

Отчёт по лабораторной работе №6

Управление процессами

Анна Саенко

Содержание

1	Цель работы	5
2	Ход выполнения работы	6
2.1	Управление заданиями в Linux	6
2.2	Управление процессами в Linux	9
2.3	Задание 1.	10
2.4	Задание 2.	11
3	Контрольные вопросы	16
4	Заключение	18

Список иллюстраций

2.1	Перемещение заданий в передний план и завершение	7
2.2	Отображение процесса dd в top	8
2.3	Завершение процесса dd через top	8
2.4	Просмотр процессов dd через ps aux	9
2.5	Дерево процессов dd через ps fax	10
2.6	Запуск процессов dd, изменение приоритетов и завершение	11
2.7	Запуск yes и просмотр заданий	12
2.8	Перезапуск процесса yes и использование nohup	12
2.9	Просмотр процессов yes через top	13
2.10	Завершение процессов yes разными методами	14
2.11	Сравнение приоритетов yes и изменение их через renice	15

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Ход выполнения работы

2.1 Управление заданиями в Linux

Сначала я получила права суперпользователя с помощью команды `su -`.

Затем я запустила несколько процессов:

- `sleep 3600 &` – процесс ожидания на 1 час в фоновом режиме
- `dd if=/dev/zero of=/dev/null &` – утилита для теста скорости ввода-вывода
- `sleep 7200` – процесс ожидания на 2 часа, который изначально запустился на переднем плане

Поскольку последняя команда блокировала терминал, я остановила её комбинацией **Ctrl+Z**. После проверки активных заданий через команду `jobs` я увидела два процесса в состоянии *Running* и один в состоянии *Stopped*.

Чтобы возобновить выполнение третьего задания в фоне, я выполнила команду `bg 3`. После этого все три процесса работали параллельно в фоновом режиме. Затем я последовательно переносила каждое задание на передний план с помощью команды `fg` и завершала его комбинацией **Ctrl+C**.

На скриншоте ниже показана работа с заданиями, их остановка и завершение:

```

aasaenko@aasaenko:~$ su
Password:
root@aasaenko:/home/aasaenko# sleep 3600 &
[1] 3302
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[2] 3352
root@aasaenko:/home/aasaenko# sleep 7200
^Z
[3]+  Stopped                  sleep 7200
root@aasaenko:/home/aasaenko# jobs
[1]  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Stopped                  sleep 7200
root@aasaenko:/home/aasaenko# bg 3
[3]+ sleep 7200 &
root@aasaenko:/home/aasaenko# jobs
[1]  Running                  sleep 3600 &
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Running                  sleep 7200 &
root@aasaenko:/home/aasaenko# fg 1
sleep 3600
^C
root@aasaenko:/home/aasaenko# fg 2
dd if=/dev/zero of=/dev/null
^C118143335+0 records in
118143334+0 records out
60489387008 bytes (60 GB, 56 GiB) copied, 79.9449 s, 757 MB/s

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

```

Рис. 2.1: Перемещение заданий в передний план и завершение

Далее я открыла второй терминал под своей учётной записью и запустила команду `dd if=/dev/zero of=/dev/null &`. После выхода из терминала процесс продолжал выполняться.

Чтобы убедиться в этом, я использовала команду `top` в другом терминале и увидела, что процесс `dd` всё ещё активен и занимает 100% CPU.

```

top - 11:03:47 up 5 min, 4 users, load average: 0.34, 0.26, 0.11
Tasks: 262 total, 2 running, 260 sleeping, 0 stopped, 0 zombie
%Cpu(s): 12.2 us, 12.2 sy, 0.0 ni, 75.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3909.0 total, 1385.5 free, 1371.1 used, 1390.7 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2537.9 avail Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3805	aasaenko	20	0	226848	1824	1824	R	100.0	0.0	0:08.71	dd
1	root	20	0	49192	41140	10192	S	0.0	1.0	0:01.51	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
8	root	20	0	0	0	0	I	0.0	0.0	0:00.02	kworker/0:0-cgroup_destroy
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-ipv6_addrconf
12	root	20	0	0	0	0	I	0.0	0.0	0:00.05	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	0:00.08	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	0:00.14	rcu_exp_gp_kthread_worker
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	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. 2.2: Отображение процесса dd в top

Затем я снова открыла top и с помощью клавиши **k** завершила процесс dd. После этого он исчез из списка выполняющихся заданий.

```

top - 11:04:04 up 6 min, 4 users, load average: 0.60, 0.33, 0.13
Tasks: 261 total, 1 running, 260 sleeping, 0 stopped, 0 zombie
%Cpu(s): 4.6 us, 6.4 sy, 0.1 ni, 88.6 id, 0.0 wa, 0.2 hi, 0.2 si, 0.0 st
MiB Mem : 3909.0 total, 1353.9 free, 1402.5 used, 1391.0 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2506.5 avail Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3101	aasaenko	20	0	3030216	311560	96120	S	2.5	7.8	0:02.62	ptxis
2114	aasaenko	20	0	4913900	312396	122588	S	1.8	7.8	0:03.94	gnome-shell
93	root	20	0	0	0	0	I	0.5	0.0	0:00.22	kworker/u19:2-xfs-blockgc/dm-0
1160	root	20	0	552492	18128	15440	S	0.3	0.5	0:00.10	NetworkManager
1	root	20	0	49192	41140	10192	S	0.0	1.0	0:01.53	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
8	root	20	0	0	0	0	I	0.0	0.0	0:00.02	kworker/0:0-cgroup_destroy
9	root	20	0	0	0	0	I	0.0	0.0	0:00.10	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-ipv6_addrconf
12	root	20	0	0	0	0	I	0.0	0.0	0:00.05	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	0:00.08	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	0:00.14	rcu_exp_gp_kthread_worker
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	migration/0

Рис. 2.3: Завершение процесса dd через top

2.2 Управление процессами в Linux

Сначала я получила права суперпользователя с помощью команды `su -`.

Затем я запустила три процесса `dd if=/dev/zero of=/dev/null &`, которые выполняли интенсивные операции записи в `/dev/null` в фоновом режиме.

После этого я проверила список процессов командой `ps aux | grep dd`. В выводе были видны три активных процесса `dd`, запущенные в системе.

```
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[1] 4016
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[2] 4028
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[3] 4040
root@aasaenko:/home/aasaenko# ps aux | grep dd
root      2  0.0  0.0      0   0 ?        S   10:57   0:00 [kthreadd]
root     11  0.0  0.0      0   0 ?        I   10:57   0:00 [kworker/u16:0-ipv6_addrconf]
root     12  0.0  0.0      0   0 ?        I   10:57   0:00 [kworker/u16:1-ipv6_addrconf]
root     92  0.0  0.0      0   0 ?        I<  10:57   0:00 [kworker/R-ipv6_addrconf]
root    1143  0.0  0.0 512956 2896 ?        Sl   10:58   0:00 /usr/sbin/VBoxService --pidfile /var/run/v
boxadd-service.sh
aasaenko 2517  0.0  0.6 962676 25280 ?        Ssl  10:58   0:00 /usr/libexec/evolution-addressbook-factory
root    4016 99.2  0.0 226848 1808 pts/0    R   11:04   0:34 dd if=/dev/zero of=/dev/null
root    4028 99.1  0.0 226848 1756 pts/0    R   11:04   0:27 dd if=/dev/zero of=/dev/null
root    4040 98.9  0.0 226848 1672 pts/0    R   11:04   0:23 dd if=/dev/zero of=/dev/null
root    4096  0.0  0.0 227688 2080 pts/0    S+  11:05   0:00 grep --color=auto dd
root@aasaenko:/home/aasaenko#
```

Рис. 2.4: Просмотр процессов `dd` через `ps aux`

Далее я использовала команду `renice -n 5 <PID>`, чтобы изменить приоритет одного из процессов `dd`. Это позволяет управлять нагрузкой на систему, задавая процессу больший или меньший приоритет выполнения.

Чтобы посмотреть дерево процессов и их взаимосвязи, я выполнила команду `ps fax | grep -B5 dd`. Благодаря параметру `-B5` вывод содержал также строки, предшествующие найденным, что позволило увидеть родительский процесс, из которого были запущены процессы `dd`.

```

1141 ?      Sl      0:00 /usr/bin/VBoxDRMClient
1143 ?      Sl      0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
--
2447 ?      Ssl     0:00 \_ /usr/libexec/evolution-calendar-factory
2451 ?      Ssl     0:00 \_ /usr/libexec/gvfs-udisks2-volume-monitor
2454 ?      Ssl     0:00 \_ /usr/libexec/goa-identity-service
2478 ?      Ssl     0:00 \_ /usr/libexec/gvfs-mtp-volume-monitor
2493 ?      Ssl     0:00 \_ /usr/libexec/gvfs-gphoto2-volume-monitor
2517 ?      Ssl     0:00 \_ /usr/libexec/evolution-addresbook-factory
--
3101 ?      Ssl     0:03 \_ /usr/bin/ptxis --gaplication-service
3109 ?      Ssl     0:00 | \_ /usr/libexec/ptxis-agent --socket-fd=3
3175 pts/0  Ss      0:00 | \_ /usr/bin/bash
3218 pts/0  S       0:00 | \_ su
3261 pts/0  S       0:00 | \_ bash
4016 pts/0  RN      1:10 | \_ dd if=/dev/zero of=/dev/null
4028 pts/0  R       1:03 | \_ dd if=/dev/zero of=/dev/null
4040 pts/0  R       0:59 | \_ dd if=/dev/zero of=/dev/null
4177 pts/0  R+      0:00 | \_ ps fax
4178 pts/0  S+      0:00 | \_ grep --color=auto -B5 dd
root@aasaenko:/home/aasaenko#

```

Рис. 2.5: Дерево процессов dd через ps fax

После этого я нашла PID родительской оболочки, из которой запускались процессы dd, и завершила её с помощью команды `kill -9 <PID>`. В результате оболочка завершила работу, а вместе с ней автоматически остановились и все её дочерние процессы dd.

2.3 Задание 1.

Сначала я получила права суперпользователя с помощью команды `su -`.

Затем трижды запустила процесс `dd if=/dev/zero of=/dev/null &` в фоновом режиме. В результате были созданы три параллельных процесса dd.

После этого я изменила приоритет одного из процессов с помощью команды `renice -n 5 <PID>`. Его старый приоритет был равен 0, а новый стал равен 5.

Затем я повторно изменила приоритет этого же процесса на более низкий уровень, выполнив `renice -n 15 <PID>`. В выводе видно, что приоритет изменился с 5 на 15.

Чем больше положительное значение nice, тем **ниже приоритет** процесса: системе позволяет уделять ему меньше процессорного времени. В то же время уменьшение значения nice (например, до -5 или -15) повышает приоритет, что делает процесс более «агрессивным» в использовании CPU.

Для завершения всех запущенных процессов `dd` я использовала команду `killall dd`. Все три процесса были корректно остановлены.

```
aasaenko@aasaenko:~$ su
Password:
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[1] 4500
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[2] 4512
root@aasaenko:/home/aasaenko# dd if=/dev/zero of=/dev/null &
[3] 4514
root@aasaenko:/home/aasaenko# renice -n 5 4500
4500 (process ID) old priority 0, new priority 5
root@aasaenko:/home/aasaenko# renice -n 15 4500
4500 (process ID) old priority 5, new priority 15
root@aasaenko:/home/aasaenko# 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@aasaenko:/home/aasaenko#
```

Рис. 2.6: Запуск процессов `dd`, изменение приоритетов и завершение

2.4 Задание 2.

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

Затем я запустила её на переднем плане и приостановила выполнение комбинацией **Ctrl+Z**. После проверки списка заданий с помощью `jobs` было видно одно задание в состоянии *Running* и одно в состоянии *Stopped*.

```

root@aasaenko:/home/aasaenko#
root@aasaenko:/home/aasaenko# yes > /dev/null &
[1] 4691
root@aasaenko:/home/aasaenko# yes > /dev/null
^Z
[2]+  Stopped                  yes > /dev/null
root@aasaenko:/home/aasaenko# yes > /dev/null
^C
root@aasaenko:/home/aasaenko# jobs
[1]-  Running                  yes > /dev/null &
[2]+  Stopped                  yes > /dev/null
root@aasaenko:/home/aasaenko#

```

Рис. 2.7: Запуск yes и просмотр заданий

Я возобновила выполнение второго процесса в фоне командой `bg 2`. После этого оба процесса находились в состоянии *Running*. Также я запустила процесс `yes` с помощью `nohup`, что позволило ему продолжить выполнение даже после закрытия терминала.

```

root@aasaenko:/home/aasaenko#
root@aasaenko:/home/aasaenko# jobs
[1]-  Running                  yes > /dev/null &
[2]+  Stopped                  yes > /dev/null
root@aasaenko:/home/aasaenko# fg 1
yes > /dev/null
^C
root@aasaenko:/home/aasaenko# bg 2
[2]+  yes > /dev/null &
root@aasaenko:/home/aasaenko# jobs
[2]+  Running                  yes > /dev/null &
root@aasaenko:/home/aasaenko# nohup yes > /dev/null &
[3] 4885
nohup: ignoring input and redirecting stderr to stdout
root@aasaenko:/home/aasaenko# jobs
[2]-  Running                  yes > /dev/null &
[3]+  Running                  nohup yes > /dev/null &
root@aasaenko:/home/aasaenko#

```

Рис. 2.8: Перезапуск процесса yes и использование nohup

Далее я открыла утилиту `top` и убедилась, что процессы `yes` действительно продолжают работать и занимают процессорное время.

```

top - 11:12:41 up 14 min, 5 users, load average: 2.09, 1.34, 0.70
Tasks: 264 total, 3 running, 261 sleeping, 0 stopped, 0 zombie
%Cpu(s): 15.0 us, 35.0 sy, 0.0 ni, 47.5 id, 0.0 wa, 2.5 hi, 0.0 st, 0.0 st
MiB Mem : 3909.0 total, 1545.5 free, 1203.3 used, 1398.6 buff/cache
MiB Swap: 4040.0 total, 4040.0 free, 0.0 used, 2705.7 avail Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4716	root	20	0	226820	1744	1744	R	100.0	0.0	2:12.50	yes
4885	root	20	0	226820	1760	1760	R	100.0	0.0	1:47.01	yes
1	root	20	0	49192	41268	10192	S	0.0	1.0	0:01.97	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
8	root	20	0	0	0	0	I	0.0	0.0	0:00.03	kworker/0:0-cgroup_destroy
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
12	root	20	0	0	0	0	I	0.0	0.0	0:00.06	kworker/u16:1-ipv6_addrconf
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	0:00.16	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	0:00.16	rcu_exp_gp_kthread_worker
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	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. 2.9: Просмотр процессов yes через top

После этого я запустила ещё несколько экземпляров программы yes в фоновом режиме и завершила их разными способами:

- через команду `kill <PID>` для конкретного процесса,
- через `kill -1 <PID>` (сигнал SIGHUP),
- через `kill %<номер_задания>` для завершения по идентификатору задания,
- командой `killall yes` для остановки всех процессов программы.

```

root@aasaenko:/home/aasaenko#
root@aasaenko:/home/aasaenko# yes > /dev/null &
[1] 5287
root@aasaenko:/home/aasaenko# yes > /dev/null &
[2] 5289
root@aasaenko:/home/aasaenko# yes > /dev/null &
[3] 5291
root@aasaenko:/home/aasaenko# kill 5287
[1] Terminated yes > /dev/null
root@aasaenko:/home/aasaenko# fg 2
yes > /dev/null
^C
root@aasaenko:/home/aasaenko# kill -1 5291
[3]+ Hangup yes > /dev/null
root@aasaenko:/home/aasaenko# kill -1 4885
root@aasaenko:/home/aasaenko# yes > /dev/null &
[1] 5453
root@aasaenko:/home/aasaenko# yes > /dev/null &
[2] 5455
root@aasaenko:/home/aasaenko#
^[A
root@aasaenko:/home/aasaenko# yes > /dev/null &
[3] 5458
root@aasaenko:/home/aasaenko# killall yes
[1] Terminated yes > /dev/null
[2]- Terminated yes > /dev/null
root@aasaenko:/home/aasaenko#
[3]+ Terminated yes > /dev/null
root@aasaenko:/home/aasaenko#
root@aasaenko:/home/aasaenko# █

```

Рис. 2.10: Завершение процессов yes разными методами

В завершение я сравнила приоритеты процессов. Один из них был запущен с обычным приоритетом, второй — с помощью команды `nice -n 5 yes > /dev/null &`, что установило ему более низкий приоритет (меньшее потребление CPU). Затем с помощью `renice` я изменила приоритет так, чтобы оба процесса имели одинаковые значения.

```

root@aasaenko:/home/aasaenko# yes > /dev/null &
[1] 5568
root@aasaenko:/home/aasaenko# nice -n 5 yes > /dev/null &
[2] 5594
root@aasaenko:/home/aasaenko# ps -l
F S  UID      PID      PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0       5190      5155  0  80   0 - 58153 do_wai pts/1      00:00:00 su
4 S   0       5201      5190  0  80   0 - 57575 do_wai pts/1      00:00:00 bash
4 R   0       5568      5201 99  80   0 - 56705 -      pts/1      00:00:14 yes
4 R   0       5594      5201 99  85   5 - 56705 -      pts/1      00:00:05 yes
4 R   0       5606      5201  0  80   0 - 57682 -      pts/1      00:00:00 ps
root@aasaenko:/home/aasaenko# renice -n 5 5568
5568 (process ID) old priority 0, new priority 5
root@aasaenko:/home/aasaenko# ps -l
F S  UID      PID      PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0       5190      5155  0  80   0 - 58153 do_wai pts/1      00:00:00 su
4 S   0       5201      5190  0  80   0 - 57575 do_wai pts/1      00:00:00 bash
4 R   0       5568      5201 99  85   5 - 56705 -      pts/1      00:00:54 yes
4 R   0       5594      5201 99  85   5 - 56705 -      pts/1      00:00:46 yes
4 R   0       5692      5201  0  80   0 - 57682 -      pts/1      00:00:00 ps
root@aasaenko:/home/aasaenko# killall yes
[1]-  Terminated                  yes > /dev/null
[2]+  Terminated                  nice -n 5 yes > /dev/null
root@aasaenko:/home/aasaenko# █

```

Рис. 2.11: Сравнение приоритетов yes и изменение их через renice

3 Контрольные вопросы

1. Какая команда даёт обзор всех текущих заданий оболочки?

Для этого используется команда `jobs`, которая показывает список всех фоновых и приостановленных заданий текущей оболочки.

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

Сначала нужно приостановить задание комбинацией клавиш **Ctrl+Z**, а затем возобновить его выполнение в фоне командой `bg`.

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

Для этого используется комбинация **Ctrl+C**, которая посылает процессу сигнал прерывания (SIGINT).

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 использовать другую оболочку или терминал и завершить процесс командой `kill <PID>` или `killall <имя_процесса>`.

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

Для этого применяется команда `ps fax`, которая выводит дерево процессов.

6. Какая команда позволит изменить приоритет процесса с идентифика-

тором 1234 на более высокий?

Для этого используется команда `renice -n -5 -p 1234`, где отрицательное значение `nice` повышает приоритет.

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

Самый простой способ — использовать команду `killall dd`.

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

Для этого применяется команда `killall mycommand`.

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

В утилите `top` используется клавиша **k**, после чего нужно указать PID процесса.

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

Для этого используют команду `nice` с положительным значением приоритета, например:

`nice -n 10 <команда>`.

Это уменьшает приоритет процесса, освобождая ресурсы для остальных.

4 Заключение

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

Были выполнены следующие действия:

- запуск процессов на переднем и фоновом режиме;
- приостановка заданий с помощью комбинации клавиш **Ctrl+Z** и их возобновление через `bg`;
- завершение процессов с использованием **Ctrl+C**, `kill`, `kill -9`, `killall` и встроенных возможностей утилиты `top`;
- использование команд `jobs`, `ps aux` и `ps fax` для мониторинга и анализа процессов;
- управление приоритетами с помощью `nice` и `renice`;
- применение `nohup` для продолжения работы процессов после закрытия терминала.

В процессе работы я закрепила знания о взаимодействии между родительскими и дочерними процессами, а также о влиянии приоритета на использование ресурсов системы.

Полученный опыт показал, как с помощью стандартных инструментов Linux можно эффективно управлять задачами, контролировать нагрузку и завершать процессы разными способами.