# Управление процессами

## Лабораторная работа №6

Казначеев С.И.

7 октября 2025

Российский университет дружбы народов, Москва, Россия

# Информация

# Докладчик

:::::::::::::: {.columns align=center} ::: {.column width="70%"}

- Казначеев Сергей Ильич
- Студент
- Российский университет дружбы народов
- [1132240693@pfur.ru] ::: ::: {.column width="30%"}

# Цель работы

Получить навыки управления процессами операционной системы.

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

```
[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
```

# Перенос заданий

После чего переводим 1 задание на передний план и отменим ее.Затем проверим статус заданий с помощью команды jobs после проверки проделаем тоже самое с 2 и 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
```

# Выход из top

Затем введем команду top чтобы проверить запущена задание dd или нет. После проверки выйдем используя q

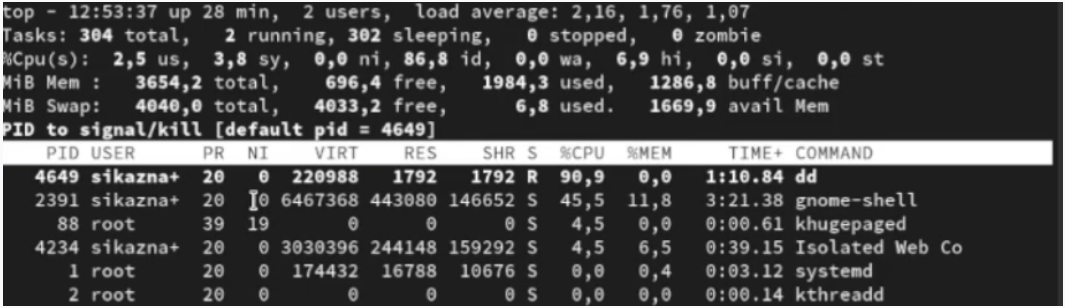



Рис. 5: 5

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 &

```
[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
```

# Просмотр строк dd

Затем введем комаду ps aux | grep dd данная команда нам покажет строки в которых есть буква dd

```
[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
ndrop
sikazna+    3072  0.1  0.0 431300  3076 ?        Sl   12:39   0:01 /usr/bin/VBoxClient --draga
ndrop
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
```

# Смена приритетов

После испльзуем PID ондого из процессов dd чтобы изменить приоритет

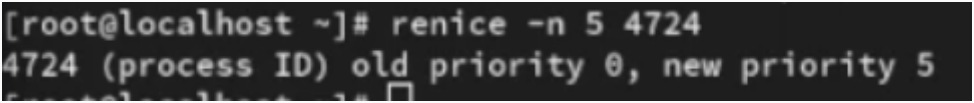
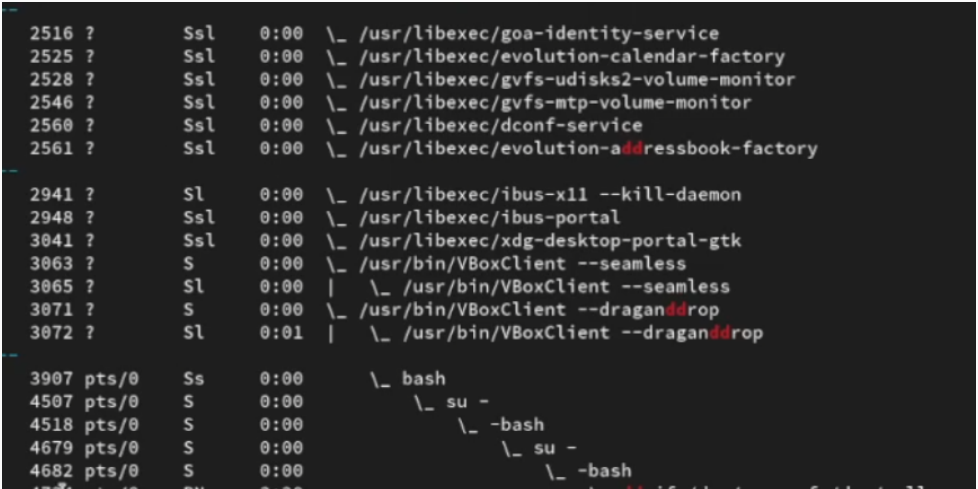



**Рис. 8:** 8

# Просмотр иерархии

Далее введем ps fax | grep -B5 dd и увидим иерархию отношений между процессами и оболочку из которой были запущены процессы dd и ее PID



# Убийство процесса

После чего находим PID корневой оболчки из которой были запущены процессы dd и введем kill -9 4682


**Рис. 10:** 10

Далее запустим команду dd if=/dev/zero of=/dev/null трижды

```
[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. 11:** 12

После чего увеличим приоритет одного из процессов на 5, а потом изменим на 15 и завершим все процессы

```
[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. 12: 13

После запусти программу yes в фоновом режиме с подавлением потока вывода,далее запустим программу yes на переднем плане с подавлением потока вывода.После чего приостоновим процесс и заново запустим с теми же параметрами и завершим ее выполение

```
[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. 13: 14

После чего проверим состояния заданий

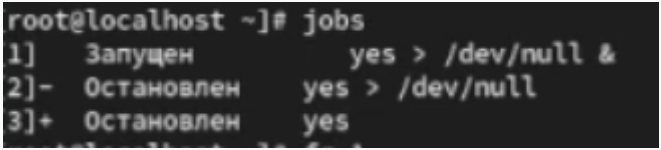



**Рис. 14:** 15

Далее переводим процесс который выполняется в фоновом режиме на передний план и останавливем его

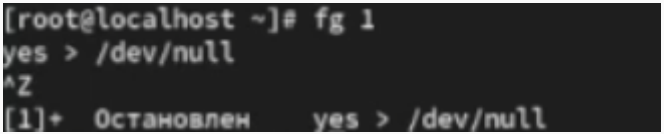



**Рис. 15:** 16

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

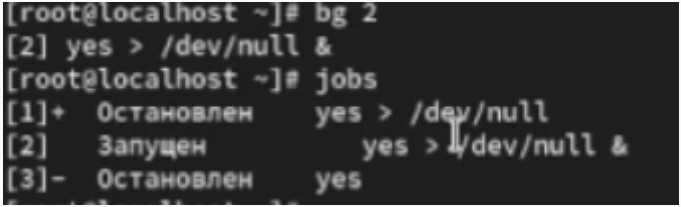



**Рис. 16:** 17

После закрываем окно и заново запускаем консоль

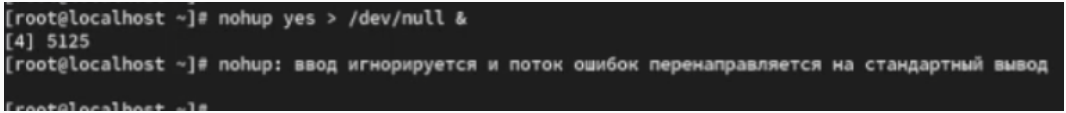
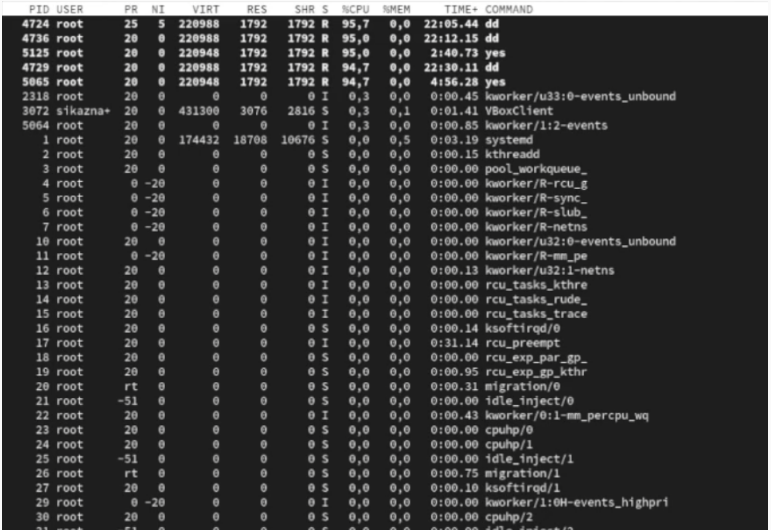



Рис. 17: 18

# Задание 10

И проверяем процессы.



```
  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  431300   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 S   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
```

Затем запустим три программы yes в фоновом режиме с подавлением потока вывода

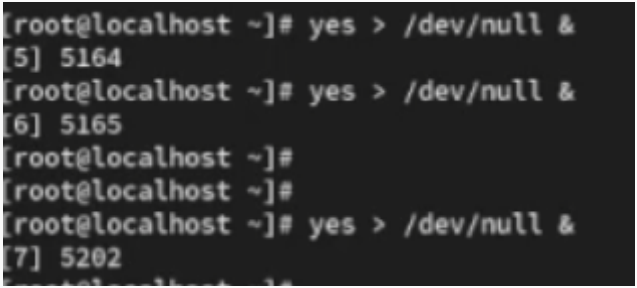



**Рис. 19:** 20

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

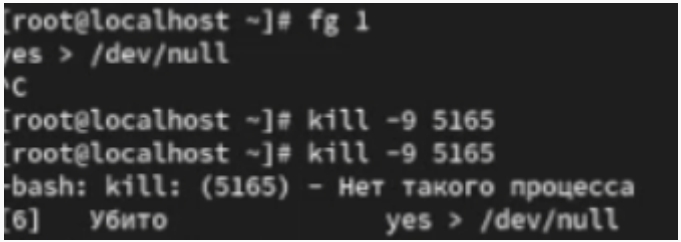



Рис. 20: 21

# Задание 13

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

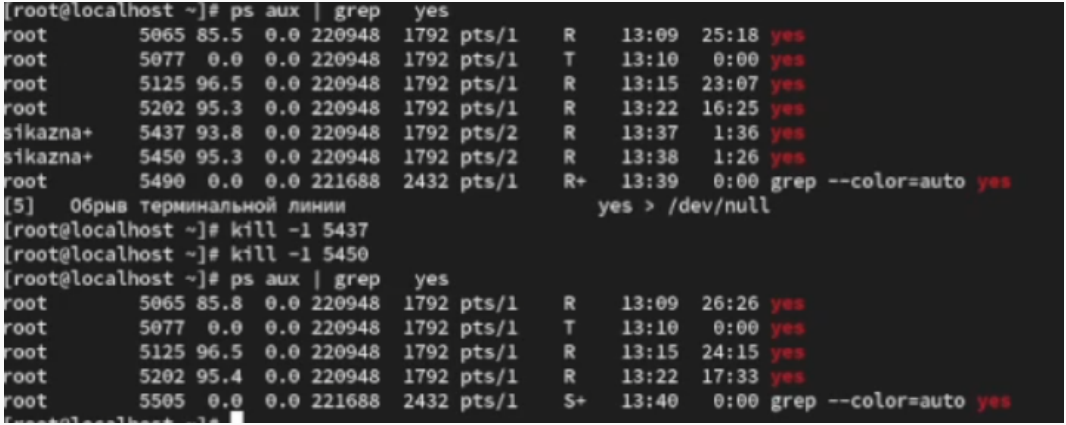
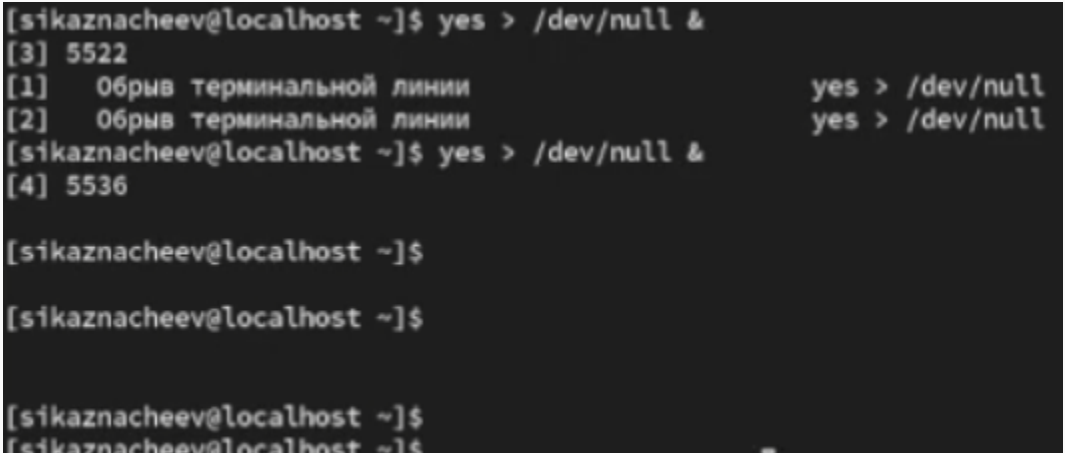



Рис. 21: 22

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



И убиваем все процессы yes с помощью команды killall yes

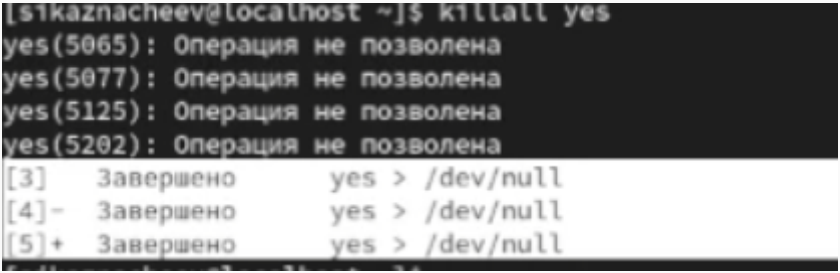



Рис. 23: 24

# Задание 16-17

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

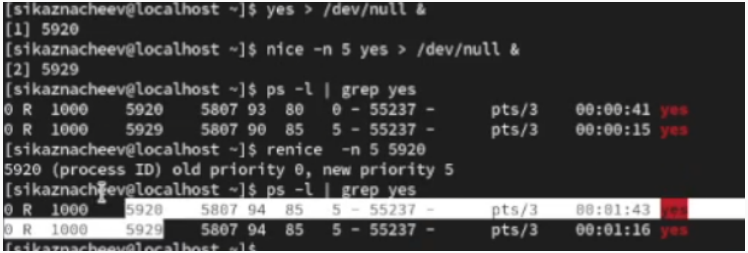



Рис. 24: 25

# Контрольный вопрос номер 1

1. Какая команда даёт обзор всех текущих заданий оболочки?

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

# Контрольный вопрос номер 2

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

Ответ - сначала приостанавливаем задание нажав на комбинаю клави Ctrl+Z затем отправляем его фон bg

# Контрольный вопрос номер 3

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

Ответ - комбинацию клавиш Ctrl+C

# Контрольный вопрос номер 4

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

# Контрольный вопрос номер 5

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

Ответ - комадна pstree

# Контрольный вопрос номер 6

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

# Контрольный вопрос номер 7

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

Ответ - команда killall dd

# Контрольный вопрос номер 8

8. Какая команда позволяет остановить команду с именем mycommand?

Ответ - команда pkill mycommand

# Контрольный вопрос номер 9

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

Ответ - во время работы top нажать на клавишу k

# Контрольный вопрос номер 10

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

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

# Вывод

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