

Лабораторная работа 10

1132232887

Накова Амина Михайловна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	14
	Список литературы	15

Список иллюстраций

2.1 шаг 1	7
2.2 шаг 2	7
2.3 шаг 3	8
2.4 шаг 4	9
2.5 шаг 5	9
2.6 шаг 6	10
2.7 шаг 7	11
2.8 шаг 8	11
2.9 шаг 8	12
2.10 шаг 8	12
2.11 Запуск программы уес в фоновом режиме с подавлением потока вывода. Запуск программы уес с теми же параметрами и с прио- ритетом, большим на 5. Сравнение абсолютных и относительных приоритетов, изменение приоритета.	12

Список таблиц

1 Цель работы

Целью данной работы является получение навыков управления процессами операционной системы.

2 Выполнение лабораторной работы

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. 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.

```

[nakova@localhost ~]$ su nakova
Пароль:
[nakova@localhost ~]$ sleep 3600 &
[1] 37580
[nakova@localhost ~]$ dd if=/dev/zero of=/dev/null &
[2] 37592
[nakova@localhost ~]$ sleep 7200

^Z
[3]+  Остановлен   sleep 7200
[nakova@localhost ~]$ jobs
[1]  Запущен       sleep 3600 &
[2]-  Запущен       dd if=/dev/zero of=/dev/null &
[3]+  Остановлен   sleep 7200
[nakova@localhost ~]$ bg 3
[3]+  sleep 7200 &
[1]  Завершён      sleep 3600

[nakova@localhost ~]$ ^C
[nakova@localhost ~]$ jobs
[2]-  Запущен       dd if=/dev/zero of=/dev/null &
[3]+  Запущен       sleep 7200 &

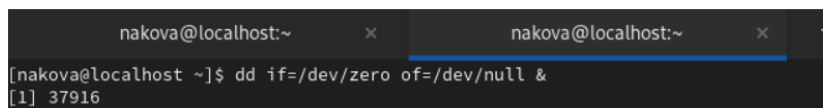
[nakova@localhost ~]$ fg 2
dd if=/dev/zero of=/dev/null
^C2654830860+0 записей получено
2654830860+0 записей отправлено
1359273400320 байт (1,4 TB, 1,2 TiB) скопирован, 3707,25 s, 367 MB/s

[nakova@localhost ~]$ jobs
[3]+  Запущен       sleep 7200 &
[nakova@localhost ~]$ fg 3
sleep 7200
^C
[nakova@localhost ~]$ jobs
[nakova@localhost ~]$

```

Рис. 2.1: шаг 1

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.



```

nakova@localhost:~  x  nakova@localhost:~  x
[nakova@localhost ~]$ dd if=/dev/zero of=/dev/null &
[1] 37916

```

Рис. 2.2: шаг 2

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 - 22:01:22 up 20:07, 2 users, load average: 0,64, 0,78, 0,78										
Tasks: 217 total, 2 running, 215 sleeping, 0 stopped, 0 zombie										
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,4 id, 0,0 wa, 0,0 hi, 0,3 si, 0,0 st										
MiB Mem : 3659,7 total, 264,1 free, 1707,8 used, 1980,9 buff/cache										
MiB Swap: 4044,0 total, 4038,2 free, 5,8 used. 1951,9 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
37916	nakova	20	0	220988	1792	1792	R	92,3	0,0	0:44.89 dd
1247	nakova	20	0	4124644	402520	136212	S	0,6	10,7	21:21.60 gnome-s+
1657	root	20	0	240744	9984	8704	S	0,3	0,3	0:37.33 sssd_kcm
37710	root	20	0	0	0	0	I	0,3	0,0	0:01.76 kworker+
37852	root	20	0	0	0	0	I	0,3	0,0	0:00.09 kworker+
37932	nakova	20	0	225884	4224	3456	R	0,3	0,1	0:00.15 top
1	root	20	0	174480	18432	10932	S	0,0	0,5	0:12.81 systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.39 kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 rcu_gp

Tasks: 217 total, 7 running, 210 sleeping, 0 stopped, 0 zombie										
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st										
MiB Mem : 3659,7 total, 263,9 free, 1708,1 used, 1980,9 buff/cache										
MiB Swap: 4044,0 total, 4038,2 free, 5,8 used. 1951,6 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
37916	nakova	20	0	220988	1792	1792	R	91,7	0,0	1:25.85 dd
1247	nakova	20	0	4124644	402520	136212	S	0,3	10,7	21:21.99 gnome-s+
37852	root	20	0	0	0	0	I	0,3	0,0	0:00.11 kworker+
37937	nakova	20	0	225884	4352	3584	R	0,3	0,1	0:00.04 top
1	root	20	0	174480	18432	10932	S	0,0	0,5	0:12.81 systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.39 kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 rcu_par+
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 slub_fl+

Рис. 2.3: шаг 3

Управление процессами: Получим полномочия администратора su - и введём следующие команды: 5 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 2682).


```

nakova@localhost:~
top - 22:02:45 up 20:08,  2 users,  load average: 0,85, 0,81, 0,79
Tasks: 217 total,  2 running, 215 sleeping,  0 stopped,  0 zombie
%Cpu(s):  3,9 us, 12,2 sy,  0,0 ni, 83,9 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem : 3659,7 total,  260,5 free, 1711,5 used, 1981,0 buff/cache
MiB Swap: 4044,0 total, 4038,2 free,  5,8 used. 1948,2 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 37916 nakova    20   0 220988   1792   1792  R   92,2   0,0   2:01.60  dd
 37968 nakova    20   0 225884   4224   3456  R    0,7   0,1   0:00.06  top
 37824 root       20   0      0      0      0  I    0,3   0,0   0:00.27  kworker+
 37852 root       20   0      0      0      0  I    0,3   0,0   0:00.12  kworker+
    1 root       20   0 174480  18432  10932  S    0,0   0,5   0:12.81  systemd
    2 root       20   0      0      0      0  S    0,0   0,0   0:00.39  kthreadd
    3 root        0 -20      0      0      0  I    0,0   0,0   0:00.00  rcu_gp
    4 root        0 -20      0      0      0  I    0,0   0,0   0:00.00  rcu_par+
    5 root        0 -20      0      0      0  T    0,0   0,0   0:00.00  slub_flo

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 1247 nakova    20   0 4120600 402512 136204  S    0,8  10,7  21:25.17  gnome-s+
 37365 nakova    20   0 836080   51092  37876  S    0,1   1,4   0:05.44  gnome-t+
 37968 nakova    20   0 225884   4224   3456  R    0,1   0,1   0:00.14  top
   947 nakova    20   0  23620  14580  10752  S    0,1   0,4   0:02.17  systemd
 36997 root       20   0      0      0      0  I    0,1   0,0   0:06.61  kworker+
 37710 root       20   0      0      0      0  I    0,1   0,0   0:01.85  kworker+
 37852 root       20   0      0      0      0  I    0,1   0,0   0:00.13  kworker+
    1 root       20   0 174480  18432  10932  S    0,0   0,5   0:12.81  systemd
    2 root       20   0      0      0      0  S    0,0   0,0   0:00.39  kthreadd

```

Рис. 2.4: шаг 4

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



Рис. 2.5: шаг 5

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 7 процесса — простой и удобный способ остановить все его дочерние процессы).

```
[2]+  Stopped                  yes > /dev/null  
[3]-  Running                  yes > /dev/null &
```

Рис. 2.6: шаг 6

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`

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 переведём процесс, который у нас выполняется в фоновом режиме, на передний план, затем остановим его (`fg 1`, после чего `Ctrl+c`). Переведём 3 процесс с подавлением потока вывода в фоновый режим (`bg 3`) и проверим состояния заданий, воспользовавшись командой `jobs`. Обратите 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

PID	USER	CPU	MEM	VSZ	RSS	TTY	STAT	COMMAND
3	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 rcu_gp
4	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 rcu_par_gp
6	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 kworker/0:0H-e+
9	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 mm_percpu_wq
10	root	20	0	0	0	0	S	0.0 0.0 0:00.00 rcu_tasks_kthre
11	root	20	0	0	0	0	S	0.0 0.0 0:00.00 rcu_tasks_rude
12	root	20	0	0	0	0	S	0.0 0.0 0:00.00 rcu_tasks_trace
13	root	20	0	0	0	0	S	0.0 0.0 0:00.07 ksoftirqd/0
14	root	20	0	0	0	0	I	0.0 0.0 0:00.12 rcu_preempt
15	root	rt	0	0	0	0	S	0.0 0.0 0:00.00 migration/0
16	root	20	0	0	0	0	S	0.0 0.0 0:00.00 cpuhp/0
18	root	20	0	0	0	0	S	0.0 0.0 0:00.00 kdevtmpfs
19	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 netns
20	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 inet_frag_wq
21	root	20	0	0	0	0	S	0.0 0.0 0:00.00 kauditd
22	root	20	0	0	0	0	S	0.0 0.0 0:00.00 khungtaskd
23	root	20	0	0	0	0	S	0.0 0.0 0:00.00 oom_reaper
24	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 writeback
25	root	20	0	0	0	0	S	0.0 0.0 0:00.03 kcompactd0
26	root	25	5	0	0	0	S	0.0 0.0 0:00.00 ksm
27	root	39	19	0	0	0	S	0.0 0.0 0:00.12 khugepaged
28	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 cryptd
29	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 kintegrityd
30	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 kblockd
31	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 blkcg_punt_bio
32	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 tpm_dev_wq
33	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 md
34	root	0	-20	0	0	0	I	0.0 0.0 0:00.00 edac-poller
35	root	-51	0	0	0	0	S	0.0 0.0 0:00.00 watchdogd
37	root	0	-20	0	0	0	I	0.0 0.0 0:00.03 kworker/0:1H-k+
38	root	20	0	0	0	0	S	0.0 0.0 0:00.00 kswapd0

Рис. 2.7: шаг 7

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. 9 Сейчас получим информацию о запущенных в операционной системе процессах с помощью утилиты top

```
# yes > /dev/null &

# yes > /dev/null &

# yes > /dev/null &

# kill -9 3098

# fg 2
```

Рис. 2.8: шаг 8

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 3098`), а для другого — его идентификатор конкретного задания (`fg 2` и `Ctrl+c`). Попробуем послать сигнал 1 (`SIGHUP`) процессу, запущенному с помощью `nohup` (`kill -1 3100`), и обычному процессу (`kill -1 2993`)

```
yes > /dev/null &
yes > /dev/null &
yes > /dev/null &
killall yes
```

Рис. 2.9: шаг 8

Запустим ещё несколько программ `yes` в фоновом режиме с подавлением потока вывода (`yes > /dev/null &`) и завершим их работу одновременно, используя команду `killall yes`

```
R  0  3109  3071 92 95 15 - 55238 - pts/0 00:01:37 yes
R  0  3113  3071  7 95 15 - 55238 - pts/0 00:00:03 yes
```

Рис. 2.10: шаг 8

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 3109`)

```
R  0  3109  3071 92 95 15 - 55238 - pts/0 00:01:37 yes
R  0  3113  3071  7 95 15 - 55238 - pts/0 00:00:03 yes
```

Рис. 2.11: 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.

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.

12 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`. 13 10. Как запустить команду с достаточно высоким приоритетом, не рискуя, что не хватит ресурсов для других процессов? Запустить команду в фоновом режиме.

3 Выводы

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

Список литературы