ?           Ssl     0:02 \_ /usr/libexec/gnome-terminal-server
3435 pts/0        Ss      0:00 | \_ bash
4083 pts/0        S        0:00 | | \_ su -
4094 pts/0        S        0:00 | | \_ -bash
4276 pts/0        RN      2:04 | | \_ dd if=/dev/zero of=/dev/null
4277 pts/0        R        2:04 | | \_ dd if=/dev/zero of=/dev/null
4278 pts/0        R        2:02 | | \_ dd if=/dev/zero of=/dev/null
4337 pts/0        R+      0:00 | | \_ ps fax
4338 pts/0        S+      0:00 | | \_ grep --color=auto -B5 dd
[root@slavinskiyvv ~]#

```

Рис. 2.14: Параметр -B5

Найдем PID корневой оболочки (у нас значение 4094), из которой были запущены процессы dd, и введем kill -9 (рис. 2.15)

```

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

--
 852 ?           Sns     0:00 /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.co
tfiler=/lib/alsa/init/00main rdaemon
 865 ?           S        0:00 /usr/sbin/chronyd -F 2
 903 ?           Ssl     0:00 /usr/sbin/ModemManager
 917 ?           Ssl     0:00 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
1129 ?           Sl      0:00 /usr/bin/VBoxDRMClient
1131 ?           Sl      0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh

--
2393 ?           Ssl     0:00 \_ /usr/libexec/goa-identity-service
2404 ?           Ssl     0:00 \_ /usr/libexec/gvfs-udisks2-volume-monitor
2413 ?           Ssl     0:00 \_ /usr/libexec/gvfs-mtp-volume-monitor
2425 ?           Ssl     0:00 \_ /usr/libexec/gvfs-gphoto2-volume-monitor
2432 ?           Ssl     0:00 \_ /usr/libexec/dconf-service
2442 ?           Ssl     0:00 \_ /usr/libexec/evolution-addressbook-factory

--
2975 ?           Ssl     0:00 \_ /usr/libexec/gvfsd-metadata
3365 ?           Ssl     0:03 \_ /usr/libexec/gnome-terminal-server
3435 pts/0        Ss      0:00 | \_ bash
4083 pts/0        S        0:00 | | \_ su -
4094 pts/0        S        0:00 | | \_ -bash
4277 pts/0        R        3:35 | | \_ dd if=/dev/zero of=/dev/null
4278 pts/0        R        3:34 | | \_ dd if=/dev/zero of=/dev/null
4362 pts/0        R+      0:00 | | \_ ps fax
4363 pts/0        S+      0:00 | | \_ grep --color=auto -B5 dd
[1] Killed dd if=/dev/zero of=/dev/null
[root@slavinskiyvv ~]# kill -9 4094
Killed
[slavinskiyvv@slavinskiyvv ~]$

```

Рис. 2.15: Удаление корневой оболочки

Введем три раза команду dd if=/dev/zero of=/dev/null. Нам нужно запустить команду как фоновое значение, поэтому в конце добавляем &.(рис. 2.16)

```
[slavinskiyvv@slavinskiyvv ~]$ su -
Password:
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[1] 5885
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[2] 5886
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[3] 5887
[root@slavinskiyvv ~]#
```

Рис. 2.16: Запуск команды в фоновом значении

Увеличим приоритет одной из этих команд, используя значение приоритета -5. Введем команду `renice -n -5 5005`(рис. 2.17)

```
[slavinskiyvv@slavinskiyvv ~]$ su -
Password:
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[1] 5885
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[2] 5886
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[3] 5887
[root@slavinskiyvv ~]# renice -n -5 5005
5005 (process ID) old priority 0, new priority -5
[root@slavinskiyvv ~]#
```

Рис. 2.17: Изменение приоритета

Изменим приоритет того же процесса ещё раз, но применим на этот раз значение -15. Разница в том, что при приоритете -15 процесс получает гораздо больше процессорного времени, чем при -5. Чем меньше значение, тем выше приоритет.(рис. 2.18)

```
[2] 5006
[root@slavinskiyvv ~]# dd if=/dev/zero of=/dev/null &
[3] 5007
[root@slavinskiyvv ~]# renice -n -5 5005
5005 (process ID) old priority 0, new priority -5
[root@slavinskiyvv ~]# renice -n -15 5005
5005 (process ID) old priority -5, new priority -15
[root@slavinskiyvv ~]#
```

Рис. 2.18: Изменение приоритета

Завершим все процессы dd, которые мы запустили.(рис. 2.19)

```
5005 (process ID) old priority -5, new priority -15
[root@slavinskiyvv ~]# killall dd
bash: killall dd: command not found...
[root@slavinskiyvv ~]# killall dd
[1] Terminated dd if=/dev/zero of=/dev/null
[2]- Terminated dd if=/dev/zero of=/dev/null
[3]+ Terminated dd if=/dev/zero of=/dev/null
[root@slavinskiyvv ~]#
```

Рис. 2.19: Завершение процессов dd

Запустим программу yes в фоновом режиме с подавлением потока вывода с помощью команды `yes > /dev/null &`.(рис. 2.20)

```
root 5047 0.0 0.0 221796 2304 pts/1 5-
[root@slavinskiyvv ~]# yes > /dev/null &
[1] 5067
[root@slavinskiyvv ~]#
```

Рис. 2.20: yes в фоновом режиме с подавлением потока вывода

Запустим программу yes на переднем плане с подавлением потока вывода с помощью команды `yes > /dev/null`. Затем приостановим программу через `ctrl+z`. Потом заново запустим и завершим процесс через `ctrl+c`(рис. 2.21)





Переведем процесс, который у нас выполняется в фоновом режиме, на передний план, затем остановим его.(рис. 2.24)

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

Рис. 2.24: Перевод процесса на передний план и его остановка

Переведем второй процесс с подавлением потока вывода в фоновый режим.(рис. 2.25)

```
[1]+  Stopped                  yes > /dev/null
[root@slavinskiyvv ~]# bg 2
[2] yes > /dev/null &
[root@slavinskiyvv ~]#
```

Рис. 2.25: Перевод процесса в фоновый режим

Проверим состояние заданий, воспользовавшись командой jobs. И видим, что второй процесс стал выполняться(рис. 2.26)

```
[root@slavinskiyvv ~]# jobs
[1]+  Stopped                  yes > /dev/null
[2]   Running                  yes > /dev/null &
[3]-  Stopped                  yes
[root@slavinskiyvv ~]#
```

Рис. 2.26: Проверка состояния заданий

Запустим процесс в фоновом режиме таким образом, чтобы он продолжил свою работу даже после отключения от терминала.(рис. 2.27)

```
[root@slavinskiyvv ~]# nohup yes > /dev/null &
[4] 7389
[root@slavinskiyvv ~]# nohup: ignoring input and redirecting stderr to stdout
```

Рис. 2.27: Запуск процесса, чтобы продолжал работу после закрытия терминала

Закроем и откроем заново консоль. Убедимся, что процесс продолжил свою работу. Получим информацию с помощью утилиты top.(рис. 2.28)

free Mem: 10012 total, 4980 free, 2200 used, 10012 avail Mem MFD Swap: 3192.0 total, 3192.0 free, 0.0 used, 5414.1 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
7309	root	20	0	220948	1664	1664	R	90.3	0.0	1:04.96 yes
7065	root	20	0	220948	1664	1664	R	90.0	0.0	6:51.45 yes
2177	slavins+	20	0	5695028	555248	100232	S	1.0	7.1	12:08.75 gnome-shell
3739	slavins+	20	0	2762324	169308	111980	S	0.3	2.2	0:03.85 Privileged Cont
1	root	20	0	173996	16540	10760	S	0.0	0.2	0:01.49 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.02 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
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/0:0H-events_highpri
10	root	20	0	0	0	0	I	0.0	0.0	0:00.00 kworker/u20:0-events_unbound
11	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/R-mm_pg
12	root	20	0	0	0	0	I	0.0	0.0	0:00.02 kworker/u20:1-netns
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_tasks_kthre

Рис. 2.28: Информация о запущенных в операционной системе процессах

Запустим ещё три программы `yes` в фоновом режиме с подавлением потока вывода..(рис. 2.29)

```

[slavinskiyvv@slavinskiyvv ~]$ yes > /dev/null &
[1] 7354
[slavinskiyvv@slavinskiyvv ~]$ yes > /dev/null &
[2] 7350
[slavinskiyvv@slavinskiyvv ~]$ yes > /dev/null &
[3] 7364
[slavinskiyvv@slavinskiyvv ~]$

```

Рис. 2.29: Запуск программ `yes` в фоновом режиме с подавлением потока вывода

Убьем два процесса: для одного используйте его PID, а для другого — его идентификатор конкретного задания.(рис. 2.30)

```

[slavinskiyvv@slavinskiyvv ~]$ kill 7354
[1] Terminated yes > /dev/null
[slavinskiyvv@slavinskiyvv ~]$ kill 2
bash: kill: (2) - Operation not permitted
[slavinskiyvv@slavinskiyvv ~]$ kill -9 3071
bash: kill: (3071) - No such process
[slavinskiyvv@slavinskiyvv ~]$ jobs
[2]- Running yes > /dev/null &
[3]+ Running yes > /dev/null &
[slavinskiyvv@slavinskiyvv ~]$ fg 2
yes > /dev/null
^C
[slavinskiyvv@slavinskiyvv ~]$ jobs
[3]+ Running yes > /dev/null &
[slavinskiyvv@slavinskiyvv ~]$

```

Рис. 2.30: Уничтожение процессов

Попробуем послать сигнал 1 (SIGHUP) процессу, запущенному с помощью `nohup`, и обычному процессу.(рис. 2.31)

```

yes > /dev/null
^C
[slavinskiyvv@slavinskiyvv ~]$ jobs
[3]+  Running                  yes > /dev/null &
[slavinskiyvv@slavinskiyvv ~]$ ps aux | grep yes
root      7305 49.5  0.0 229948 1664 pts/1    R    00:05 14:20 yes
root      7309 98.2  0.0 229948 1664 pts/1    R    00:25  8:34 yes
slavins+  7304 97.8  0.0 229948 1664 pts/1    R    00:28  5:30 yes
slavins+  7432  0.0  0.0 221664 2176 pts/1    S+   00:34  0:00 grep --color=auto yes
[slavinskiyvv@slavinskiyvv ~]$ kill -L 7309
bash: kill: (7309): - Operation not permitted
[slavinskiyvv@slavinskiyvv ~]$ sudo kill -L 7309
[sudo] password for slavinskiyvv:
[slavinskiyvv@slavinskiyvv ~]$ kill -L 7304
[3]+  Hangup                  yes > /dev/null
[slavinskiyvv@slavinskiyvv ~]$

```

Рис. 2.31: Послание сигнала 1 процессам

Запустим ещё несколько программ yes в фоновом режиме с подавлением потока вывода.(рис. 2.32)

```

[slavinskiyvv@slavinskiyvv ~]$ kill -L 7304
[3]+  Hangup                  yes > /dev/null
[slavinskiyvv@slavinskiyvv ~]$ yes > /dev/null &
[1] 7541
[slavinskiyvv@slavinskiyvv ~]$ yes > /dev/null &
[2] 7546
[slavinskiyvv@slavinskiyvv ~]$ yes > /dev/null &
[3] 7551
[slavinskiyvv@slavinskiyvv ~]$

```

Рис. 2.32: Запуск программ yes в фоновом режиме с подавлением потока вывода

Завершим их работу одновременно, используя команду killall.(рис. 2.33)

```

[3] 7551
[slavinskiyvv@slavinskiyvv ~]$ killall yes
yes(7005): Operation not permitted
yes(7309): Operation not permitted
[1]-  Terminated              yes > /dev/null
[2]-  Terminated              yes > /dev/null
[3]+  Terminated              yes > /dev/null
[slavinskiyvv@slavinskiyvv ~]$

```

Рис. 2.33: Завершение их работы с помощью killall

Запустим программу yes в фоновом режиме с подавлением потока вывода. Используя утилиту nice, запустим программу yes с теми же параметрами и с приоритетом, большим на 5. Видим, что приоритеты у них разные.(рис. 2.34)

```
Y
y
^C
[slavinskiyvv@slavinskiyvv ~]$ ps -l | grep yes
0 R 1000 7506 4521 98 80 0 - 55237 - pts/1 00:02:54 yes
0 R 1000 7519 4521 96 85 5 - 55237 - pts/1 00:02:10 yes
[slavinskiyvv@slavinskiyvv ~]$
```

Рис. 2.34: Запуск двух программ yes, но у одной программы приоритет больше на 5

Используя утилиту renice, изменим приоритет у одного из потоков yes таким образом, чтобы у обоих потоков приоритеты были равны. Меняем приоритет у 7506 на 5 и получается, что приоритеты теперь равны.(рис. 2.35)

```
0 R 1000 7519 4521 98 85 5 - 55237 - pts/1 00:02:13 yes
[slavinskiyvv@slavinskiyvv ~]$ renice -n 5 7506
7506 (process ID): old priority 8, new priority 5
[slavinskiyvv@slavinskiyvv ~]$ ps -l | grep yes
0 R 1000 7506 4521 98 85 5 - 55237 - pts/1 00:04:25 yes
0 R 1000 7519 4521 97 85 5 - 55237 - pts/1 00:03:43 yes
[slavinskiyvv@slavinskiyvv ~]$
```

Рис. 2.35: Установка равных приоритетов

## **3 Выводы**

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

## 4 Ответы на контрольные вопросы

1. jobs
2. ctrl+z, bg
3. ctrl+c
4. kill
5. ps fax
6. renice -n -5 1234
7. killall -9 dd
8. killall mycommand
9. В интерфейсе top, чтобы убить процесс нужно нажать клавишу k.
10. nice -n mycommand