Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа № 3

по OS Linux

Процессы ОС Linux

Студент Комаричев А. В.

Группа АИ-19

Руководитель Кургасов В. В.

Липецк 2021г.

Оглавление

Цель работы

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

Ход выполнения работы

1) Отобразить информацию о процессах, начиная с указанного идентификатора, с выделением цветом текущего процесса и его предков.

Сценарий loop:

Loop: while true; do true; done

Выполним sh loop& (запускаем сценарий в фоне).

ps –efH (показать все процессы в системе, полная форма вывода, вывод иерархии в форме дерева).

```
alex@alexserver:~$ ps -efH | grep 1354
alex        1354     662  0 18:19 tty1     00:00:00     -bash
alex        1556    1354 98 18:41 tty1     00:08:00       sh loop
alex        1603    1354  0 18:49 tty1     00:00:00       ps -efH
alex        1604    1354  0 18:49 tty1     00:00:00       grep --color=auto 1354
```

Рисунок 1 – Информация о процессе 1354

```
1354 alex      20   0    8648   5600   3584 S   0,0   0,1   0:00.21       `- bash
1741 alex      20   0    2608    544    476 R  99,3   0,0   1:21.46          `- sh
1742 alex      20   0    9140   3880   3360 R   0,0   0,1   0:00.12          `- top
```

Рисунок 2 – Информация о процессе в top

2) Завершить выполнение процесса, владельцем которого является текущий пользователь, с помощью сигнала SIGINT двумя способами: задав имя сигнала и используя комбинацию клавиш.

kill int 1640 (посылает сигнал Interrupt процессу 1640)

```
alex@alexserver:~$ ps -fH
UID          PID    PPID  C STIME TTY          TIME CMD
alex        1354     662  0 18:19 tty1     00:00:00 -bash
alex        1640    1354 99 19:01 tty1     00:00:34   sh loop
alex        1644    1354  0 19:01 tty1     00:00:00   ps -fH
alex@alexserver:~$ kill int 1640
```

```
[1]+  Terminated              sh loop
alex@alexserver:~$ ps -fH
UID          PID    PPID  C STIME TTY          TIME CMD
alex        1354     662  0 18:19 tty1     00:00:00 -bash
alex        1645    1354  0 19:01 tty1     00:00:00   ps -fH
```

Рисунок 3 – Завершение с помощью PID

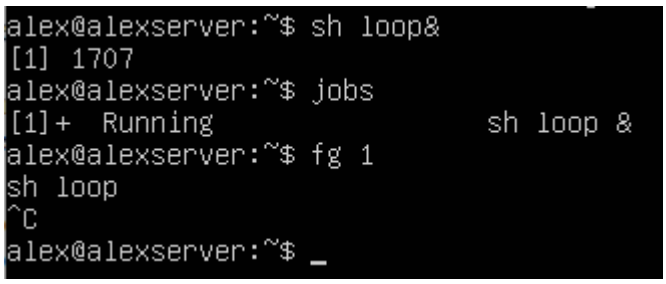

Рисунок 4 – Завершение с помощью комбинации клавиш (Ctrl+C)

3) Запустите редактор nano, определите приоритет редактора. Запустите новый процесс данного редактора с увеличенным на 2 значением приоритета.

Запускаем редактор nano (команда: nano &), определяем приоритет редактора (команда: renice –n 1 1685). Запускаем новый процесс редактора с увеличенным на 2 значением приоритета (команда: nice –n 3 nano &). Выведем процессы (ps –fHl). NI-значение nice.

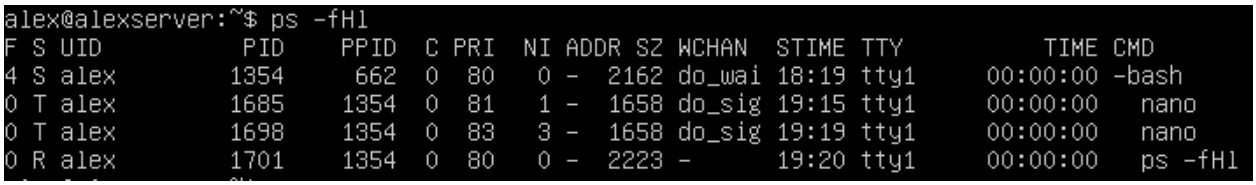

Рисунок 5 – Два процесса nano с разным nice

4) Вывести общую информацию о системе

echo $SHELL или ps –p $$ выведут текущий интерпретатор команд.

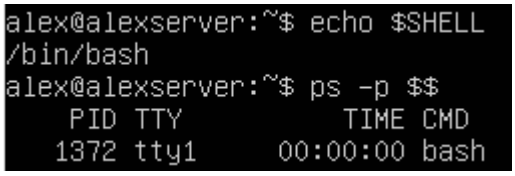

Рисунок 6 – Информация о текущем интерпретаторе команд

whoami выведет текущего пользователя.

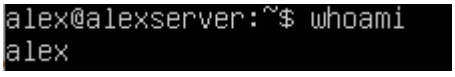

Рисунок 7 – Информация о текущем пользователе

Вывести информацию о текущем каталоге можно командой pwd.

Рисунок 8 – Информация о текущем каталоге

Команда free выведет информацию об оперативной памяти и файле подкачки.

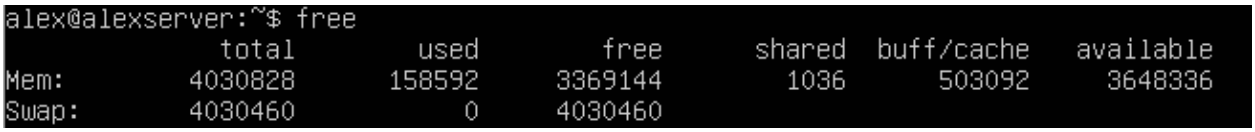

Рисунок 9 – Информация об оперативной памяти и области подкачки

df позволяет получить информацию о дисковой памяти.

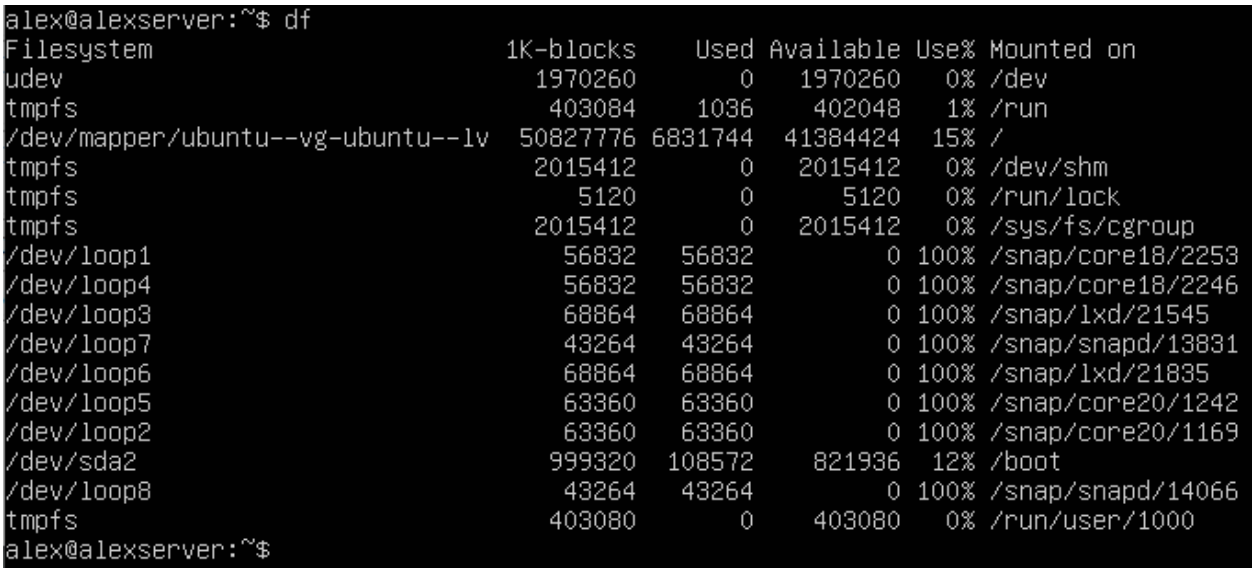

Рисунок 10 – Информация о дисковой памяти

5) Выполнить команды получения информации о процессах

Команда echo $$ или ps –p $$

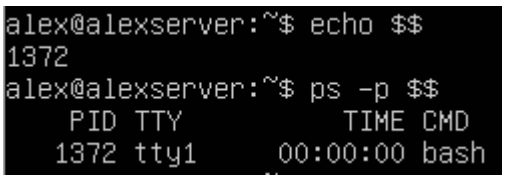

Рисунок 11 – Идентификатор текущего процесса (PID)

Рисунок 12 – Идентификатор родительского процесса (PPID)

Идентификатор можно получить с помощью команды pidof <имя процесса>, процесс init – первый запускаемый в системе процесс (не считая загрузки

ядра), который является родителем (прямым или косвенным) всех других запущенных процессов, и которому присваивается PID=1.

```
alex@alexserver:~$ pidof init
1
```

Рисунок 13 – Идентификатор процесса инициализации системы

Команда ps без параметра выведет информацию о выполняемых процессах текущего пользователя в текущем интерпретаторе команд.

```
alex@alexserver:~$ ps
    PID TTY          TIME CMD
   1372 tty1     00:00:00 bash
   1534 tty1     00:00:00 ps
```

Рисунок 14 – Информация о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

Команда ps –e отобразит все процессы.

```
alex@alexserver:~$ ps -e | head
    PID TTY          TIME CMD
      1 ?        00:00:03 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      6 ?        00:00:00 kworker/0:0H-kblockd
      8 ?        00:00:00 mm_percpu_wq
      9 ?        00:00:00 ksoftirqd/0
     10 ?        00:00:00 rcu_sched
     11 ?        00:00:00 migration/0
```

Рисунок 15 – Отображение всех процессов

6) Выполнить команды управления процессами

```
alex@alexserver:~$ ps
    PID TTY          TIME CMD
   1372 tty1     00:00:00 bash
   1541 tty1     00:00:00 ps
```

Рисунок 16 – Информация о выполняющихся процессах текущего пользователя в текущем интерпретаторе

Проверить значение nice можно с помощью команды nice.

```
alex@alexserver:~$ nice
0
```

Рисунок 17 – Определение значения nice по умолчанию

Команда nice –n 10 bash – запустить интерпретатор с понижением приоритета на 10.

```
alex@alexserver:~$ nice -n 10 bash
alex@alexserver:~$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1680    1600  0  80   0 -  2066 do_wai tty1     00:00:00 bash
0 S  1000    1690    1680  0  90  10 -  2068 do_wai tty1     00:00:00 bash
0 R  1000    1696    1690  0  90  10 -  2203 -      tty1     00:00:00 ps
```

Рисунок 18 – Запуск интерпретатора bash с понижением приоритета

```
alex@alexserver:~$ ps -p $$
    PID TTY          TIME CMD
   1690 tty1     00:00:00 bash
```

Рисунок 19 – Определение PID запущенного интерпретатора

Для повышения приоритета необходимы права root

```
alex@alexserver:~$ sudo renice -n 5 1690
1690 (process ID) old priority 10, new priority 5
```

Рисунок 20 – Установка приоритета запущенного интерпретатора равным 5

Командой ps lax можно отобразить информацию о значениях nice процессов

```
alex@alexserver:~$ ps lax | grep bash
4  1000    1680    1600  20   0   8264  5272 do_wai S    tty1       0:00 -bash
0  1000    1690    1680  25   5   8272  5148 do_wai SN   tty1       0:00 bash
0  1000    1777    1690  25   5   6300   724 -      RN+  tty1       0:00 grep --color=auto bash
```

Рисунок 21 – Получение информации о процессах bash

Контрольные вопросы

1) Перечислите состояния задачи в ОС Ubuntu.

Таблица 1. Состояния задачи Linux

Состояние	Описание
running	Задача переходит в состояние выполнения после выделения ей процессора.
sleeping	При блокировке задача переходит в состояние спячки.
stopped	При остановке работы задача переходит в режим остановки.
zombie	Состояние зомби показывает, что выполнение задачи прекратилось, однако она еще не была удалена из системы.
dead	Задача в состоянии смерти может быть удалена из системы.

Также при планировании выполнения процесса используются состояния active и expired.

2) Как создаются задачи в ОС Ubuntu?

Задачи создаются путем вызова системной функции clone.

Любые обращения к fork преобразуются в системные вызовы clone во время компиляции.

Каждый процесс порождается другим процессом, использующим для этого системный вызов fork(). Таким образом, структура процессов, подобно файловой системе, древовидна. Корнем этого дерева служит init — процесс инициализации системы. Он запускается ядром первым, получает идентификатор 1 и порождает еще несколько процессов (сколько и каких, можно узнать из его конфигурационного файла /etc/inittab), которые, в свою очередь, при участии пользователя порождают другие процессы. В результате системного вызова fork() родительский процесс полностью копирует свое окружение, включая адресное пространство, в дочерний, так что в момент рождения дочерний процесс отличается только своим ID. Потом дочерний процесс с помощью вызова exec() загружает в свое адресное пространство какой-нибудь исполняемый файл и начинает исполнять

содержащуюся в нем программу.

3) Назовите классы потоков ОС Ubuntu.

Потоки реального времени, обслуживаемые по алгоритму FIFO. Имеют наивысшие приоритеты и не могут прерываться другими потоками, за исключением такого же потока реального времени FIFO, перешедшего в состояние готовности.

Потоки реального времени, обслуживаемые в порядке циклической очереди. Представляют собой то же самое что и потоки FIFO, но могут прерываться таймером.

Потоки разделения времени.

4) Как используется приоритет планирования при запуске задачи?

У каждого потока есть приоритет планирования. В планировщике Linux приоритет задачи влияет на размер кванта времени и порядок выполнения задач процессором.

Во время создания каждой задаче присваивается статический приоритет, называемый также правильным значением (nice value).

Цель алгоритма планирования состоит в том, чтобы обеспечить грубое пропорциональное соответствие качества обслуживания приоритету, то есть чем выше приоритет, тем меньше должно быть время отклика и тем большая доля процессорного времени достанется процессу.

5) Как можно изменить приоритет для выполняющейся задачи?

Для того, чтобы изменить приоритет существующего процесса, необходимо воспользоваться командой renice. При понижении приоритета у процесса, который является вашим (т.е. запущен под той же учетной записью, под которой вы работаете в системе) - права суперпользователя не требуются, но при повышении приоритета у процесса, требуется запускать команду renice с правами суперпользователя