Липецкий государственный технический университет

Кафедра прикладной математики

Отчет по лабораторной работе № 3
«Процессы в операционной системе Linux»
по курсу «Операционная система Linux»

Студент _____ Заев В.В.
подпись, дата фамилия, инициалы

Группа

Руководитель
Доцент, к. пед. наук _____ Кургасов В.В.
ученая степень, ученое звание подпись, дата фамилия, инициалы

Липецк 2021 г.

Содержание

Цель работы **3**

Задание кафедры **4**

Цель работы **4**

Часть 1 4

Часть 2 4

Ход работы **6**

Часть 1. Вариант 8. 6

Часть 2. 8

Выводы **14**

Контрольные вопросы **15**

Цель работы

Ознакомиться на практике с понятием процесса в операционной системе. Приобрести опыт и навыки управления процессами в операционной системе Linux.

Задание кафедры

Часть 1. Вариант 3.

1. Сгенерировать следующую информацию о m (m>2) процессах системы, имеющих значение идентификатора больше заданного n: флаг — сведения о процессе, статус, PID, PPID, приоритет, использованное время и имя программы.

2. Послать сигнал SIGINT всем процессам, запущенным командой vi. Сообщить, успешно ли был послан сигнал

3. Завершить выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершить с помощью сигнала SIGKILL, задав его имя, второй — с помощью сигнала SIGINT, задав его номер.

Часть 2.

1. Запустить программу виртуализации Oracle VM VirtualBox

2. Запустить виртуальную машину Uduntu

3. Открыть окно интерпретатора команд

4. Вывести общую информацию о системе

   (a) Вывести информацию о текущем интерпретаторе команд

   (b) Вывести информацию о текущем пользователе

   (c) Вывести информацию о текущем каталоге.

   (d) Вывести информацию об оперативной памяти и области подкачки.

   (e) Вывести информацию о дисковой памяти.

5. Выполнить команды получения информации о процессах

   (a) Получить идентификатор текущего процесса(PID)

   (b) Получить идентификатор родительского процесса(PPID).

(c) Получить идентификатор процесса инициализации системы.

(d) Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

(e) Отобразить все процессы

6. Выполнить команды управления процессами.

(a) Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе.

(b) Определить текущее значение nice по умолчанию.

(c) ПЗапустить интерпретатор bash с понижением приоритета nice –n 10 bash

(d) Определить PID запущенного интерпретатора.

(e) Установить приоритет запущенного интерпретатора равным 5

(f) Получить информацию о процессах bash

Ход работы

Часть 1. Вариант 3.

1. Сгенерировать следующую информацию о m (m>2) процессах системы, имеющих значение идентификатора больше заданного n: флаг — сведения о процессе, статус, PID, PPID, приоритет, использованное время и имя программы.

```
ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk '$2 > 990'
```

```
user@test:~$ ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk '$2 > 990'
R        997     985   0 00:00:04 sh loop
R        998     985   0 00:00:02 sh loop
R        999     985   0 00:00:01 sh loop
R+      1000     985   0 00:00:00 ps -A -o stat=,pid=,ppid=,nice=,time=,cmd=
S+      1001     985   0 00:00:00 awk $2 > 990
```

Рисунок 1

2. Послать сигнал SIGINT всем процессам, запущенным командой vi. Сообщить, успешно ли был послан сигнал.

```
kill -9 $(pgrep vi)
ps -f
```

```
user@test:~$ kill -9 $(pgrep vi)
user@test:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
user        1722    1629  0 19:21 tty1     00:00:00 -bash
user        1779    1722  0 19:25 tty1     00:00:00 ps -f
[1]+  Killed                  vi
user@test:~$
```

Рисунок 2

3. Завершить выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершить с помощью сигнала SIGKILL, задав его имя, второй — с помощью сигнала SIGINT, задав его номер.

```
kill -9 1124
killall -2 'sh loop'
```

```
user@test:~$ ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk '$2 > 990'
I       1054       2   0 00:00:00 [kworker/u2:1-events_unbound]
R       1124     985   0 00:00:27 sh loop
R+      1126     985   0 00:00:00 ps -A -o stat=,pid=,ppid=,nice=,time=,cmd=
S+      1127     985   0 00:00:00 awk $2 > 990
user@test:~$ kill -9 1124
user@test:~$ ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk '$2 > 990'
I       1054       2   0 00:00:00 [kworker/u2:1-events_power_efficient]
R+      1128     985   0 00:00:00 ps -A -o stat=,pid=,ppid=,nice=,time=,cmd=
S+      1129     985   0 00:00:00 awk $2 > 990
[1]+  Killed                  sh loop
```

Рисунок 3

```
user@test:~$ killall -2 'sh loop'
sh loop: no process found
[1]   Interrupt               sh loop
[2]-  Interrupt               sh loop
[3]+  Interrupt               sh loop
user@test:~$ ps -A -o stat=,pid=,ppid=,nice=,time=,cmd= | awk '$2 > 990'
I       1054       2   0 00:00:00 [kworker/u2:1-events_unbound]
R+      1119     985   0 00:00:00 ps -A -o stat=,pid=,ppid=,nice=,time=,cmd=
S+      1120     985   0 00:00:00 awk $2 > 990
```

Рисунок 4

Часть 2.

1. Запустить программу виртуализации Oracle VM VirtualBox
2. Запустить виртуальную машину Uduntu
3. Открыть окно интерпретатора команд
4. Вывести общую информацию о системе
   (a) Вывести информацию о текущем интерпретаторе команд

   ```
   echo $SHELL
   ```

   

   Рисунок 5

   (b) Вывести информацию о текущем пользователе

   ```
   whoami
   ```

   

   Рисунок 6

   (c) Вывести информацию о текущем каталоге.

   ```
   pwd
   ```

   

   Рисунок 7

(d) Вывести информацию об оперативной памяти и области подкачки.

```
free
```

```
user@test:~$ free
              total        used        free      shared  buff/cache   available
Mem:        4425736      148964     3896852        1020      379920     4046424
Swap:       4194300           0     4194300
```

Рисунок 8

(e) Вывести информацию о дисковой памяти.

```
df
```

```
user@test:~$ df
Filesystem                         1K-blocks    Used Available Use% Mounted on
udev                                 2167888       0   2167888   0% /dev
tmpfs                                 442576    1020    441556   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv   20511312 6491284  12955068  34% /
tmpfs                                2212868       0   2212868   0% /dev/shm
tmpfs                                   5120       0      5120   0% /run/lock
tmpfs                                2212868       0   2212868   0% /sys/fs/cgroup
/dev/loop0                             56832   56832         0 100% /snap/core18/2128
/dev/loop1                             33152   33152         0 100% /snap/snapd/12704
/dev/loop2                             72064   72064         0 100% /snap/lxd/21029
/dev/sda2                             999320  110432    820076  12% /boot
tmpfs                                 442572       0    442572   0% /run/user/1000
```

Рисунок 9

5. Выполнить команды получения информации о процессах

   (a) Получить идентификатор текущего процесса(PID)

   ```
   echo $$
   ```

   

   Рисунок 10

   (b) Получить идентификатор родительского процесса(PPID).

   ```
   echo $PPID
   ```

   

   Рисунок 11

   (c) Получить идентификатор процесса инициализации системы.

   ```
   pidof init
   ```

   

   Рисунок 12

(d) Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

```
ps
```

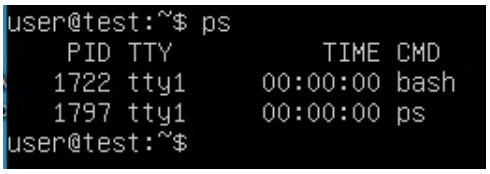

Рисунок 13

(e) Отобразить все процессы

```
ps -e
```

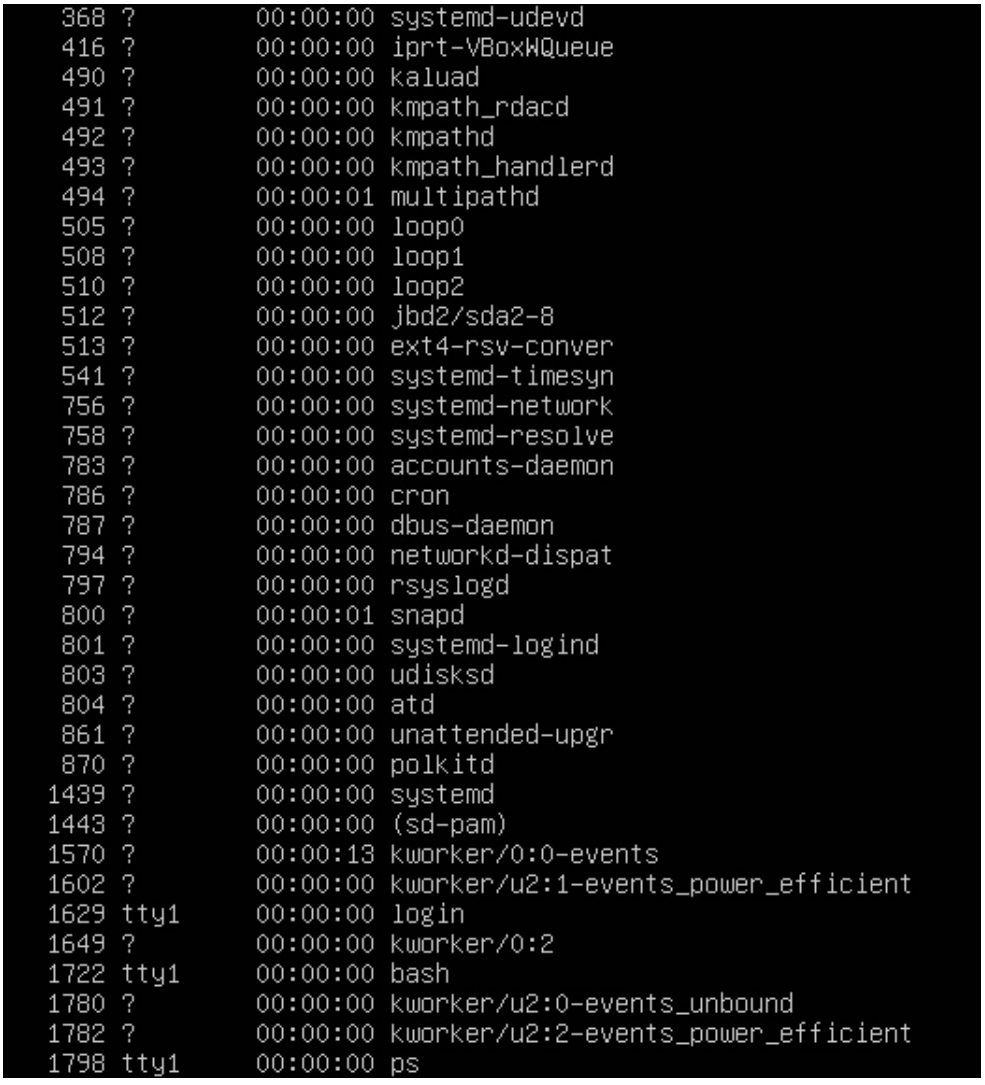

Рисунок 14

6. Выполнить команды управления процессами.

   (a) Определить текущее значение nice по умолчанию.

```
nice
```

Рисунок 15

   (b) Запустить интерпретатор bash с понижением приоритета nice –n 10 bash

```
nice -n 10 bash
```

Рисунок 16

   (c) Определить PID запущенного интерпретатора.

```
echo $$
```

Рисунок 17

(d) Установить приоритет запущенного интерпретатора равным 5

```
renice -n 5 -p <PID процесса>
```

```
root@test:/home/user# renice -n 5 -p 1833
1833 (process ID) old priority 19, new priority 5
```

Рисунок 18

(e) Получить информацию о процессах bash

```
ps lax | grep bash
```

```
root@test:/home/user# ps lax | grep bash
4  1000    1722    1629  20   0   8264  5340 do_wai S    tty1       0:00 -bash
0  1000    1801    1722  30  10   8272  5152 do_wai SN   tty1       0:00 bash
4     0    1824    1801  30  10   9260  4700 poll_s SN   tty1       0:00 sudo nice -10 bash
4     0    1825    1824  39  19   7236  4076 do_wai SN   tty1       0:00 bash
0     0    1833    1825  25   5   7236  4144 do_wai SN   tty1       0:00 bash
0     0    1842    1833  25   5   6300   672 -      RN+  tty1       0:00 grep --color=auto bash
```

Рисунок 19

Вывод

В ходе данной лабораторной работы были приобретены навыки использования виртуальной машины, а также освоены некоторые моменты работы в терминале ОС Linux, в особенности порядок работы с текстовым редактором Vi, создание сценариев процессов, работа с командами и сигналами для управления процессами: запуском, остановкой, переводом на передний план, удалением процесса и др.

Контрольные вопросы

1. Перечислите состояния задачи в ОС Ubuntu

   - running (выполнение) – после выделения ей процессора.
   - sleeping (спячка) – при блокировке экрана
   - stopped (остановлена) – выполнение задачи прекращено, но из системы не удалена.
   - dead (смерть) – может быть удалена из системы
   - active (активный) – используются при планировании выполнения процесса.
   - expired (неактивный) – используются при планировании выполнения процесса

2. Как создаются задачи в ОС Ubuntu?
   Функция clone позволяет создовать задачи.

3. Назовите классы потоков в ОС Ubuntu

   - Потоки реального времени, обслуживаемые по алгоритму FIFO.
   - Потоки реального времени, обслуживаемые в порядке циклической очереди.
   - Потоки разделения времени.

4. Как используется приоритет планирования при запуске задачи.
   У каждого потока есть приоритет планирования. Значение по умолчанию равно 20, но оно может быть изменено при помощи системного вызова nice(value), вычитающего значение value из 20. Поскольку value должно находиться в диапазоне от -20 до +19, приоритеты всегда попадают в промежуток от 1 до 40.

5. Как можно изменить приоритет планирования для выполняющейся задачи?
   Используя команду nice.