

Отчёт по лабораторной работе №6

Дисциплина: Основы администрирования операционных систем

Верниковская Екатерина Андреевна

Содержание

1	Цель работы	6
2	Задание	7
3	Выполнение лабораторной работы	8
3.1	Управление заданиями	8
3.2	Управление процессами	12
4	Самостоятельная работа	14
4.1	Задание 1	14
4.2	Задание 2	15
5	Контрольные вопросы + ответы	22
6	Выводы	24
7	Список литературы	25

Список иллюстраций

3.1	Режим суперпользователя	8
3.2	Запуск нужных команд и остановка последней	8
3.3	Команда jobs (1)	9
3.4	Продолжение выполнения задания 3 в фоновом режиме	9
3.5	Команда jobs (2)	9
3.6	Перемещение задания 1 на передний план и его отмена	9
3.7	Команда jobs (3)	10
3.8	Отмена задания 2 и 3	10
3.9	Команда dd if=/dev/zero of=/dev/null & (1)	10
3.10	Заккрытие второго терминала	10
3.11	Команда top	11
3.12	Вывод команды top	11
3.13	Убийство задания dd	11
3.14	Убитое задание dd	11
3.15	Команда dd if=/dev/zero of=/dev/null & (2)	12
3.16	Команда ps aux grep dd	12
3.17	Меняем приоритет 1-ого процесса на 5	12
3.18	Команда ps fax grep -B5 dd	13
3.19	Остановка родительского процесса	13
4.1	Запуск нужной команды как фоновое задание	14
4.2	Изменение приоритета 1-ого процесса	15
4.3	Завершение всех процессов dd	15
4.4	Программа yes в фоновом режиме с подавлением потока вывода .	15
4.5	Программа yes на переднем плане с подавлением потока вывода и её приостановка	16
4.6	Программа yes на переднем плане с подавлением потока вывода и её завершение	16
4.7	Всё вместе	16
4.8	Запуск программы yes на переднем плане без подавления потока вывода (1)	16
4.9	Программа yes на переднем плане без подавления потока вывода	17
4.10	Приостановка программы yes	17
4.11	Запуск программы yes на переднем плане без подавления потока вывода (2)	17
4.12	Завершение программы yes	17
4.13	Команда jobs (4)	18

4.14	Перевод процесса из фонового режима на передний план + его остановка	18
4.15	Перевод процесса с подавление потока в фоновый режим	18
4.16	Команда jobs (5)	18
4.17	Команда nohup	19
4.18	Проверка с помощью top	19
4.19	Ещё 3 команды yes в фоновом режиме с подавлением потока вывода	19
4.20	Убийства двух процессов с помощью PID и идентификатора кон- кретного задания	19
4.21	Попытки послать сигналы	20
4.22	Запуск нескольких команд yes и их одновременное завершение .	20
4.23	Запуск программы yes в фоновом режиме с подавлением потока вывода. Запуск программы yes с теми же параметрами и с прио- ритетом, большим на 5. Сравнение абсолютных и относительных приоритетов	21
4.24	Изменение приоритета	21

Список таблиц

1 Цель работы

Получить навыки управления процессами операционной системы.

2 Задание

1. Продемонстрировать навыки управления заданиями операционной системы
2. Продемонстрировать навыки управления процессами операционной системы
3. Выполнить задания для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Управление заданиями

В консоли переходим в режим работы суперпользователя, используя *su* - (рис. 3.1)

```
[eavernikovskaya@eavernikovskaya ~]$ su -  
Password:  
[root@eavernikovskaya ~]# █
```

Рис. 3.1: Режим суперпользователя

- Вводим следующие команды:
 - *sleep 3600 &*
 - *dd if=/dev/zero of=/dev/null &*
 - *sleep 7200*

Так как последнюю команду мы запустили без *&* (т.е не в фоновом режиме), то у нас есть 2 часа, прежде чем мы снова получим контроль над оболочкой. Чтобы остановить процесс, мы вводим *ctrl+z* (рис. 3.2)

```
[root@eavernikovskaya ~]# sleep 3600 &  
[1] 2955  
[root@eavernikovskaya ~]# dd if=/dev/zero of=/dev/null &  
[2] 2972  
[root@eavernikovskaya ~]# sleep 7200  
^Z  
[3]+  Stopped                  sleep 7200  
[root@eavernikovskaya ~]# █
```

Рис. 3.2: Запуск нужных команд и остановка последней

Введя *jobs* мы увидим 3 задания, которые только что запустили. Первые два имеют состояние Running, а последнее задание в настоящее время находится в состоянии Stopped (так как до этого мы его остановили) (рис. 3.3)

```
[root@eavernikovskaya ~]# jobs
[1]  Running          sleep 3600 &
[2]-  Running          dd if=/dev/zero of=/dev/null &
[3]+  Stopped          sleep 7200
[root@eavernikovskaya ~]#
```

Рис. 3.3: Команда jobs (1)

Вводим команду *bg 3*, чтобы продолжить выполнение 3-его задания в фоновом режиме (рис. 3.4)

```
[root@eavernikovskaya ~]# bg 3
[3]+  sleep 7200 &
[root@eavernikovskaya ~]#
```

Рис. 3.4: Продолжение выполнения задания 3 в фоновом режиме

С помощью команды *jobs* смотрим изменения в статусе заданий (рис. 3.5)

```
[root@eavernikovskaya ~]# jobs
[1]  Running          sleep 3600 &
[2]-  Running          dd if=/dev/zero of=/dev/null &
[3]+  Running          sleep 7200 &
[root@eavernikovskaya ~]#
```

Рис. 3.5: Команда jobs (2)

Вводим команду *fg 1* для перемещения задания 1 на передний план. А после вводим *ctrl+c*, чтобы отменить 1-ое задание (рис. 3.6)

```
[root@eavernikovskaya ~]# fg 1
sleep 3600
^C
[root@eavernikovskaya ~]#
```

Рис. 3.6: Перемещение задания 1 на передний план и его отмена

Снова вводим команду *jobs*, чтобы посмотреть изменения в статусе заданий (рис. 3.7)

```
[root@eavernikovskaya ~]# jobs
[2]-  Running                  dd if=/dev/zero of=/dev/null &
[3]+  Running                  sleep 7200 &
[root@eavernikovskaya ~]#
```

Рис. 3.7: Команда *jobs* (3)

Тоже самое делаем для отмены заданий 2 и 3. Сначала с помощью *fg* перемещаем задания на передний план, а далее отменяем их с помощью *ctrl+c* (рис. 3.8)

```
[root@eavernikovskaya ~]# fg 2
dd if=/dev/zero of=/dev/null
^C136084790+0 records in
136084790+0 records out
69675412480 bytes (70 GB, 65 GiB) copied, 203.159 s, 343 MB/s

[root@eavernikovskaya ~]# fg 3
sleep 7200
^C
[root@eavernikovskaya ~]# jobs
[root@eavernikovskaya ~]#
```

Рис. 3.8: Отмена задания 2 и 3

Открываем второй терминал и под учётной записью своего пользователя вводим в нём: *dd if=/dev/zero of=/dev/null &*. Далее закрываем второй терминал (рис. 3.9), (рис. 3.10)

```
[eavernikovskaya@eavernikovskaya ~]$ dd if=/dev/zero of=/dev/null &
[1] 3204
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 3.9: Команда *dd if=/dev/zero of=/dev/null &* (1)

```
[eavernikovskaya@eavernikovskaya ~]$ exit
```

Рис. 3.10: Закрытие второго терминала

Снова открываем второй терминал и вводим *top* (команда, которая позволяет

пользователям отслеживать процессы и использование системных ресурсов в Linux). Мы увидим что задание dd всё ещё запущено (рис. 3.11), (рис. 3.12)

```
[eavernikovskaya@eavernikovskaya ~]$ top
```

Рис. 3.11: Команда top

```
top - 15:10:48 up 23 min, 2 users, load average: 1.15, 1.09, 0.65
Tasks: 211 total, 3 running, 208 sleeping, 0 stopped, 0 zombie
%Cpu(s): 15.6 us, 12.3 sy, 0.0 ni, 70.8 id, 0.0 wa, 1.3 hi, 0.0 si, 0.0 st
MiB Mem : 1967.5 total, 174.5 free, 1036.4 used, 932.4 buff/cache
MiB Swap: 2096.0 total, 2093.5 free, 2.5 used, 931.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM    TIME+  COMMAND
 3204 eaverni+  20   0  220988    1792    1792  R   98.7   0.1   0:27.43 dd
 1885 eaverni+  20   0 4093340  372396  127128  S    4.6  18.5   0:34.39 gnome-shell
 2850 eaverni+  20   0  772500  52704  39812  S    1.7   2.6   0:05.40 gnome-terminal-
 3235 eaverni+  20   0  226028   4096   3328  R    0.3   0.2   0:00.02 top
    1 root      20   0  173144  16600  10652  S    0.0   0.8   0:01.74 systemd
    2 root      20   0      0      0      0  S    0.0   0.0   0:00.00 kthreadd
```

Рис. 3.12: Вывод команды top

Вводим *k*, чтобы убить задание dd (рис. 3.13), (рис. 3.14)

```
top - 15:11:06 up 23 min, 2 users, load average: 1.11, 1.08, 0.65
Tasks: 211 total, 2 running, 209 sleeping, 0 stopped, 0 zombie
%Cpu(s): 11.0 us, 8.5 sy, 0.0 ni, 79.7 id, 0.0 wa, 0.8 hi, 0.0 si, 0.0 st
MiB Mem : 1967.5 total, 162.5 free, 1048.4 used, 932.4 buff/cache
MiB Swap: 2096.0 total, 2093.5 free, 2.5 used, 919.1 avail Mem

  PID to signal/kill [default pid = 3204] 3204
  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM    TIME+  COMMAND
 3204 eaverni+  20   0  220988    1792    1792  R   99.3   0.1   0:46.04 dd
 1885 eaverni+  20   0 4096924  372396  127128  S    1.7  18.5   0:34.93 gnome-shell
 2850 eaverni+  20   0  772500  52704  39812  S    0.7   2.6   0:05.57 gnome-terminal-
 521  root      20   0      0      0      0  S    0.3   0.0   0:00.12 xfsaild/dm-0
    1 root      20   0  173144  16600  10652  S    0.0   0.8   0:01.74 systemd
    2 root      20   0      0      0      0  S    0.0   0.0   0:00.00 kthreadd
```

Рис. 3.13: Убийство задания dd

```
top - 15:11:28 up 23 min, 2 users, load average: 1.00, 1.06, 0.66
Tasks: 209 total, 1 running, 208 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.0 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.2 hi, 0.0 si, 0.0 st
MiB Mem : 1967.5 total, 164.7 free, 1046.2 used, 932.4 buff/cache
MiB Swap: 2096.0 total, 2093.5 free, 2.5 used, 921.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM    TIME+  COMMAND
 1885 eaverni+  20   0 4096940  372396  127128  S    0.7  18.5   0:35.42 gnome-shell
 2274 eaverni+  20   0  167596  46932  39116  S    0.3   2.3   0:00.79 Xwayland
 2850 eaverni+  20   0  772500  52704  39812  S    0.3   2.6   0:05.69 gnome-terminal-
    1 root      20   0  173144  16600  10652  S    0.0   0.8   0:01.74 systemd
    2 root      20   0      0      0      0  S    0.0   0.0   0:00.00 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_gp
    5 root      0 -20      0      0      0  I    0.0   0.0   0:00.00 slub_flushwq
    6 root      0 -20      0      0      0  I    0.0   0.0   0:00.00 netns
    8 root      0 -20      0      0      0  I    0.0   0.0   0:00.00 kworker/0:0H-events_highpri
   10 root      0 -20      0      0      0  I    0.0   0.0   0:00.00 mm_percpu_wq
   11 root      20   0      0      0      0  I    0.0   0.0   0:00.64 kworker/u4:1-events_unbound
```

Рис. 3.14: Убитое задание dd

3.2 Управление процессами

Снова получаем права пользователя root и вводим три раза команду `dd if=/dev/zero of=/dev/null &` (рис. 3.15)

```
[root@eavernikovskaya ~]# dd if=/dev/zero of=/dev/null &
[1] 3245
[root@eavernikovskaya ~]# dd if=/dev/zero of=/dev/null &
[2] 3246
[root@eavernikovskaya ~]# dd if=/dev/zero of=/dev/null &
[3] 3247
[root@eavernikovskaya ~]#
```

Рис. 3.15: Команда `dd if=/dev/zero of=/dev/null &` (2)

Далее вводим `ps aux | grep dd`. Эта команда показывает все строки, в которых есть буквы dd. Запущенные процессы dd идут последними (рис. 3.16)

```
[root@eavernikovskaya ~]# ps aux | grep dd
root      2  0.0  0.0   0   0 ?        Ss   14:47   0:00 [kthreadd]
root     65  0.0  0.0   0   0 ?        I<   14:47   0:00 [ipv6_addrconf]
root    1035  0.0  0.1 508552 3712 ?        Sl   14:47   0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-se
rvise.sh
eaverni+ 2043  0.0  1.5 953800 31984 ?        Ssl  14:48   0:00 /usr/libexec/evolution-addressbook-factory
eaverni+ 2475  0.0  0.0 232496 1540 ?        S    14:48   0:00 /usr/bin/VBoxClient --dragandddrop
eaverni+ 2477  0.1  0.1 431308 2948 ?        Sl   14:48   0:02 /usr/bin/VBoxClient --dragandddrop
root     3245 64.3  0.0 220988 1792 pts/1    R    15:12   0:28 dd if=/dev/zero of=/dev/null
root     3246 65.6  0.0 220988 1792 pts/1    R    15:12   0:28 dd if=/dev/zero of=/dev/null
root     3247 63.4  0.0 220988 1792 pts/1    R    15:12   0:26 dd if=/dev/zero of=/dev/null
root     3250  0.0  0.1 221796 2304 pts/1    S+   15:13   0:00 grep --color=auto dd
```

Рис. 3.16: Команда `ps aux | grep dd`

Далее используем (идентификатор процесса в системе) первого процесса dd, чтобы изменить приоритет. 1-ого процесса равен 3245. Для изменения приоритета используется команда `renice -n 5` (рис. 3.17)

```
[root@eavernikovskaya ~]# renice -n 5 3245
3245 (process ID) old priority 0, new priority 5
[root@eavernikovskaya ~]#
```

Рис. 3.17: Меняем приоритет 1-ого процесса на 5

Далее вводим `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 можем увидеть оболочку, из которой были запущены все процессы dd, и её PID (рис. 3.18)

```

[root@eavernikovskaya ~]# ps fax | grep -B5 dd
PID TTY STAT TIME COMMAND
2 ? S 0:00 [kthreadd]

55 ? I< 0:00 \_ [kthrotld]
60 ? I< 0:00 \_ [acpi_thermal_pm]
61 ? I< 0:00 \_ [kmpath_rdadcd]
62 ? I< 0:00 \_ [kaluad]
63 ? I< 0:00 \_ [mld]
65 ? I< 0:00 \_ [ipv6_adddrconf]

764 ? S 0:00 /usr/sbin/chronyd -F 2
767 ? Sns 0:00 /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --initfile=/lib/
alsa/init/00main rdaemon
775 ? Ssl 0:00 /usr/sbin/ModemManager
777 ? Ssl 0:00 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
1032 ? Sl 0:00 /usr/bin/VBoxDRMClient
1035 ? Sl 0:00 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh

1986 ? Ssl 0:00 \_ /usr/libexec/goa-daemon
1999 ? Ssl 0:00 \_ /usr/libexec/evolution-calendar-factory
2009 ? Ssl 0:00 \_ /usr/libexec/goa-identity-service
2031 ? Ssl 0:00 \_ /usr/libexec/gvfs-udisks2-volume-monitor
2037 ? Ssl 0:00 \_ /usr/libexec/dconf-service
2043 ? Ssl 0:00 \_ /usr/libexec/evolution-addressbook-factory

2328 ? Ssl 0:00 \_ /usr/libexec/gsd-xsettings
2377 ? Sl 0:00 \_ /usr/libexec/ibus-x11 --kill-daemon
2381 ? Ssl 0:00 \_ /usr/libexec/ibus-portal
2467 ? S 0:00 \_ /usr/bin/VBoxClient --seamless
2469 ? Sl 0:01 | \_ /usr/bin/VBoxClient --seamless
2475 ? S 0:00 \_ /usr/bin/VBoxClient --draganddrop
2477 ? Sl 0:02 | \_ /usr/bin/VBoxClient --draganddrop

2893 pts/0 S+ 0:00 | \_ tmux
2895 ? Ss 0:00 \_ tmux
2896 pts/1 Ss 0:00 \_ -bash
2919 pts/1 S 0:00 \_ su -
2930 pts/1 S 0:00 \_ -bash
3245 pts/1 RN 2:13 \_ dd if=/dev/zero of=/dev/null
3246 pts/1 R 3:00 \_ dd if=/dev/zero of=/dev/null
3247 pts/1 R 2:59 \_ dd if=/dev/zero of=/dev/null
3254 pts/1 R+ 0:00 \_ ps fax
3255 pts/1 S+ 0:00 \_ grep --color=auto -B5 dd

[root@eavernikovskaya ~]#

```

Рис. 3.18: Команда `ps fax | grep -B5 dd`

Находим PID корневой оболочки, из которой были запущены процессы `dd` (это 2919), после вводим `kill -9`, заменив в на значение PID оболочки (т.е на 2919). После ввода этой команды мы увидим, что наша корневая оболочка закрылась, а вместе с ней и все процессы `dd`. Остановка родительского процесса — простой и удобный способ остановить все его дочерние процессы (рис. 3.19)

```

[root@eavernikovskaya ~]# kill -9 2919
[root@eavernikovskaya ~]# Killed
[eavernikovskaya@eavernikovskaya ~]$

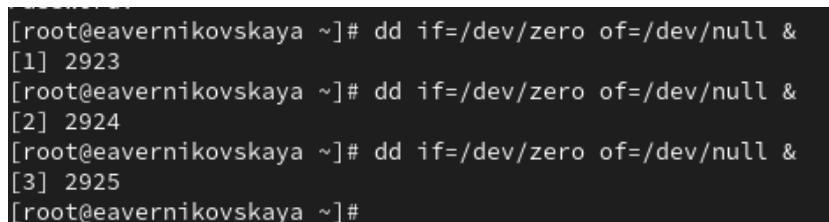
```

Рис. 3.19: Остановка родительского процесса

4 Самостоятельная работа

4.1 Задание 1

Запускаем команду `dd if=/dev/zero of=/dev/null` трижды как фоновое задание (рис. 4.1)



```
[root@eavernikovskaya ~]# dd if=/dev/zero of=/dev/null &
[1] 2923
[root@eavernikovskaya ~]# dd if=/dev/zero of=/dev/null &
[2] 2924
[root@eavernikovskaya ~]# dd if=/dev/zero of=/dev/null &
[3] 2925
[root@eavernikovskaya ~]#
```

Рис. 4.1: Запуск нужной команды как фоновое задание

Увеличиваем приоритет одной из этих команд (например 1-ой), используя значение приоритета `-5`. После изменяем значение приоритета того же процесса ещё раз, но используя на этот раз значение `-15`.

Разница между `-5` и `-15`:

- Приоритет `-5`: Увеличивает приоритет команды, делая ее более “важной” для системы. `-5` запустит команду `dd` с более высоким приоритетом, чем обычный процесс, но не будет сильно влиять на другие приложения
- Приоритет `-15`: Значительно повышает приоритет, делая команду еще более “важной”. `-15` запустит `dd` с очень высоким приоритетом. Это может привести к замедлению других процессов, так как система будет отдавать большую часть ресурсов `dd`

Чем ниже значение приоритета (более отрицательное), тем выше приоритет процесса (рис. 4.2)

```
[root@eavernikovskaya ~]# renice -n -5 2923
2923 (process ID) old priority 0, new priority -5
[root@eavernikovskaya ~]# renice -n -15 2923
2923 (process ID) old priority -5, new priority -15
[root@eavernikovskaya ~]#
```

Рис. 4.2: Изменение приоритета 1-ого процесса

Завершаем все процессы `dd`, которые мы запустили командой `killall dd` (рис. 4.3)

```
[root@eavernikovskaya ~]# 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@eavernikovskaya ~]# jobs
[root@eavernikovskaya ~]#
```

Рис. 4.3: Завершение всех процессов `dd`

4.2 Задание 2

Запускаем программу `yes` в фоновом режиме с подавлением потока вывода с помощью команды `yes > /dev/null &` (`/dev/null` отвечает за подавление потока вывода) (рис. 4.4)

```
[root@eavernikovskaya ~]# yes > /dev/null &
[1] 3498
[root@eavernikovskaya ~]# █
```

Рис. 4.4: Программа `yes` в фоновом режиме с подавлением потока вывода

Далее запускаем программу `yes` на переднем плане с подавлением потока вывода. После приостанавливаем выполнение программы с помощью `ctrl+z` (рис. 4.5)

```
[root@eavernikovskaya ~]# yes > /dev/null
^Z
[2]+  Stopped                  yes > /dev/null
[root@eavernikovskaya ~]# █
```

Рис. 4.5: Программа `yes` на переднем плане с подавлением потока вывода и её приостановка

Заново запускаем программу `yes` с теми же параметрами командой `fg 2`, затем завершаем её выполнение введя `ctrl+c` (рис. 4.6), (рис. 4.7)

```
[root@eavernikovskaya ~]# fg 2
yes > /dev/null
^C
```

Рис. 4.6: Программа `yes` на переднем плане с подавлением потока вывода и её завершение

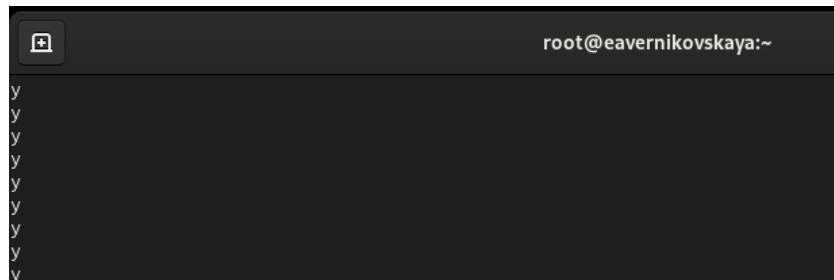
```
[root@eavernikovskaya ~]# yes > /dev/null &
[1] 3498
[root@eavernikovskaya ~]# yes > /dev/null
^Z
[2]+  Stopped                  yes > /dev/null
[root@eavernikovskaya ~]# fg 2
yes > /dev/null
^C
[root@eavernikovskaya ~]# jobs
[1]+  Running                  yes > /dev/null &
[root@eavernikovskaya ~]# █
```

Рис. 4.7: Всё вместе

Запускаем программу `yes` на переднем плане без подавления потока вывода с помощью команды просто `yes`. После приостанавливаем выполнение программы с помощью `ctrl+z` (рис. 4.8), (рис. 4.9), (рис. 4.10)

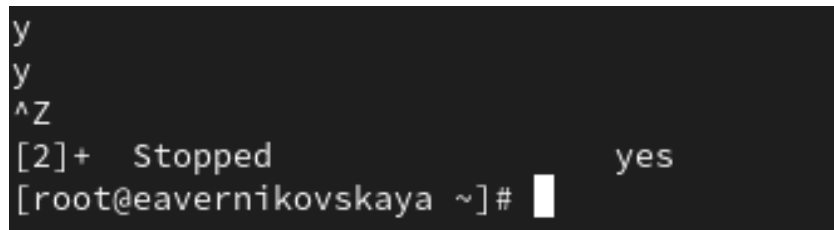
```
[root@eavernikovskaya ~]# yes
```

Рис. 4.8: Запуск программы `yes` на переднем плане без подавления потока вывода (1)



```
root@eavernikovskaya:~  
y  
y  
y  
y  
y  
y  
y  
y  
y  
y  
y
```

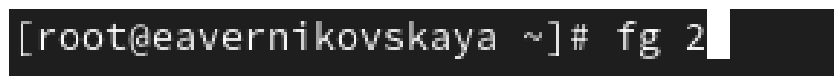
Рис. 4.9: Программа `yes` на переднем плане без подавления потока вывода



```
y  
y  
^Z  
[2]+  Stopped      yes  
[root@eavernikovskaya ~]#
```

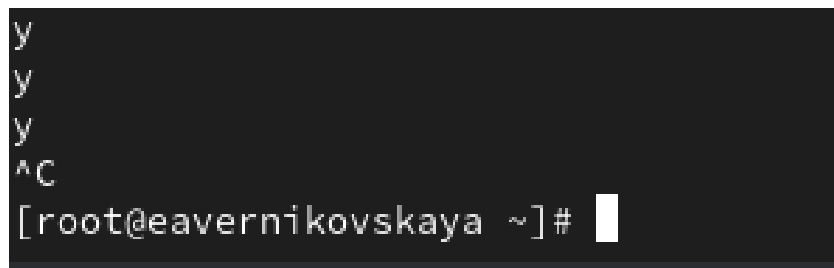
Рис. 4.10: Приостановка программы `yes`

Заново запускаем программу `yes` с теми же параметрами командой `fg 2`, затем завершаем её выполнение введя `ctrl+c` (рис. 4.11), (рис. 4.12)



```
[root@eavernikovskaya ~]# fg 2
```

Рис. 4.11: Запуск программы `yes` на переднем плане без подавления потока вывода (2)



```
y  
y  
y  
^C  
[root@eavernikovskaya ~]#
```

Рис. 4.12: Завершение программы `yes`

Проверяем состояния заданий, воспользовавшись командой `jobs` (рис. 4.13)

```
[root@eavernikovskaya ~]# jobs
[1]+  Running                  yes > /dev/null &
[root@eavernikovskaya ~]#
```

Рис. 4.13: Команда jobs (4)

Переводим процесс, который у нас выполняется в фоновом режиме, на передний план, и затем останавливаем его (комндой *fg1*, а затем *ctrl+z*) (рис. 4.14)

```
[root@eavernikovskaya ~]# fg 1
yes > /dev/null
^Z
[1]+  Stopped                  yes > /dev/null
[root@eavernikovskaya ~]#
```

Рис. 4.14: Перевод процесса из фонового режима на передний план + его остановка

Переводим любой наш процесс (он у нас один) с подавлением потока вывода в фоновый режим (командой *bg1*) (рис. 4.15)

```
[root@eavernikovskaya ~]# bg 1
[1]+ yes > /dev/null &
[root@eavernikovskaya ~]#
```

Рис. 4.15: Перевод процесса с подавление потока в фоновый режим

Снова проверяем состояния заданий, воспользовавшись командой *jobs*. Процесс стал выполняющимся (Running) в фоновом режиме (рис. 4.16)

```
[root@eavernikovskaya ~]# jobs
[1]+  Running                  yes > /dev/null &
[root@eavernikovskaya ~]#
```

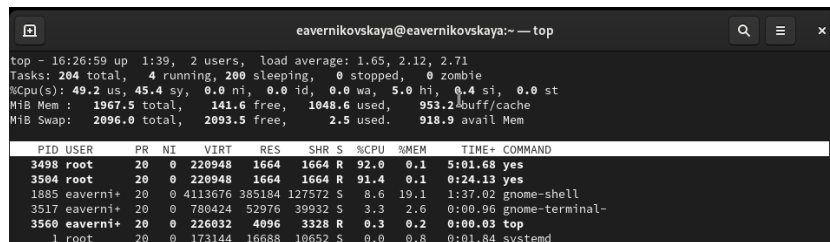
Рис. 4.16: Команда jobs (5)

Далее запускаем процесс в фоновом режиме таким образом, чтобы он продолжил свою работу даже после отключения от терминала. Это можно сделать с помощью команду *nohup*. После закрываем терминал (рис. 4.17)

```
[root@eavernikovskaya ~]# nohup yes > /dev/null &
[2] 3504
[root@eavernikovskaya ~]# nohup: ignoring input and redirecting stderr to stdout
```

Рис. 4.17: Команда nohup

Заново запускаем консоль и с помощью команды top убеждаемся, что процесс продолжил свою работу (рис. 4.18)



```
eavernikovskaya@eavernikovskaya:~$ top
top - 16:26:59 up 1:39, 2 users, load average: 1.65, 2.12, 2.71
Tasks: 204 total, 4 running, 200 sleeping, 0 stopped, 0 zombie
%Cpu(s): 49.2 us, 45.4 sy, 0.0 ni, 0.0 id, 0.0 wa, 5.0 hi, 0.4 si, 0.0 st
MiB Mem : 1967.5 total, 141.6 free, 1048.6 used, 953.2 buff/cache
MiB Swap: 2096.0 total, 2093.5 free, 2.5 used, 918.9 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
 3498 root        20   0  220948  1664  1664 R   92.0   0.1   5:01.68 yes
 3504 root        20   0  220948  1664  1664 R   91.4   0.1   0:24.13 yes
 1885 eaverni+    20   0  4113676 385184 127572 S    8.6  19.1  1:37.02 gnome-shell
 3517 eaverni+    20   0  780424  52976  39932 S    3.3   2.6   0:00.96 gnome-terminal-
 3560 eaverni+    20   0  226032  4096   3328 R    0.3   0.2   0:00.03 top
    1 root        20   0  173144  16688 10652 S    0.0   0.8   0:01.84 systemd
```

Рис. 4.18: Проверка с помощью top

После запускаем ещё три программы yes в фоновом режиме с подавлением потока вывода (рис. 4.19)

```
[root@eavernikovskaya ~]# yes > /dev/null &
[1] 3609
[root@eavernikovskaya ~]# yes > /dev/null &
[2] 3610
[root@eavernikovskaya ~]# yes > /dev/null &
[3] 3611
[root@eavernikovskaya ~]#
```

Рис. 4.19: Ещё 3 команды yes в фоновом режиме с подавлением потока вывода

Убиваем два процесса: для одного используем его PID (*kill -9 3609*), а для другого — его идентификатор конкретного задания (*fg2 + ctrl+c*) (рис. 4.20)

```
[root@eavernikovskaya ~]# kill -9 3609
[root@eavernikovskaya ~]# fg 2
yes > /dev/null
^C
[1]  Killed                  yes > /dev/null
```

Рис. 4.20: Убийства двух процессов с помощью PID и идентификатора конкретного задания

Далее пробуем послать сигнал 1 (SIGHUP) процессу, запущенному с помощью `nohup`, и обычному процессу. В первой случае это команда `kill -1 3504`, т.к. PID процесса, запущенного с помощью `nohup` это 3504. Во втором это команда `kill -1 3611` (PID обычного процесса 3611) (рис. 4.21)

```
[root@eavernikovskaya ~]# kill -1 3504
[root@eavernikovskaya ~]# kill -1 3611
[root@eavernikovskaya ~]# jobs
[3]+  Hangup                  yes > /dev/null
[root@eavernikovskaya ~]#
```

Рис. 4.21: Попытки послать сигналы

Запускаем ещё несколько программ `yes` в фоновом режиме с подавлением потока вывода. После завершаем их работу одновременно, используя команду `killall yes` (рис. 4.22)

```
[root@eavernikovskaya ~]# yes > /dev/null &
[1] 3619
[root@eavernikovskaya ~]# yes > /dev/null &
[2] 3620
[root@eavernikovskaya ~]# yes > /dev/null &
[3] 3621
[root@eavernikovskaya ~]# killall dd
dd: no process found
[root@eavernikovskaya ~]# killall yes
[1] Terminated                  yes > /dev/null
[2]- Terminated                  yes > /dev/null
[3]+ Terminated                  yes > /dev/null
[root@eavernikovskaya ~]#
```

Рис. 4.22: Запуск нескольких команд `yes` и их одновременное завершение

Запускаем программу `yes` в фоновом режиме с подавлением потока вывода. Используя утилиту `nice`, запускаем программу `yes` с теми же параметрами и с приоритетом, большим на 5

Абсолютный приоритет определяется ядром операционной системы и показывает, с каким приоритетом выполняет процесс. Относительный приоритет является тем значением, которое может быть установлено пользователем для управления тем, как процесс будет приоритизироваться системой

Процесс 1 (PID 3662):

- Абсолютный приоритет: 80
- Относительный приоритет: 0

Процесс 1 имеет более высокий абсолютный приоритет (80), чем процесс 2 (85)

Процесс 2 (PID 3666):

- Абсолютный приоритет: 85
- Относительный приоритет: 5

Процесс 2 имеет более низкий приоритет, потому что он был понижен с помощью команды `nice` (рис. 4.23)

```
[root@eavernikovskaya ~]# yes > /dev/null &
[1] 3662
[root@eavernikovskaya ~]# nice -n 5 yes > /dev/null &
[2] 3666
[root@eavernikovskaya ~]# ps -l | grep yes
0 R      0      3662    3582  96   80      0 - 55237 -      pts/1    00:00:53  yes
0 R      0      3666    3582  91   85      5 - 55237 -      pts/1    00:00:34  yes
[root@eavernikovskaya ~]#
```

Рис. 4.23: 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

Используя утилиту `renice`, меняем приоритет у одного из потоков `yes` таким образом, чтобы у обоих потоков приоритеты были равны. В нашем случае для этого понижаем приоритет процесса 1 (PID 3662) на 5 с помощью команды `renice -n 5 3662` (рис. 4.24)

```
[root@eavernikovskaya ~]# renice -n 5 3662
3662 (process ID) old priority 0, new priority 5
[root@eavernikovskaya ~]# ps -l | grep yes
0 R      0      3662    3582  94   85      5 - 55237 -      pts/1    00:01:39  yes
0 R      0      3666    3582  90   85      5 - 55237 -      pts/1    00:01:18  yes
[root@eavernikovskaya ~]#
```

Рис. 4.24: Изменение приоритета

Теперь у обоих потоков приоритеты равны. Всё хорошо!

5 Контрольные вопросы + ответы

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

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 1234`

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 `myscommand`?

Сначала узнаем PID процесса `myscommand` `-ps aux | grep myscommand`. Далее останавливаем с помощью команды `kill -9`.

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

`k`

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

Запустить в фоновом режиме

6 Выводы

В ходе выполнения лабораторной работы мы получили навыки управления процессами операционной системы

7 Список литературы

1. Лабораторная работа №6 [Электронный ресурс] URL: