

Отчёт по лабораторной работе №6

Управление процессами

Амина Аджигалиева

Содержание

1	Цель работы	5
2	Ход выполнения работы	6
2.1	Управление заданиями	6
2.2	Управление процессами	9
2.3	Задание 1. Управление приоритетами процессов	11
3	Ход выполнения работы	12
3.1	Задание 2. Управление процессами yes	12
4	Контрольные вопросы	16
5	Заключение	18

Список иллюстраций

2.1	Запуск и просмотр фоновых заданий	7
2.2	Управление заданиями с помощью fg и bg	8
2.3	Мониторинг процесса dd с помощью top	9
2.4	Просмотр процессов dd через ps aux	10
2.5	Отображение иерархии процессов через ps fax	10
2.6	Управление приоритетами и завершение процессов dd	11
3.1	Запуск и остановка процессов yes	12
3.2	Перевод процессов между фоном и передним планом	13
3.3	Мониторинг процессов yes через top	13
3.4	Завершение процессов по PID и job ID	14
3.5	Изменение приоритета	15

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Ход выполнения работы

2.1 Управление заданиями

Сначала я получила права суперпользователя с помощью команды `su`.
После этого были запущены несколько процессов:

- команда `sleep 3600 &` запустила задачу в фоновом режиме,
- команда `dd if=/dev/zero of=/dev/null &` создала ещё один фоновый процесс,
- команда `sleep 7200` была запущена без символа `&`, поэтому процесс занял терминал.

Для остановки выполнения последнего процесса я использовала сочетание клавиш **Ctrl+Z**.

Затем с помощью команды `jobs` проверила список активных заданий.

```

aradzhigalieva@aradzhigalieva:~$ su
Password:
root@aradzhigalieva:/home/aradzhigalieva# sleep 3600 $
sleep: invalid time interval '$'
Try 'sleep --help' for more information.
root@aradzhigalieva:/home/aradzhigalieva# sleep 3600 &
[1] 3535
root@aradzhigalieva:/home/aradzhigalieva# dd if=/dev/zero of=/dev/null &
[2] 3582
root@aradzhigalieva:/home/aradzhigalieva# sleep 7200
^Z
[3]+  Stopped                  sleep 7200
root@aradzhigalieva:/home/aradzhigalieva# jobs
[1]  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Stopped                  sleep 7200
root@aradzhigalieva:/home/aradzhigalieva# bg 3
[3]+ sleep 7200 &
root@aradzhigalieva:/home/aradzhigalieva# fg 1
sleep 3600
^C
root@aradzhigalieva:/home/aradzhigalieva# fg 2
dd if=/dev/zero of=/dev/null
^C124140364+0 records in
124140364+0 records out
63559866368 bytes (64 GB, 59 GiB) copied, 82.7918 s, 768 MB/s

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

```

Рис. 2.1: Запуск и просмотр фоновых заданий

Далее я перевела задание №3 в фоновый режим с помощью команды `bg 3`. После этого выполнила последовательное переключение процессов на передний план:

- `fg 1` — процесс был остановлен с помощью **Ctrl+C**,
- `fg 2` — также был завершён через **Ctrl+C**,
- `fg 3` — аналогично остановлен.

```

top - 15:32:48 up 8 min, 4 users, load average: 0.29, 0.33, 0.18
Tasks: 262 total, 2 running, 260 sleeping, 0 stopped, 0 zombie
%Cpu(s): 9.5 us, 16.7 sy, 0.0 ni, 73.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3909.0 total, 1249.0 free, 1477.5 used, 1421.3 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2431.5 avail Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4061	aradzhi+	20	0	226848	1752	1752	R	91.7	0.0	0:07.48	dd
1	root	20	0	49192	41240	10316	S	0.0	1.0	0:01.24	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	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
9	root	20	0	0	0	0	I	0.0	0.0	0:00.05	kworker/0:1-rcu_gp
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/u16:0-events_unbound
12	root	20	0	0	0	0	I	0.0	0.0	0:00.03	kworker/u16:1-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	0:00.00	ksoftirqd/0
18	root	20	0	0	0	0	I	0.0	0.0	0:00.05	rcu_preempt
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_exp_par_gp_kthread_worker/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.01	rcu_exp_gp_kthread_worker
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

Рис. 2.2: Управление заданиями с помощью fg и bg

В другом терминале под своей учётной записью я снова запустила процесс `dd if=/dev/zero of=/dev/null &`.

Затем закрыла сессию командой `exit`.

После этого в новом терминале открыла монитор процессов `top` и убедилась, что процесс `dd` продолжает выполняться.


```

top - 15:33:01 up 9 min, 4 users, load average: 0.45, 0.37, 0.19
Tasks: 260 total, 1 running, 259 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.9 us, 6.8 sy, 0.0 ni, 86.9 id, 0.0 wa, 0.3 hi, 0.1 si, 0.0 st
MiB Mem : 3909.0 total, 1430.7 free, 1301.0 used, 1416.2 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2608.0 avail Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3163	aradzhi+	20	0	3027976	304540	100004	S	3.1	7.6	0:02.62	ptxvis
2118	aradzhi+	20	0	5045944	312020	122164	S	2.2	7.8	0:03.59	gnome-shell
3893	root	20	0	0	0	0	I	0.5	0.0	0:00.03	kworker/u19:1-events_unbound
676	root	20	0	41864	10864	9328	S	0.2	0.3	0:00.19	systemd-journal
2887	aradzhi+	20	0	903456	59000	47236	S	0.2	1.5	0:00.09	xdg-desktop-por
1	root	20	0	49192	41240	10316	S	0.0	1.0	0:01.24	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	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	0:00.05	kworker/0:1-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/u16:0-events_unbound
12	root	20	0	0	0	0	I	0.0	0.0	0:00.03	kworker/u16:1-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	0:00.00	ksoftirqd/0
18	root	20	0	0	0	0	I	0.0	0.0	0:00.05	rcu_preempt
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_exp_par_gp_kthread_worker/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.01	rcu_exp_gp_kthread_worker
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0

Рис. 2.3: Мониторинг процесса dd с помощью top

Для завершения процесса я снова запустила `top`, использовала клавишу `k` и вручную завершила задание `dd`. После этого вышла из `top` с помощью клавиши `q`.

2.2 Управление процессами

Сначала я получила права администратора с помощью команды `su`. Затем последовательно запустила три процесса:

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

После этого я выполнила команду `ps aux | grep dd`, чтобы просмотреть все процессы, содержащие строку `dd`.

В выводе отобразились три запущенных процесса с их PID.

```

root@aradzhigalieva:/home/aradzhigalieva#
root@aradzhigalieva:/home/aradzhigalieva# dd if=/dev/zero of=/dev/null &
[1] 4449
root@aradzhigalieva:/home/aradzhigalieva# dd if=/dev/zero of=/dev/null &
[2] 4451
root@aradzhigalieva:/home/aradzhigalieva# dd if=/dev/zero of=/dev/null &
[3] 4453
root@aradzhigalieva:/home/aradzhigalieva# ps aux | grep dd
bash: ps: command not found...
root@aradzhigalieva:/home/aradzhigalieva# ps aux | grep dd
root      2  0.0  0.0      0  0 ?        S   15:24   0:00 [kthreadd]
root     92  0.0  0.0      0  0 ?        I<  15:24   0:00 [kworker/R-ipv6_addrconf]
root    1147  0.0  0.0 578492 3080 ?        Sl   15:24   0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-se
rvice.sh
aradzhi+ 2512  0.0  0.6 962676 25508 ?        Ssl  15:26   0:00 /usr/libexec/evolution-addressbook-factory
root    4449 98.9  0.0 226848 1808 pts/0    R   15:35   0:23 dd if=/dev/zero of=/dev/null
root    4451 99.2  0.0 226848 1780 pts/0    R   15:35   0:22 dd if=/dev/zero of=/dev/null
root    4453 98.7  0.0 226848 1752 pts/0    R   15:35   0:21 dd if=/dev/zero of=/dev/null
root    4515  0.0  0.0 227688 2180 pts/0    S+  15:35   0:00 grep --color=auto dd
root@aradzhigalieva:/home/aradzhigalieva#

```

Рис. 2.4: Просмотр процессов dd через ps aux

Затем я изменила приоритет одного из процессов dd с помощью команды `renice -n 5 <PID>`.

Для анализа иерархии процессов я использовала команду `ps fax | grep -B5 dd`. Это позволило увидеть не только процессы dd, но и их родительскую оболочку.

```

1147 ?      Sl      0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
--
2447 ?      Ssl     0:00 \_ /usr/libexec/gvfs-mtp-volume-monitor
2463 ?      Ssl     0:00 \_ /usr/libexec/gvfs-gphoto2-volume-monitor
2472 ?      Ssl     0:00 \_ /usr/libexec/evolution-calendar-factory
2478 ?      Ssl     0:00 \_ /usr/libexec/goa-identity-service
2493 ?      Ssl     0:00 \_ /usr/libexec/gvfs-goa-volume-monitor
2512 ?      Ssl     0:00 \_ /usr/libexec/evolution-addressbook-factory
--
3163 ?      Ssl     0:04 \_ /usr/bin/ptxis --gaplication-service
3174 ?      Ssl     0:00 | \_ /usr/libexec/ptxis-agent --socket-fd=3
3243 pts/0  Ss      0:00 | \_ /usr/bin/bash
3445 pts/0  S        0:00 | | \_ su
3479 pts/0  S        0:00 | | | \_ bash
4449 pts/0  R        1:33 | | | \_ dd if=/dev/zero of=/dev/null
4451 pts/0  RN       1:31 | | | \_ dd if=/dev/zero of=/dev/null
4453 pts/0  R        1:30 | | | \_ dd if=/dev/zero of=/dev/null
4667 pts/0  R+       0:00 | | | \_ ps fax
4668 pts/0  S+       0:00 | | | \_ grep --color=auto -B5 dd
root@aradzhigalieva:/home/aradzhigalieva#

```

Рис. 2.5: Отображение иерархии процессов через ps fax

После определения PID оболочки, из которой были запущены процессы dd, я применила команду `kill -9 <PID>`.

В результате оболочка завершила работу, а вместе с ней были остановлены все дочерние процессы dd.

2.3 Задание 1. Управление приоритетами процессов

Сначала я запустила три процесса с помощью команды `dd if=/dev/zero of=/dev/null &`, чтобы они выполнялись в фоновом режиме.

Далее изменила приоритет одного из процессов с помощью команды `renice -n -5 <PID>`.

После этого снова применила `renice`, но уже с параметром `-15`.

Разница заключается в том, что чем меньше значение `nice` (и больше отрицательное), тем выше приоритет процесса.

Таким образом, процесс с `-15` получает больше процессорного времени по сравнению с тем же процессом при значении `-5`.

В завершение я остановила все процессы `dd`. Сначала команда `killall dd` была выполнена без прав суперпользователя и вернула сообщение об ошибке «Operation not permitted».

После получения полномочий администратора через `su` команда `killall dd` успешно завершила все запущенные процессы `dd`.

```
aradzhigalieva@aradzhigalieva:~$  
aradzhigalieva@aradzhigalieva:~$ dd if=/dev/zero of=/dev/null &  
[1] 4858  
aradzhigalieva@aradzhigalieva:~$ dd if=/dev/zero of=/dev/null &  
[2] 4870  
aradzhigalieva@aradzhigalieva:~$ dd if=/dev/zero of=/dev/null &  
[3] 4872  
aradzhigalieva@aradzhigalieva:~$ renice -n 5 4858  
4858 (process ID) old priority 0, new priority 5  
aradzhigalieva@aradzhigalieva:~$ renice -n 15 4858  
4858 (process ID) old priority 5, new priority 15  
aradzhigalieva@aradzhigalieva:~$ killall dd  
dd(4449): Operation not permitted  
dd(4451): Operation not permitted  
dd(4453): Operation not permitted  
[2]- Terminated          dd if=/dev/zero of=/dev/null  
[1]- Terminated          dd if=/dev/zero of=/dev/null  
[3]+ Terminated          dd if=/dev/zero of=/dev/null  
aradzhigalieva@aradzhigalieva:~$ su  
Password:  
root@aradzhigalieva:/home/aradzhigalieva# killall dd  
root@aradzhigalieva:/home/aradzhigalieva#
```

Рис. 2.6: Управление приоритетами и завершение процессов `dd`

3 Ход выполнения работы

3.1 Задание 2. Управление процессами yes

Сначала я запустила программу `yes` в фоновом режиме с перенаправлением вывода в `/dev/null`.

Затем второй процесс `yes` был запущен на переднем плане и остановлен с помощью **Ctrl+Z**, после чего переведён в фоновый режим.

Проверка состояний через `jobs` показала наличие двух процессов: один в статусе **Running**, другой в статусе **Stopped**.

```
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null &
[1] 5330
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null
^Z
[2]+  Stopped                  yes > /dev/null
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null
^C
root@aradzhigalieva:/home/aradzhigalieva# jobs
[1]-  Running                  yes > /dev/null &
[2]+  Stopped                  yes > /dev/null
root@aradzhigalieva:/home/aradzhigalieva# █
```

Рис. 3.1: Запуск и остановка процессов `yes`

Далее я использовала команду `fg 1`, чтобы вернуть первый процесс на передний план и завершить его с помощью **Ctrl+C**.

Процесс со статусом **Stopped** был переведён в фоновый режим с помощью `bg 2`. После этого оба процесса находились в состоянии **Running**.

```

root@aradzhigalieva:/home/aradzhigalieva# jobs
[1]-  Running                  yes > /dev/null &
[2]+  Stopped                  yes > /dev/null
root@aradzhigalieva:/home/aradzhigalieva# fg 1
yes > /dev/null
^C
root@aradzhigalieva:/home/aradzhigalieva#
root@aradzhigalieva:/home/aradzhigalieva# jobs
[2]+  Stopped                  yes > /dev/null
root@aradzhigalieva:/home/aradzhigalieva# bg 2
[2]+ yes > /dev/null &
root@aradzhigalieva:/home/aradzhigalieva# jobs
[2]+  Running                  yes > /dev/null &
root@aradzhigalieva:/home/aradzhigalieva# nohup yes > /dev/null &
[3] 5610
nohup: ignoring input and redirecting stderr to stdout
root@aradzhigalieva:/home/aradzhigalieva# jobs
[2]-  Running                  yes > /dev/null &
[3]+  Running                  nohup yes > /dev/null &
root@aradzhigalieva:/home/aradzhigalieva#

```

Рис. 3.2: Перевод процессов между фоном и передним планом

Для того чтобы процесс продолжал выполняться даже после выхода из терминала, я запустила его с использованием `nohup yes > /dev/null &`.

Проверка через `jobs` подтвердила, что процесс выполняется.

Далее я открыла монитор процессов `top` и убедилась, что процессы `yes` продолжают работать и загружают процессор.

```

top - 15:44:15 up 20 min,  5 users,  load average: 1.63, 1.42, 0.94
Tasks: 267 total,   3 running, 264 sleeping,   0 stopped,   0 zombie
%Cpu(s): 14.3 us, 28.6 sy,   0.0 ni, 57.1 id,   0.0 wa,   0.0 hi,   0.0 si,   0.0 st
MiB Mem : 3909.0 total, 1323.8 free, 1398.2 used, 1426.2 buff/cache
MiB Swap: 4040.0 total, 4040.0 free,   0.0 used, 2510.8 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 5376 root        20   0 226820 1676 1676 R  90.9   0.0   1:11.03 yes
 5610 root        20   0 226820 1700 1700 R  81.8   0.0   0:39.04 yes
    1 root        20   0 49192 41240 10316 S   0.0   1.0   0:01.86 systemd
    2 root        20   0    0    0    0 S   0.0   0.0   0:00.00 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   0:00.10 kworker/0:1-ata_sff
   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/u16:0-events_unbound
   12 root        20   0    0    0    0 I   0.0   0.0   0:00.07 kworker/u16:1-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   0:00.12 rcu_preempt
   19 root        20   0    0    0    0 S   0.0   0.0   0:00.00 rcu_exp_par_gp_kthread_worker/0
   20 root        20   0    0    0    0 S   0.0   0.0   0:00.04 rcu_exp_gp_kthread_worker
   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

```

Рис. 3.3: Мониторинг процессов `yes` через `top`

После этого я запустила ещё три процесса `yes` и завершила два из них разными способами:

один процесс был остановлен по его PID, а другой — по идентификатору задания. Также я протестировала отправку сигнала **SIGHUP** процессам, запущенным как обычным способом, так и через `nohup`.

```
root@aradzhigalieva: /home/aradzhigalieva#  
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null &  
[1] 5975  
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null &  
[2] 5979  
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null &  
[3] 5982  
root@aradzhigalieva:/home/aradzhigalieva# kill 5975  
root@aradzhigalieva:/home/aradzhigalieva# fg 2  
yes > /dev/null  
^C  
[1] Terminated yes > /dev/null  
root@aradzhigalieva:/home/aradzhigalieva# jobs  
[3]+ Running yes > /dev/null &  
root@aradzhigalieva:/home/aradzhigalieva# kill -1 5982  
[3]+ Hup yes > /dev/null  
root@aradzhigalieva:/home/aradzhigalieva# kill -1 5376  
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null &  
[1] 6093  
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null &  
[2] 6095  
root@aradzhigalieva:/home/aradzhigalieva# killall yes  
[2]+ Terminated yes > /dev/null  
[1]+ Terminated yes > /dev/null  
root@aradzhigalieva:/home/aradzhigalieva# █
```

Рис. 3.4: Завершение процессов по PID и job ID

В конце я запустила несколько процессов `yes` в фоновом режиме с подавлением вывода и завершила их все одновременно с помощью `killall yes`.

Также я применила `nice`, чтобы запустить процесс с приоритетом выше на 5, и сравнила приоритеты через `ps -l`.

Затем изменила приоритет уже запущенного процесса с помощью `renice`.

Это позволило на практике убедиться в различиях между абсолютными и относительными приоритетами процессов.

```

root@aradzhigalieva:/home/aradzhigalieva#
root@aradzhigalieva:/home/aradzhigalieva# yes > /dev/null &
[1] 6430
root@aradzhigalieva:/home/aradzhigalieva# nice -n 5 yes > /dev/null &
[2] 6454
root@aradzhigalieva:/home/aradzhigalieva# ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0        5698    5675  0  80   0 - 58153 do_wai pts/1      00:00:00 su
4 S   0        5720    5698  0  80   0 - 57575 do_wai pts/1      00:00:00 bash
4 R   0        6430    5720 99  80   0 - 56705 -      pts/1      00:00:11 yes
4 R   0        6454    5720 99  85   5 - 56705 -      pts/1      00:00:02 yes
4 R   0        6456    5720  0  80   0 - 57682 -      pts/1      00:00:00 ps
root@aradzhigalieva:/home/aradzhigalieva# renice -n 5 6430
6430 (process ID) old priority 0, new priority 5
root@aradzhigalieva:/home/aradzhigalieva# ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0        5698    5675  0  80   0 - 58153 do_wai pts/1      00:00:00 su
4 S   0        5720    5698  0  80   0 - 57575 do_wai pts/1      00:00:00 bash
4 R   0        6430    5720 99  85   5 - 56705 -      pts/1      00:00:23 yes
4 R   0        6454    5720 98  85   5 - 56705 -      pts/1      00:00:14 yes
4 R   0        6495    5720  0  80   0 - 57682 -      pts/1      00:00:00 ps
root@aradzhigalieva:/home/aradzhigalieva# killall yes
[1]-  Terminated                  yes > /dev/null
[2]+  Terminated                  nice -n 5 yes > /dev/null
root@aradzhigalieva:/home/aradzhigalieva# █

```

Рис. 3.5: Изменение приоритета

4 Контрольные вопросы

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

Для просмотра используется команда `jobs`.

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

Нужно нажать **Ctrl+Z**, а затем выполнить команду `bg`.

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

Для завершения используется 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?**

Можно воспользоваться командой `kill <PID>` или `killall <имя_процесса>` в другой оболочке.

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

Для этого подходит команда `ps fax`.

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

Следует выполнить: `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 `myscommand`?**

Для этого используется `killall myscommand`.

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

Внутри `top` необходимо нажать **k**, ввести PID и подтвердить.

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

Запуск осуществляется с помощью `nice`. Например: `nice -n 10 команда` — чтобы процесс имел более низкий приоритет по сравнению с другими.

5 Заключение

В ходе выполнения лабораторной работы я освоила базовые приёмы управления заданиями и процессами в Linux.

Я научилась запускать процессы в фоновом и переднем режиме, приостанавливать и возобновлять их работу, изменять приоритет с помощью `nice` и `renice`, а также завершать процессы через `kill`, `killall` и утилиту `top`.

Кроме того, я закрепила навыки анализа иерархии процессов с использованием команды `ps` и убедилась в том, что завершение родительского процесса приводит к остановке всех его дочерних задач.

Полученные знания позволяют более эффективно администрировать систему, управлять её ресурсами и обеспечивать стабильную работу в многозадачной среде.