

Отчёт по лабораторной работе №6

Управление процессами

Яковлева Дарья Сергеевна

Содержание

| | | |
|----------|---------------------------------------|-----------|
| 1 | Цель работы | 5 |
| 2 | Выполнение лабораторной работы | 6 |
| 2.1 | Управление заданиями | 6 |
| 2.2 | Управление процессами | 9 |
| 2.3 | Задание 1 | 11 |
| 2.4 | Задание 2 | 11 |
| 3 | Контрольные вопросы | 16 |
| 4 | Заключение | 18 |

Список иллюстраций

| | | |
|------|---|----|
| 2.1 | Запуск и управление заданиями | 7 |
| 2.2 | Работа с утилитой top | 8 |
| 2.3 | Завершение процесса dd | 9 |
| 2.4 | Запуск процессов dd | 9 |
| 2.5 | Изменение приоритета процесса | 10 |
| 2.6 | Просмотр иерархии процессов | 10 |
| 2.7 | Запуск процессов dd | 11 |
| 2.8 | Запуск программы yes | 12 |
| 2.9 | Управление заданиями yes | 12 |
| 2.10 | Процессы yes в top | 13 |
| 2.11 | Завершение процессов yes | 13 |
| 2.12 | Проверка работы с сигналами | 14 |
| 2.13 | Завершение процессов командой killall | 14 |

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Выполнение лабораторной работы

2.1 Управление заданиями

Получаю полномочия администратора с помощью `su` (см. рис. fig. 2.1).

Запускаю три задания:

- `sleep 3600 &` — первое задание в фоне;
- `dd if=/dev/zero of=/dev/null &` — второе задание в фоне;
- `sleep 7200` — третье задание, запущенное без `&`.

Поскольку третье задание блокирует терминал, прерываю его комбинацией **Ctrl+Z**, после чего проверяю список заданий с помощью `jobs` (см. рис. fig. 2.1).

```

dsyakovleva@dsyakovleva:~$ su
Password:
root@dsyakovleva:/home/dsyakovleva# sleep 3600 &
[1] 3405
root@dsyakovleva:/home/dsyakovleva#
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &
[2] 3531
root@dsyakovleva:/home/dsyakovleva# sleep 7200
^Z
[3]+  Stopped                  sleep 7200
root@dsyakovleva:/home/dsyakovleva# jobs
[1]  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Stopped                  sleep 7200
root@dsyakovleva:/home/dsyakovleva# bg 3
[3]+  sleep 7200 &
root@dsyakovleva:/home/dsyakovleva# jobs
[1]  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Running                  sleep 7200 &
root@dsyakovleva:/home/dsyakovleva# fg 1
sleep 3600
^C
root@dsyakovleva:/home/dsyakovleva# jobs
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Running                  sleep 7200 &
root@dsyakovleva:/home/dsyakovleva# fg 2
dd if=/dev/zero of=/dev/null
^C47736287+0 records in
47736286+0 records out
24440978432 bytes (24 GB, 23 GiB) copied, 95.5181 s, 256 MB/s

root@dsyakovleva:/home/dsyakovleva# fg 3
sleep 7200
^C
root@dsyakovleva:/home/dsyakovleva# █

```

Рис. 2.1: Запуск и управление заданиями

Вижу, что первые два задания находятся в состоянии *Running*, а третье — *Stopped*. Перевожу задание 3 в фоновый режим командой `bg 3`, снова проверяю список заданий через `jobs`. Теперь все три задания работают в фоне.

Переношу задание 1 на передний план с помощью `fg 1`, затем останавливаю его комбинацией **Ctrl+C**. Аналогично завершаю задания 2 и 3 (см. рис. fig. 2.1).

Открываю новый терминал от имени пользователя и запускаю в нём процесс `dd if=/dev/zero of=/dev/null &`.

Закрываю терминал командой `exit`.

В другом терминале под тем же пользователем запускаю утилиту `top`, где вижу, что процесс `dd` продолжает работать (см. рис. fig. 2.2).

| dsyakovleva@dsyakovleva:/home/dsyakovleva - top | | | | | | | | | | |
|--|----------|----|-----|---------|--------|--------|---|------|------|--|
| top - 14:22:36 up 6 min, 4 users, load average: 1.09, 1.01, 0.53 | | | | | | | | | | |
| Tasks: 290 total, 2 running, 288 sleeping, 0 stopped, 0 zombie | | | | | | | | | | |
| %Cpu(s): 10.8 us, 11.1 sy, 0.0 ni, 76.5 id, 0.0 wa, 1.6 hi, 0.1 si, 0.0 st | | | | | | | | | | |
| MiB Mem : 3908.5 total, 1205.0 free, 1519.6 used, 1421.4 buff/cache | | | | | | | | | | |
| MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2388.8 avail Mem | | | | | | | | | | |
| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ COMMAND |
| 3924 | dsyakov+ | 20 | 0 | 226848 | 1780 | 1780 | R | 97.0 | 0.0 | 0:11.89 dd |
| 2151 | dsyakov+ | 20 | 0 | 5481008 | 310600 | 122192 | S | 11.9 | 7.8 | 0:17.02 gnome-shell |
| 2972 | dsyakov+ | 20 | 0 | 3795996 | 325452 | 96200 | S | 10.9 | 8.1 | 0:09.65 ptyxis |
| 155 | root | 20 | 0 | 0 | 0 | 0 | I | 9.3 | 0.0 | 0:01.03 kworker/u29:3-events_unbound |
| 173 | root | 20 | 0 | 0 | 0 | 0 | I | 2.3 | 0.0 | 0:00.38 kworker/u25:2-events_unbound |
| 3957 | root | 20 | 0 | 231604 | 5436 | 3260 | R | 0.7 | 0.1 | 0:00.09 top |
| 8 | root | 20 | 0 | 0 | 0 | 0 | I | 0.3 | 0.0 | 0:00.06 kworker/0:0-mm_percpu_wq |
| 962 | root | 39 | 19 | 4292 | 2692 | 2436 | S | 0.3 | 0.1 | 0:00.02 alsactl |
| 1200 | root | 20 | 0 | 578492 | 3224 | 2960 | S | 0.3 | 0.1 | 0:00.25 VBoxService |
| 2615 | dsyakov+ | 9 | -11 | 322424 | 11992 | 7620 | S | 0.3 | 0.3 | 0:00.17 pipewire |
| 1 | root | 20 | 0 | 49192 | 40844 | 10144 | S | 0.0 | 1.0 | 0:07.47 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_release |
| 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_flushwq |
| 7 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 kworker/R-netns |
| 9 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.04 kworker/0:1-cgroup_destroy |
| 10 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 kworker/0:0H-events_highpri |
| 11 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 kworker/u24:0-events_unbound |
| 12 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.07 kworker/u24:1-netns |
| 13 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 kworker/R-mm_percpu_wq |
| 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_kthread |
| 16 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 rcu_tasks_trace_kthread |
| 17 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.01 ksoftirqd/0 |
| 18 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.30 rcu_preempt |
| 19 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 rcu_exp_par_gp_kthread_worker+ |
| 20 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.06 rcu_exp_gp_kthread_worker |

Рис. 2.2: Работа с утилитой top

Выход из top выполняю клавишей **q**. Затем снова запускаю top и использую клавишу **k** для завершения процесса dd. В списке процессов он больше не отображается (см. рис. fig. 2.3).


```

top - 14:23:04 up 6 min,  4 users,  load average: 2.03, 1.23, 0.62
Tasks: 288 total,  1 running, 287 sleeping,  0 stopped,  0 zombie
%Cpu(s): 11.0 us,  8.9 sy,  0.6 ni, 78.7 id,  0.0 wa,  0.8 hi,  0.1 si,  0.0 st
MiB Mem : 3908.5 total, 1205.3 free, 1518.9 used, 1421.9 buff/cache
MiB Swap: 4040.0 total, 4040.0 free,  0.0 used, 2389.6 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 2972 dsyakov+  20   0 379324 328812 96200 S 10.4   8.2   0:12.70  pttyxis
2151 dsyakov+  20   0 5481008 310600 122192 S  4.4   7.8   0:19.31  gnome-shell
  158 root      20   0      0      0      0 I  1.3   0.0   0:00.57  kworker/u28:3-events_unbound
    1 root      20   0  49192  40844 10144 S  0.5   1.0   0:07.68  systemd
4042 root      20   0 231604  5244  3196 R  0.3   0.1   0:00.10  top
   72 root      39  19      0      0      0 S  0.2   0.0   0:00.52  khugepaged
  206 root      20   0      0      0      0 I  0.2   0.0   0:00.88  kworker/u27:2-xfs-cil/dm-0
    20 root      20   0      0      0      0 S  0.1   0.0   0:00.07  rcu_exp_gp_kthread_worker
   33 root      20   0      0      0      0 S  0.1   0.0   0:00.01  ksoftirqd/2
  107 root      20   0      0      0      0 I  0.1   0.0   0:00.05  kworker/5:2-events
  173 root      20   0      0      0      0 I  0.1   0.0   0:00.39  kworker/u25:2-writeback
  709 root      20   0  25436  9604  8196 S  0.1   0.2   0:00.76  systemd-journal
  918 dbus      20   0   8080  5724  2376 S  0.1   0.1   0:01.00  dbus-broker
1198 root      20   0 574184  2520  2392 S  0.1   0.1   0:00.25  VBoxDRMClient
1256 root      20   0 488244 29384 14408 S  0.1   0.7   0:00.61  tuned
2669 dsyakov+  20   0 454692  6724  6212 S  0.1   0.2   0:00.28  ibus-engine-sim
2956 root      20   0 643416 22148 17796 S  0.1   0.6   0:00.36  fwupd
    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_release
    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_flushwq
    7 root      0 -20      0      0      0 I  0.0   0.0   0:00.00  kworker/R-netns
    8 root      20   0      0      0      0 I  0.0   0.0   0:00.06  kworker/0:0-rcu_gp
    9 root      20   0      0      0      0 I  0.0   0.0   0:00.04  kworker/0:1-cgroup_destroy
   10 root      0 -20      0      0      0 I  0.0   0.0   0:00.00  kworker/0:0H-events_highpri
   11 root      20   0      0      0      0 I  0.0   0.0   0:00.00  kworker/u24:0-events_unbound
   12 root      20   0      0      0      0 I  0.0   0.0   0:00.08  kworker/u24:1-netns
   13 root      0 -20      0      0      0 I  0.0   0.0   0:00.00  kworker/R-mm_percpu_wq

```

Рис. 2.3: Завершение процесса dd

2.2 Управление процессами

Получаю полномочия администратора с помощью su.

Запускаю три процесса dd, которые пишут поток данных из /dev/zero в /dev/null (см. рис. fig. 2.4).

```

root@dsyakovleva:/home/dsyakovleva#
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &
[1] 4518
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &
[2] 4556
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &
[3] 4558
root@dsyakovleva:/home/dsyakovleva# ps aux | grep dd
root      2   0.0  0.0      0  0 ?        S   14:16   0:00 [kthreadd]
root      12   0.0  0.0      0  0 ?        I   14:16   0:00 [kworker/u24:1-ipv6_addrconf]
root     109   0.0  0.0      0  0 ?        I<  14:16   0:00 [kworker/R-ipv6_addrconf]
root    1200   0.0  0.0 578492 3224 ?        Sl   14:16   0:00 /usr/sbin/VBoxService --pidfile /va
i/run/vboxadd-service.sh
dsyakov+ 2725   0.0  0.6 1036404 25008 ?        Ssl  14:17   0:00 /usr/libexec/evolution-addressbook-
factory
root     4518 98.5  0.0 226848 1780 pts/0    R   14:24   0:34 dd if=/dev/zero of=/dev/null
root     4556 98.6  0.0 226848 1792 pts/0    R   14:24   0:30 dd if=/dev/zero of=/dev/null
root     4558 98.4  0.0 226848 1672 pts/0    R   14:24   0:29 dd if=/dev/zero of=/dev/null
root     4628  0.0  0.0 227688 2036 pts/0    S+  14:24   0:00 grep --color=auto dd
root@dsyakovleva:/home/dsyakovleva#

```

Рис. 2.4: Запуск процессов dd

С помощью команды `ps aux | grep dd` просматриваю список процессов. Вижу три активных процесса `dd` с различными PID (см. рис. fig. 2.4).

Меняю приоритет одного из процессов, используя команду `renice -n 5 <PID>`. В результате приоритет изменяется с 0 на 5 (см. рис. fig. 2.5).

```
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &
[1] 4518
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &
[2] 4556
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &
[3] 4558
root@dsyakovleva:/home/dsyakovleva# ps aux | grep dd
root      2  0.0  0.0      0  0 ?        S   14:16   0:00 [kthreadd]
root     12  0.0  0.0      0  0 ?        I   14:16   0:00 [kworker/u24:1-ipv6_addrconf]
root     109  0.0  0.0      0  0 ?        I<  14:16   0:00 [kworker/R-ipv6_addrconf]
root    1200  0.0  0.0 578492 3224 ?        Sl  14:16   0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
dsyakov+ 2725  0.0  0.6 1036404 25008 ?        Ssl 14:17   0:00 /usr/libexec/evolution-addressbook-factory
root     4518 98.5  0.0 226848 1780 pts/0    R   14:24   0:34 dd if=/dev/zero of=/dev/null
root     4556 98.6  0.0 226848 1792 pts/0    R   14:24   0:30 dd if=/dev/zero of=/dev/null
root     4558 98.4  0.0 226848 1672 pts/0    R   14:24   0:29 dd if=/dev/zero of=/dev/null
root     4628  0.0  0.0 227688 2036 pts/0    S+  14:24   0:00 grep --color=auto dd
root@dsyakovleva:/home/dsyakovleva#
root@dsyakovleva:/home/dsyakovleva# renice -n 5 4518
4518 (process ID) old priority 0, new priority 5
root@dsyakovleva:/home/dsyakovleva#
```

Рис. 2.5: Изменение приоритета процесса

Для отображения иерархии процессов использую команду `ps fax | grep -B5 dd`. Вижу дерево выполнения, где процессы `dd` являются дочерними для корневой оболочки `bash`, запущенной из-под `su` (см. рис. fig. 2.6).

```
962 ?      SNs   0:00 /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf
initfile=/lib/alsa/init/00main rdaemon
995 ?      S      0:00 /usr/sbin/chronyd -F 2
1011 ?     Ssl    0:00 /usr/sbin/ModemManager
1032 ?     Ssl    0:01 /usr/bin/python3 -sP /usr/sbin/firewalld --nofork --nopid
1198 ?     Sl      0:00 /usr/bin/VBoxDRMClient
1200 ?     Sl      0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh

2562 ?     Ssl    0:00 \_ /usr/libexec/gvfs-goa-volume-monitor
2598 ?     Ssl    0:00 \_ /usr/libexec/evolution-calendar-factory
2615 ?     S<sl   0:00 \_ /usr/bin/pipewire
2616 ?     S<sl   0:00 \_ /usr/bin/wireplumber
2617 ?     S<sl   0:00 \_ /usr/bin/pipewire-pulse
2725 ?     Ssl    0:00 \_ /usr/libexec/evolution-addressbook-factory

2972 ?     Ssl    0:26 \_ /usr/bin/ptxis --gaplication-service
2984 ?     Ssl    0:00 | \_ /usr/libexec/ptxis-agent --socket-fd=3
3074 pts/0 Ss      0:00 | \_ /usr/bin/bash
3289 pts/0 S        0:00 | \_ su
3334 pts/0 S        0:00 | \_ bash
4518 pts/0 RN      2:57 | \_ dd if=/dev/zero of=/dev/null
4556 pts/0 R        2:53 | \_ dd if=/dev/zero of=/dev/null
4558 pts/0 R        2:52 | \_ dd if=/dev/zero of=/dev/null
4912 pts/0 R+       0:00 | \_ ps fax
4913 pts/0 S+       0:00 | \_ grep --color=auto -B5 dd
root@dsyakovleva:/home/dsyakovleva#
```

Рис. 2.6: Просмотр иерархии процессов

Нахожу PID родительского процесса `bash`, из которого были запущены процес-

сы dd, и завершаю его с помощью `kill -9 <PID>`. В результате оболочка закрывается, и все дочерние процессы dd останавливаются автоматически.

2.3 Задание 1

Запускаю три процесса dd, которые пишут данные из /dev/zero в /dev/null, в фоновом режиме (см. рис. fig. 2.7).

```
root@dsyakovleva:/home/dsyakovleva#  
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &  
[1] 5279  
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &  
[2] 5288  
root@dsyakovleva:/home/dsyakovleva# dd if=/dev/zero of=/dev/null &  
[3] 5293  
root@dsyakovleva:/home/dsyakovleva# renice -n 5 5288  
5288 (process ID) old priority 0, new priority 5  
root@dsyakovleva:/home/dsyakovleva# renice -n 15 5288  
5288 (process ID) old priority 5, new priority 15  
root@dsyakovleva:/home/dsyakovleva# 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@dsyakovleva:/home/dsyakovleva#
```

Рис. 2.7: Запуск процессов dd

Меняю приоритет одного из процессов на значение -5 с помощью команды `renice -n -5 <PID>`. В результате приоритет изменяется с 0 на 5.

Затем повторно изменяю приоритет этого же процесса, установив значение -15. Теперь приоритет изменяется с 5 на 15 (см. рис. fig. 2.7).

Для завершения всех процессов dd использую команду `killall dd`. Вижу сообщение о завершении трёх процессов (см. рис. fig. 2.7).

2.4 Задание 2

Запускаю программу `yes` в фоновом режиме с подавлением потока вывода. Аналогично запускаю её несколько раз на переднем плане, приостанавливаю выполнение комбинацией **Ctrl+Z**, затем перевожу процессы в фон с помощью `bg` и завершаю их при необходимости (см. рис. fig. 2.8).

```

root@dsyakovleva:/home/dsyakovleva#
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[1] 5711
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[2] 5713
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null
^Z
[3]+  Stopped                  yes > /dev/null
root@dsyakovleva:/home/dsyakovleva# fg 3
yes > /dev/null
^C
root@dsyakovleva:/home/dsyakovleva# █

```

Рис. 2.8: Запуск программы yes

С помощью команды `jobs` проверяю состояния заданий. Вижу, что процессы находятся в состоянии *Running*. Один из процессов перевожу на передний план командой `fg`, затем завершаю его (см. рис. fig. 2.9).

```

root@dsyakovleva:/home/dsyakovleva# jobs
[1]-  Running                  yes > /dev/null &
[2]+  Running                  yes > /dev/null &
root@dsyakovleva:/home/dsyakovleva# fg 2
yes > /dev/null
^C
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null
^Z
[2]+  Stopped                  yes > /dev/null
root@dsyakovleva:/home/dsyakovleva# bg 2
[2]+  yes > /dev/null &
root@dsyakovleva:/home/dsyakovleva# jobs
[1]-  Running                  yes > /dev/null &
[2]+  Running                  yes > /dev/null &
root@dsyakovleva:/home/dsyakovleva# nohup /dev/null &
[3] 6059
nohup: ignoring input and appending output to 'nohup.out'
nohup: root@dsyakovleva:/home/dsyakovleva# failed to run command '/dev/null': Permission denied

[3]+  Exit 126                  nohup /dev/null
root@dsyakovleva:/home/dsyakovleva# nohup yes > /dev/null &
[3] 6083
nohup: ignoring input and redirecting stderr to stdout
root@dsyakovleva:/home/dsyakovleva# █

```

Рис. 2.9: Управление заданиями yes

Для запуска процесса, продолжающего работу после выхода из терминала, использую `nohup yes > /dev/null &`. Проверяю результат через `jobs`, где видно, что процесс перенаправил вывод в файл `nohup.out`. После закрытия терминала и повторного входа убеждаюсь, что процесс остался активным (см. рис. fig. 2.9).

Получаю информацию о процессах с помощью утилиты `top`. Наглядно видно, что процессы `yes` потребляют значительные ресурсы процессора (см. рис. fig. 2.10).

```

top - 14:37:38 up 21 min, 5 users, load average: 3.31, 2.35, 1.65
Tasks: 287 total, 4 running, 283 sleeping, 0 stopped, 0 zombie
%Cpu(s): 26.2 us, 29.7 sy, 0.8 ni, 41.9 id, 0.0 wa, 1.3 hi, 0.1 si, 0.0 st
MiB Mem : 3908.5 total, 1490.1 free, 1228.3 used, 1427.9 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2680.1 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 5993 root        20   0 226820 1796 1796 R  98.4   0.0   1:48.31 yes
 5711 root        20   0 226820 1744 1744 R  98.0   0.0   4:08.56 yes
 6083 root        20   0 226820 1724 1724 R  98.0   0.0   1:09.11 yes
2151 dsyakov+   20   0 5485912 317876 122172 S 12.2   7.9   0:56.49 gnome-shell
6142 dsyakov+  20   0 3983964 212364 96516 S  9.2   5.3   0:04.01 ptxixis
5620 root        20   0      0      0      0 I  6.6   0.0   0:01.22 kworker/u27:1-flush-253:0
173  root        20   0      0      0      0 I  3.0   0.0   0:01.35 kworker/u25:2-events_unbound
55  root        20   0      0      0      0 I  1.0   0.0   0:02.47 kworker/u26:0-flush-253:0
 6366 root        20   0 231604 5476 3300 R   1.0   0.1   0:00.20 top
   1  root        20   0  49192  40844 10144 S   0.7   1.0   0:13.33 systemd
  18  root        20   0      0      0      0 I   0.3   0.0   0:00.99 rcu_preempt
2017 root        20   0      0      0      0 I   0.3   0.0   0:03.02 kworker/u26:3-events_unbound
   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_release
   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_flushwq
   7  root        0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
   8  root        20   0      0      0      0 I   0.0   0.0   0:00.16 kworker/0:0-events
  10  root        0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
  11  root        20   0      0      0      0 I   0.0   0.0   0:00.00 kworker/u24:0-events_unbound
  12  root        20   0      0      0      0 I   0.0   0.0   0:00.29 kworker/u24:1-netns
  13  root        0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-mm_percpu_wq
  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_kthread
  16  root        20   0      0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
  17  root        20   0      0      0      0 S   0.0   0.0   0:00.02 ksoftirqd/0
  19  root        20   0      0      0      0 S   0.0   0.0   0:00.00 rcu_exp_par_gp_kthread_worker+
  20  root        20   0      0      0      0 S   0.0   0.0   0:00.34 rcu_exp_gp_kthread_worker

```

Рис. 2.10: Процессы yes в top

Запускаю ещё три процесса yes в фоне. Завершаю один из них по PID, а второй — по идентификатору задания. Использую команду kill и наблюдаю завершение процессов (см. рис. fig. 2.11).

```

root@dsyakovleva:/home/dsyakovleva#
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[1] 6568
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[2] 6621
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[3] 6632
root@dsyakovleva:/home/dsyakovleva# kill 6621
[2]-  Terminated                  yes > /dev/null
root@dsyakovleva:/home/dsyakovleva# fg 1
yes > /dev/null
^C
root@dsyakovleva:/home/dsyakovleva#

```

Рис. 2.11: Завершение процессов yes

Далее отправляю сигнал SIGHUP обычному процессу и процессу, запущенному с помощью nohup. При этом обычный процесс завершается, а процесс под nohup продолжает работу (см. рис. fig. 2.12).

```

root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[1] 6568
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[2] 6621
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[3] 6632
root@dsyakovleva:/home/dsyakovleva# kill 6621
[2]- Terminated          yes > /dev/null
root@dsyakovleva:/home/dsyakovleva# fg 1
yes > /dev/null
^C
root@dsyakovleva:/home/dsyakovleva#
root@dsyakovleva:/home/dsyakovleva# kill -1 5993
root@dsyakovleva:/home/dsyakovleva# kill -1 5711
root@dsyakovleva:/home/dsyakovleva# kill -1 6083
root@dsyakovleva:/home/dsyakovleva#
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[4] 6825
root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[5] 6827
root@dsyakovleva:/home/dsyakovleva# killall yes
[3] Terminated          yes > /dev/null
[4]- Terminated          yes > /dev/null
[5]+ Terminated          yes > /dev/null
root@dsyakovleva:/home/dsyakovleva# █

```

Рис. 2.12: Проверка работы с сигналами

Запускаю несколько процессов `yes` в фоне, а затем завершаю их одновременно командой `killall yes` (см. рис. fig. 2.13).

```

root@dsyakovleva:/home/dsyakovleva# yes > /dev/null &
[1] 6939
root@dsyakovleva:/home/dsyakovleva# nice -n 5 yes > /dev/null &
[2] 7064
root@dsyakovleva:/home/dsyakovleva# ps -l
F S  UID      PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0        6243    6205  0  80   0 - 58153 do_wai pts/1      00:00:00 su
4 S   0        6265    6243  0  80   0 - 57575 do_wai pts/1      00:00:00 bash
4 R   0        6939    6265  97  80   0 - 56705 -      pts/1      00:01:03 yes
4 R   0        7064    6265  97  85   5 - 56705 -      pts/1      00:00:03 yes
4 R   0        7066    6265  99  80   0 - 57682 -      pts/1      00:00:00 ps
root@dsyakovleva:/home/dsyakovleva# renice -n 5 6939
6939 (process ID) old priority 0, new priority 5
root@dsyakovleva:/home/dsyakovleva# ps -l
F S  UID      PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0        6243    6205  0  80   0 - 58153 do_wai pts/1      00:00:00 su
4 S   0        6265    6243  0  80   0 - 57575 do_wai pts/1      00:00:00 bash
4 R   0        6939    6265  97  85   5 - 56705 -      pts/1      00:01:41 yes
4 R   0        7064    6265  97  85   5 - 56705 -      pts/1      00:00:42 yes
4 R   0        7156    6265  99  80   0 - 57682 -      pts/1      00:00:00 ps
root@dsyakovleva:/home/dsyakovleva# killall yes
[1]- Terminated          yes > /dev/null
[2]+ Terminated          nice -n 5 yes > /dev/null
root@dsyakovleva:/home/dsyakovleva# █

```

Рис. 2.13: Завершение процессов командой `killall`

Запускаю процесс `yes` с повышенным приоритетом при помощи `nice -n -5`. Сравниваю приоритеты процессов через `ps -l`: вижу различие в значениях. Затем с помощью `renice` выравниваю приоритеты, установив одинаковое значение

для обоих процессов (см. рис. fig. 2.13).

3 Контрольные вопросы

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

Используется команда `jobs`.

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

Используется комбинация клавиш **Ctrl+Z**, затем команда `bg`.

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

Для завершения задания применяется комбинация **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?**

Использовать команду `kill <PID>` из другой оболочки или завершить родительский процесс, от которого оно запущено.

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

Команда `ps fax`.

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

Команда `renice -n -5 -p 1234`.

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

Использовать команду `killall dd`.

8. **Какая команда позволяет остановить команду с именем `mycommand`?**

Используется команда `killall mycommand`.

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

Внутри `top` используется клавиша **k** и вводится PID процесса.

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

Используется команда `nice`, например: `nice -n 10 mycommand`.

4 Заключение

В ходе лабораторной работы были приобретены практические навыки управления заданиями и процессами в Linux: запуск программ в фоновом и переднем режиме, приостановка и возобновление выполнения, завершение отдельных процессов и групп процессов. Освоены приёмы работы с приоритетами процессов с помощью утилит `nice` и `renice`, а также использование команд `jobs`, `ps`, `top`, `kill` и `killall`. Полученные знания позволяют эффективно контролировать работу процессов и управлять распределением ресурсов в системе.