

# Лабораторная работа №6

Управление процессами

Перфилов Александр Константинович | группа НПИбд 03-24

# 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>Выполнение заданий для самостоятельной работы</b>	<b>12</b>
4.1	Самостоятельная работа (задание 1) . . . . .	12
4.2	Самостоятельная работа (задание 2) . . . . .	13
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>17</b>
<b>6</b>	<b>Выводы</b>	<b>19</b>

# Список иллюстраций

3.1	Получение полномочий администратора, ввод трёх команд, остановка процесса, установка выполнения задания 3 в фоновом режиме, просмотр изменений в статусе заданий . . . . .	8
3.2	Перемещение заданий на передний план и их последующая отмена. . . . .	8
3.3	Ввод команды и закрытие терминала. . . . .	9
3.4	Убийство задания dd в top. . . . .	9
3.5	Получение полномочий администратора, ввод команд. Просмотр всех строк, в которых есть dd. Изменение приоритета. . . . .	10
3.6	Просмотр иерархии отношений между процессами. . . . .	11
3.7	Закрытие корневой оболочки. . . . .	11
4.1	Получение полномочий администратора, запуск команды трижды как фоновое задание. . . . .	12
4.2	Увеличение приоритета первой команды. . . . .	12
4.3	Увеличение приоритета первой команды. . . . .	12
4.4	Завершение всех процессов. . . . .	13
4.5	Получение полномочий администратора. Запуск программы уes в фоновом режиме с подавлением потока вывода. Запуск программы уes 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. . . . .	14
4.6	Получение информации о запущенных в операционной системе процессах. . . . .	15
4.7	Запуск трёх программ уes в фоновом режиме с подавлением потока вывода, убийство двух процессов, попытка послать сигнал 1 (SIGHUP). . . . .	15
4.8	Запуск программ уes в фоновом режиме с подавлением потока вывода и одновременное завершение их работы. . . . .	16
4.9	Запуск программы уes в фоновом режиме с подавлением потока вывода. Запуск программы уes с теми же параметрами и с приоритетом, большим на 5. Сравнение абсолютных и относительных приоритетов, изменение приоритета. . . . .	16

## **Список таблиц**

# **1 Цель работы**

Целью данной работы является получение навыков управления процессами операционной системы.

## 2 Задание

1. Продемонстрируйте навыки управления заданиями операционной системы
2. Продемонстрируйте навыки управления процессами операционной системы
3. Выполните задания для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 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

```

akpperfilov@akpperfilov:~$ su -
Пароль:
su: Сбой при проверке подлинности
akpperfilov@akpperfilov:~$ su -
Пароль:
Последний вход в систему: Пт окт  3 23:02:33 MSK 2025 на pts/0
Последняя неудачная попытка входа в систему: Вт окт  7 21:33:46 MSK 2025 на pts/0
Со времени последнего входа была 1 неудачная попытка.
root@akpperfilov:~# sleep 3600 &
[1] 4477
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &
[2] 4496
root@akpperfilov:~# sleep 7200
^Z
[3]+  Остановлен      sleep 7200
root@akpperfilov:~# jobs
[1]  Запущен          sleep 3600 &
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Остановлен      sleep 7200
root@akpperfilov:~# bg 3
[3]+ sleep 7200 &
root@akpperfilov:~# jobs
[1]  Запущен          sleep 3600 &
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Запущен          sleep 7200 &

```

Рисунок 3.1: 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

```

root@akpperfilov:~# fg 1
sleep 3600
^C
root@akpperfilov:~# jobs
[2]-  Запущен          dd if=/dev/zero of=/dev/null &
[3]+  Запущен          sleep 7200 &
root@akpperfilov:~# fg 2
dd if=/dev/zero of=/dev/null
^C412276466+0 records in
412276466+0 records out
211085550592 bytes (211 GB, 197 01B) copied, 132,653 s, 1,6 GB/s
root@akpperfilov:~# fg 3
sleep 7200
^C
root@akpperfilov:~# jobs
root@akpperfilov:~#

```

Рисунок 3.2: 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



```
akpperfilov@akpperfilov:~$ dd if=/dev/zero of=/dev/null &
[1] 4633
akpperfilov@akpperfilov:~$ exit
```

Рисунок 3.3: 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`

```
top - 21:40:11 up 12 min, 4 users, load average: 1,31, 1,03, 0,58
Tasks: 284 total, 2 running, 282 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6,6 us, 11,3 sy, 0,0 ni, 81,7 id, 0,0 wa, 0,3 hi, 0,0 si, 0,0 st
MiB Mem : 7678,5 total, 4455,5 free, 2106,6 used, 1431,6 buff/cache
MiB Swap: 3312,0 total, 3312,0 free, 0,0 used, 5571,9 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 4633 akpperf+  20   0 227308 2164 2164 R 99,0   0,0   2:54.25 dd
 4215 akpperf+  20   0 3964508 361728 98236 S  7,3   4,6   0:09.44 ptxis
 2359 akpperf+  20   0 5505248 397936 132596 S  4,7   5,1   1:24.39 gnome-shell
   67 root        20   0      0      0      0 I   0,3   0,0   0:01.02 kworker/u26:2-even+
  720 root        20   0      0      0      0 I   0,3   0,0   0:00.62 kworker/u29:3-even+
 4716 root        20   0 231604 5500 3324 R  0,3   0,1   0:00.03 top
    1 root        20   0  49192  41012 10140 S   0,0   0,5   0:01.64 systemd
    2 root        20   0      0      0      0 S   0,0   0,0   0:00.01 kthreadd
    3 root        20   0      0      0      0 S   0,0   0,0   0:00.00 pool_workqueue_rel+
    4 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kworker/R-rcu_gp
    5 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kworker/R-sync_wq
    6 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kworker/R-slub_flu+
    7 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kworker/R-netns
   10 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-event+
   11 root        20   0      0      0      0 I   0,0   0,0   0:00.00 kworker/u24:0-even+
   12 root        20   0      0      0      0 I   0,0   0,0   0:00.01 kworker/u24:1-ipv6+
   13 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kworker/R-mm_percp+
   14 root        20   0      0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_kthread
   15 root        20   0      0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_rude_kth+
   16 root        20   0      0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_trace_kt+
   17 root        20   0      0      0      0 S   0,0   0,0   0:00.00 ksoftirqd/0
   18 root        20   0      0      0      0 I   0,0   0,0   0:00.28 rcu_preeempt
   19 root        20   0      0      0      0 S   0,0   0,0   0:00.00 rcu_exp_par_gp_kth+
   20 root        20   0      0      0      0 S   0,0   0,0   0:00.01 rcu_exp_gp_kthread+
   21 root        rt    0      0      0      0 S   0,0   0,0   0:00.00 migration/0
   22 root       -51   0      0      0      0 S   0,0   0,0   0:00.00 idle_inject/0
   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   0:00.36 migration/1
```

Рисунок 3.4: Убийство задания `dd` в `top`.

## 3.2 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`)

```

root@akpperfilov:~# ps aux | grep dd
root      2  0.0  0.0      0   0 ?        S   21:27   0:00 [kthreadd]
root     12  0.0  0.0      0   0 ?        I   21:27   0:00 [kworker/u24:1-tpv6_add
conf]
root     114  0.0  0.0      0   0 ?        I<  21:27   0:00 [kworker/R-ipv6_addrcnf
]
akpperf+ 2771  0.0  0.3 1037064 26248 ?        Ssl  21:28   0:00 /usr/libexec/evolution-a
ddressbook-factory
akpperf+ 3556  0.0  0.4 260964 34792 ?        Sl   21:28   0:00 /usr/lib64/firefox/firef
ox -contentproc -parentBuildID 20250918202509 -prefsHandle 0:35214 -prefMapHandle 1:271119
-sandboxReporter 2 -chrootClient 3 -ipcHandle 4 -initialChannelId {f60f3ef7-e618-481b-aa2b-
2cf88708562c} -parentPid 3437 -appDir /usr/lib64/firefox/browser 3 rdd
akpperf+ 3697  4.3  3.0 2985700 236156 ?        Sl   21:28   0:33 /usr/lib64/firefox/firef
ox -contentproc -isForBrowser -prefsHandle 0:40970 -prefMapHandle 1:271119 -jsInitHandle 2:
242716 -parentBuildID 20250918202509 -sandboxReporter 3 -chrootClient 4 -ipcHandle 5 -initi
alChannelId {8318f7a5-7dd0-4bfe-8e1b-1e1962fc2447} -parentPid 3437 -greomni /usr/lib64/fire
fox/omni.ja -appomni /usr/lib64/firefox/browser/omni.ja -appDir /usr/lib64/firefox/browser
8 tab
akpperf+ 4093  0.0  0.1 609620 9440 ?        Sl   21:29   0:00 /usr/libexec/gvfsd-wsdd
--spanner :1.17 /org/gtk/gvfs/exec_spaw/4
akpperf+ 4098  0.0  0.3 259076 28944 ?        S   21:29   0:00 /usr/bin/python3 /usr/bi
n/wsdd --no-host --discovery --listen /run/user/1000/gvfsd/wsdd
root     4966 98.7  0.0 227308 2196 pts/0    R   21:41   0:12 dd if=/dev/zero of=/dev/
null
root     4970 98.7  0.0 227308 2088 pts/0    R   21:41   0:11 dd if=/dev/zero of=/dev/
null
root     4972 98.2  0.0 227308 2008 pts/0    R   21:41   0:10 dd if=/dev/zero of=/dev/
null
root     4986  0.0  0.0 227712 2376 pts/0    S+  21:41   0:00 grep --color=auto dd
root@akpperfilov:~# renice -n 5 4966
4966 (process ID) old priority 0, new priority 5
root@akpperfilov:~#

```

Рисунок 3.5: 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

```

root@akpperfilov:~# ps fax | grep -B5 dd
  PID TTY          STAT TIME   COMMAND
    2 ?        S      0:00 [kthreadd]
--
 108 ?        I<      0:00 \_ [kworker/R-acpi_thermal_pm]
 109 ?        I<      0:00 \_ [kworker/R-kmpath_rdad]
 110 ?        I<      0:00 \_ [kworker/R-kalud]
 111 ?        I<      0:00 \_ [kworker/R-mld]
 113 ?        I<      0:00 \_ [kworker/4:1H-xfs-log/dm-0]
 114 ?        I<      0:00 \_ [kworker/R-ipv6_addrconf]
--
 2365 ?        Ssl     0:00 \_ /usr/libexec/gvfsd
 4004 ?        Sl      0:00 | \_ /usr/libexec/gvfsd-trash --spawnner :1.17 /org/gtk/gvfs/e
xec_spaw/0
 4072 ?        Sl      0:00 | \_ /usr/libexec/gvfsd-recent --spawnner :1.17 /org/gtk/gvfs/
exec_spaw/1
 4073 ?        Sl      0:00 | \_ /usr/libexec/gvfsd-network --spawnner :1.17 /org/gtk/gvfs
/exec_spaw/2
 4086 ?        Sl      0:00 | \_ /usr/libexec/gvfsd-dnssd --spawnner :1.17 /org/gtk/gvfs/e
xec_spaw/3
 4093 ?        Sl      0:00 | \_ /usr/libexec/gvfsd-wsdd --spawnner :1.17 /org/gtk/gvfs/ex
ec_spaw/4
--
 2648 ?        Ssl     0:00 \_ /usr/bin/gjs -m /usr/share/gnome-shell/org.gnome.ScreenSaver
 2661 ?        Ssl     0:00 \_ /usr/libexec/ibus-portal
 2736 ?        Ssl     0:00 \_ /usr/libexec/evolution-calendar-factory
 2743 ?        Ssl     0:00 \_ /usr/libexec/goa-identity-service
 2753 ?        Ssl     0:00 \_ /usr/libexec/gvfs-mtp-volume-monitor
 2771 ?        Ssl     0:00 \_ /usr/libexec/evolution-addressbook-factory
--
 3002 ?        Ssl     0:00 \_ /usr/libexec/xdg-desktop-portal-gnome
 3016 ?        Ssl     0:00 \_ /usr/libexec/xdg-desktop-portal-gtk
 3927 ?        Ssl     0:00 \_ /usr/libexec/gvfsd-metadata

```

Рисунок 3.6: 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)

```

root@akpperfilov:~# kill -9 4419

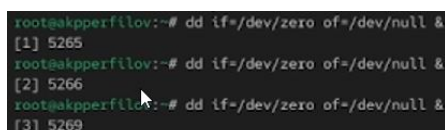
```

Рисунок 3.7: Закрытие корневой оболочки.

## 4 Выполнение заданий для самостоятельной работы

### 4.1 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`



```
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &
[1] 5265
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &
[2] 5266
root@akpperfilov:~# dd if=/dev/zero of=/dev/null &
[3] 5269
```

Рисунок 4.1: Получение полномочий администратора, запуск команды трижды как фоновое задание.



```
root@akpperfilov:~# renice -n -5 5266
5266 (process ID) old priority 0, new priority -5
```

Рисунок 4.2: Увеличение приоритета первой команды.



```
root@akpperfilov:~# renice -n -15 5266
5266 (process ID) old priority -5, new priority -15
```

Рисунок 4.3: Увеличение приоритета первой команды.

```

root@akpperfilov:~# fg 1
dd if=/dev/zero of=/dev/null
^C296294751+0 records in
296294750+0 records out
151702912000 bytes (152 GB, 141 GiB) copied, 103,217 s, 1,5 GB/s

root@akpperfilov:~# fg 2
dd if=/dev/zero of=/dev/null
^C300621645+0 records in
300621645+0 records out
153918282240 bytes (154 GB, 143 GiB) copied, 107,674 s, 1,4 GB/s

root@akpperfilov:~# fg 3^C
root@akpperfilov:~# fg 3
dd if=/dev/zero of=/dev/null
^C316103869+0 records in
316103868+0 records out
161845180416 bytes (162 GB, 151 GiB) copied, 115,315 s, 1,4 GB/s

root@akpperfilov:~# jobs
root@akpperfilov:~#

```

Рисунок 4.4: Завершение всех процессов.

## 4.2 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 переведем процесс, который у нас выполняется в фоновом режиме, на передний план, затем остановим его (`fg 1`, после чего `Ctrl+c`). Переведем 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

```
root@akpperfilov:~# yes > /dev/null &
[1] 5293
root@akpperfilov:~# yes > /dev/null
^Z
[2]+  Остановлен   yes > /dev/null
root@akpperfilov:~# yes > /dev/null
^C
root@akpperfilov:~# jobs
[1]-  Запущен      yes > /dev/null &
[2]+  Остановлен   yes > /dev/null
root@akpperfilov:~# yes > /dev/null
^C
root@akpperfilov:~# jobs
[1]-  Запущен      yes > /dev/null &
[2]+  Остановлен   yes > /dev/null
root@akpperfilov:~# fg 1
yes > /dev/null
^Z
[1]+  Остановлен   yes > /dev/null
root@akpperfilov:~# bg 2
[2]- yes > /dev/null &
root@akpperfilov:~# jobs
[1]+  Остановлен   yes > /dev/null
[2]- Запущен      yes > /dev/null &
root@akpperfilov:~# nohup yes > /dev/null &
[3] 5360
nohup: ввод игнорируется и поток ошибок перенаправляется на стандартный вывод
root@akpperfilov:~#
```

Рисунок 4.5: 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

```
top - 22:01:54 up 34 min, 5 users, load average: 4,32, 4,24, 3,39
Tasks: 287 total, 6 running, 281 sleeping, 0 stopped, 0 zombie
%Cpu(s): 29,9 us, 61,2 sy, 6,0 ni, 1,5 id, 0,0 wa, 1,5 hi, 0,0 si, 0,0 st
MiB Mem : 7678,5 total, 4385,9 free, 2174,5 used, 1433,7 buff/cache
MiB Swap: 3312,0 total, 3312,0 free, 0,0 used, 5503,9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4970	root	20	0	227308	2088	2088	R	100,0	0,0	20:29.25	dd
4972	root	20	0	227308	2008	2008	R	100,0	0,0	20:28.34	dd
4966	root	25	5	227308	2196	2196	R	90,9	0,0	20:24.90	dd
5295	root	20	0	227280	2128	2128	R	90,9	0,0	1:44.24	yes
5360	root	20	0	227280	2044	2044	R	90,9	0,0	0:23.43	yes
1	root	20	0	49192	41012	10140	S	0,0	0,5	0:01.69	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.00	pool_workqueue_relea+
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-rcu_gp
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-sync_wq
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-slab_flush+
7	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-netns
10	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H-events_+
11	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker/u24:0-events+
12	root	20	0	0	0	0	I	0,0	0,0	0:00.01	kworker/u24:1-ipv6_a+
13	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-mm_percpu_+
14	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_kthread
15	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_rude_kthre+
16	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_trace_kthr+
17	root	20	0	0	0	0	S	0,0	0,0	0:00.00	ksoftirqd/0
18	root	20	0	0	0	0	I	0,0	0,0	0:00.83	rcu_preempt
19	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_exp_par_gp_kthre+
20	root	20	0	0	0	0	S	0,0	0,0	0:00.01	rcu_exp_gp_kthread_w+
21	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
22	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/0
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

Рисунок 4.6: 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`), а для другого — его идентификатор конкретного задания (`fg 2` и `Ctrl+c`). Попробуем послать сигнал 1 (`SIGHUP`) процессу, запущенному с помощью `nohup` (`kill -1`), и обычному процессу (`kill -1`)

```
akpperfilov@akpperfilov:~$ yes > /dev/null &
[1] 5612
akpperfilov@akpperfilov:~$ yes > /dev/null &
[2] 5618
akpperfilov@akpperfilov:~$ yes > /dev/null &
[3] 5624
akpperfilov@akpperfilov:~$ fg 1
yes > /dev/null
^C
akpperfilov@akpperfilov:~$ kill -9 5618
[2]- Убито yes > /dev/null
akpperfilov@akpperfilov:~$ jobs
[3]+ Запущен yes > /dev/null &
akpperfilov@akpperfilov:~$
```

Рисунок 4.7: 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`

```
akpperfilov@akpperfilov:~$ yes > /dev/null &
[1] 5776
akpperfilov@akpperfilov:~$ yes > /dev/null &
[2] 5778
akpperfilov@akpperfilov:~$ yes > /dev/null &
[3] 5780
akpperfilov@akpperfilov:~$ killall yes
yes(5295): Операция не позволена
yes(5360): Операция не позволена
[1] Завершено yes > /dev/null
[2]- Завершено yes > /dev/null
[3]- Завершено yes > /dev/null
akpperfilov@akpperfilov:~$
```

Рисунок 4.8: 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`) (рис.

```
akpperfilov@akpperfilov:~$ yes > /dev/null &
[1] 5792
akpperfilov@akpperfilov:~$ nice -n 5 yes > /dev/null &
[2] 5796
akpperfilov@akpperfilov:~$ ps -l | grep yes
0 R 1000 5792 4298 98 80 0 - 56820 - pts/0 00:01:22 yes
0 R 1000 5796 4298 91 85 5 - 56820 - pts/0 00:00:25 yes
akpperfilov@akpperfilov:~$ renice -n 5 4298
4298 (process ID) old priority 0, new priority 5
akpperfilov@akpperfilov:~$ ps -l | grep yes
0 R 1000 5792 4298 98 80 0 - 56820 - pts/0 00:02:13 yes
0 R 1000 5796 4298 91 85 5 - 56820 - pts/0 00:01:13 yes
akpperfilov@akpperfilov:~$ renice -n 5 5792
5792 (process ID) old priority 0, new priority 5
akpperfilov@akpperfilov:~$
```

Рисунок 4.9: 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.



## 5 Ответы на контрольные вопросы

1. Какая команда даёт обзор всех текущих заданий оболочки?

`jobs.`

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

`bg номер_задания`

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

`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?

Внутри `top` использовать `k`, чтобы убить задание.

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

`ps fax.`

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

`renice -n приоритет_процесса .`

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

`killall dd.`

8. Какая команда позволяет остановить команду с именем `mysommand`?

Сначала узнаем PID процесса `mysommand` -`ps aux | grep mysommand` далее команда `kill -9 .`

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

`k.`

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

Запустить команду в фоновом режиме.

## **6 Выводы**

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