

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ



**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт

компьютерных наук

Кафедра

автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине "Операционные системы Linux"

На тему "Процессы и управление ими в операционной системе Linux"

Студент

ПИ-22-1

подпись, дата

Кистерёв В.А.

Руководитель

канд.техн.наук, доцент

ученая степень, ученое звание

подпись, дата

Кургасов В.В.

Липецк, 2024 г.

Оглавление

Цель работы	3
Ход работы	4
1. Часть I.....	4
2. Часть II	10
3. Часть III	13
4. Часть IV.....	16
Контрольные вопросы.....	21
Вывод	25

Цель работы

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

Ход работы

1. Часть I

1.1. Войти под пользовательской учетной записью (не root). Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.

Чтобы выяснить номер версии Linux по имени файла, нужно перейти в директорию boot командой `ls -l /boot`. Содержимое директории boot представлено на рисунке 1.

```
$ ls -l /boot
итого 76912
-rw-r--r-- 1 root root 259508 сен 30 22:08 config-6.1.0-26-amd64
drwxr-xr-x 5 root root 4096 окт 6 11:17 grub
-rw-r--r-- 1 root root 30956499 окт 6 11:19 initrd.img-6.1.0-26-amd64
-rw-r--r-- 1 root root 83 сен 30 22:08 System.map-6.1.0-26-amd64
-rw-r--r-- 1 root root 8185792 сен 30 22:08 vmlinuz-6.1.0-26-amd64
$ _
```

Рисунок 1 – Содержимое директории boot

Файл с образом ядра – `vmlinuz-6.1.0.26-amd64` (рисунок 1). Из названия файла: 6.1.0.26 – версия ядра.

1.2. Посмотреть процессы `ps -f`. Прокомментировать, изучив предварительно справку командой `man ps`.

Посмотрим процессы командой `ps -f` (рисунок 2).

```
$ ps -f
  UID          PID    PPID  C STIME TTY          TIME CMD
user          506      483  0 18:55 tty1        00:00:00 -sh
user          520      506  99 19:11 tty1        00:00:00 ps -f
$
```

Рисунок 2 – Вывод процессов

На рисунке 2: первый процесс – оболочка `bash` (запущена при входе в систему), второй – команда `ps -f`.

Команда `ps -f` в Linux выводит расширенный список процессов, запущенных в системе, с более подробной информацией о каждом процессе. Опция `-f` означает *full-format* и добавляет такие поля, как:

- 1) UID — пользователь, запустивший процесс,
- 2) PID — идентификатор процесса,
- 3) PPID — ID родительского процесса,

- 4) C — приоритет,
- 5) STIME — время запуска,
- 6) TTY — терминал, к которому прикреплен процесс,
- 7) TIME — суммарное процессорное время,
- 8) CMD — команда запуска процесса.

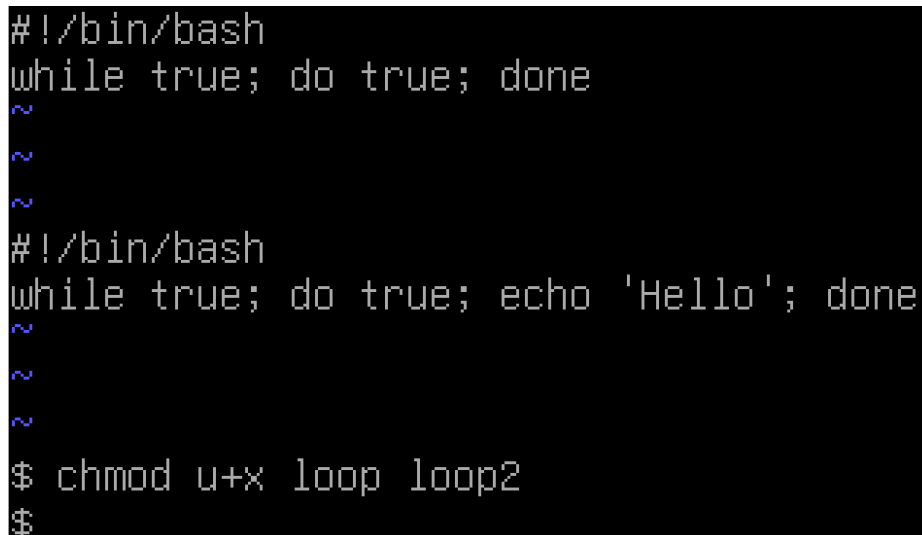
Эта команда полезна для получения полной информации о каждом процессе, что помогает в диагностике и управлении процессами.

1.3. Написать с помощью редактора vi два сценария loop и loop2.

С помощью редактора vi создадим два файла loop и loop2 (рисунок 3).

1) loop: while true; do true; done (бесконечный цикл без вывода в терминал);

2) loop2: while true; do true; echo 'Hello'; done (бесконечный цикл с выводом в терминал);



```
#!/bin/bash
while true; do true; done
~
~
~

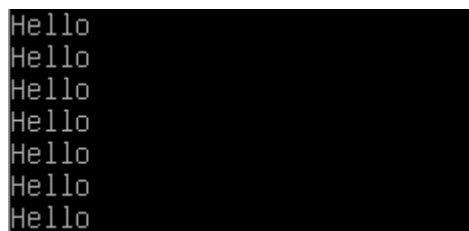
#!/bin/bash
while true; do true; echo 'Hello'; done
~
~
~

$ chmod u+x loop loop2
$
```

Рисунок 3 – Создание сценариев loop и loop2

1.4. Запустить loop2 на переднем плане.

Командой sh loop2 запустим процесс loop2 (рисунок 4).



```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Рисунок 4 – Результат работы loop2

1.5. Остановить, послав сигнал STOP.

Чтобы остановить процесс, можно послать сигнал STOP сочетанием клавиш CTRL + Z (рисунок 5).

```
Hello
Hello
^Z[2] + Stopped          sh loop2
$ _
```

Рисунок 5 – Остановка процесса

1.6. Посмотреть последовательно несколько раз ps -f. Записать сообщение, объяснить.

```
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user          525       517  0  20:23 tty1        00:00:00 -sh
user          531       525  3  20:24 tty1        00:00:00 sh loop2
user          533       525  0  20:24 tty1        00:00:00 ps -f
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user          525       517  0  20:23 tty1        00:00:00 -sh
user          531       525  1  20:24 tty1        00:00:00 sh loop2
user          534       525  0  20:24 tty1        00:00:00 ps -f
$ _
```

Рисунок 6 – Последовательный просмотр ps -f

Рисунок 7: TIME – затраченное время на выполнение процесса. После повторного выполнения команды ps -f время не изменилось, следовательно, процесс loop2 остановлен.

1.7. Убить процесс loop2, пошлав сигнал kill -9 PID. Записать сообщение. Прокомментировать.

Командой kill -9 531 убьём процесс loop2 (рисунок 8).

```
$ kill -9 531
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user          525       517  0  20:23 tty1        00:00:00 -sh
user          535       525  0  20:25 tty1        00:00:00 ps -f
[1] + Killed                  sh loop2
$ _
```

Рисунок 7 – Выполнение команды kill

Команда kill -9 PID отправляет процессу с указанным идентификатором (PID) сигнал SIGKILL, который принудительно завершает его без возможности корректной обработки. Это жесткий метод завершения процесса, который немедленно прерывает выполнение, игнорируя любые процедуры завершения.

1.8. Запустить в фоне процесс loop: `sh loop &`. Не останавливая, посмотреть несколько раз `ps -f`. Записать значение, объяснить.

Командой `sh loop &` запустим процесс в фоне (рисунок 9).

```
$ sh loop &
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user          525       517  0  20:23 tty1        00:00:00 -sh
user          547       525  99  20:30 tty1        00:00:03 sh loop
user          548       525   0  20:30 tty1        00:00:00 ps -f
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user          525       517  0  20:23 tty1        00:00:00 -sh
user          547       525  99  20:30 tty1        00:00:09 sh loop
user          549       525   0  20:30 tty1        00:00:00 ps -f
$
```

Рисунок 8 – Запуск процесса в фоне

Рисунок 9: при первом запуске команды `ps -f` процесс `loop` выполнялся 9 секунд, а при втором запуске `ps -f` – 9 секунд, следовательно, процесс корректно работает в фоновом режиме.

1.9. Завершить процесс `loop` командой `kill -15 PID`. Записать сообщение, прокомментировать.

Командой `kill -15 PID` завершим процесс `loop` (рисунок 10).

```
$ kill -15 547
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user          525       517  0  20:23 tty1        00:00:00 -sh
user          550       525   0  20:31 tty1        00:00:00 ps -f
[1] + Terminated
$ _
```

Рисунок 9 – Завершение процесса loop

Команда `kill -15 PID` отправляет процессу сигнал `SIGTERM`, который корректно завершает процесс.

1.10. Третий раз запустить в фоне. Не останавливая, убить командой `kill -9 PID`.

Запустим процесс `loop` в фоне, принудительно завершим командой `kill -9 PID` (рисунок 11).

```

$ sh loop &
$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
user          525      517  0  20:23 tty1        00:00:00 -sh
user          557      525  99  20:39 tty1        00:00:02 sh loop
user          558      525  0  20:39 tty1        00:00:00 ps -f
$ kill -9 557
$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
user          525      517  0  20:23 tty1        00:00:00 -sh
user          559      525  0  20:39 tty1        00:00:00 ps -f
[1] + Killed                  sh loop
$ _

```

Рисунок 10 – Принудительное завершение процесса

1.11. Запустить еще один экземпляр оболочки: bash.

Воспользуемся сочетанием клавиш CTRL + ALT + F<номер оболочки>, чтобы запустить еще один экземпляр оболочки (рисунок 12).

```

Debian GNU/Linux 12 debian tty3

debian login: user
Password:
Linux debian 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Oct 25 20:23:45 MSK 2024 on tty1
$ _

```

Рисунок 11 – Запуск нового экземпляра оболочки

1.12. Запустить процесс в фоне. Останавливать и снова запускать его.

Записать результаты просмотра командой ps -f.

Командой sh loop & запустим процесс в фоновом режиме, командой kill -STOP PID остановим его, потом снова запустим командой kill -CONT PID (рисунок 13).


```

$ sh loop &
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user         569       564  0  20:43 tty3        00:00:00 -sh
user         575       569 99  20:45 tty3        00:00:01 sh loop
user         576       569  0  20:45 tty3        00:00:00 ps -f
$ kill -STOP 575
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user         569       564  0  20:43 tty3        00:00:00 -sh
user         575       569 82  20:45 tty3        00:00:16 sh loop
user         577       569  0  20:45 tty3        00:00:00 ps -f
[1] + Stopped (signal)          sh loop
$ kill -CONT 575
$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
user         569       564  0  20:43 tty3        00:00:00 -sh
user         575       569 49  20:45 tty3        00:00:21 sh loop
user         578       569  0  20:46 tty3        00:00:00 ps -f
$

```

Рисунок 12 – Остановка и возобновление процесса loop

2. Часть II

2.1. Запустить в консоли на выполнение три задачи: две в интерактивном режиме, одну – в фоновом.

Запустим процесс loop в фоновом режиме в TTY1, процесс loop2 в интерактивном режиме в TTY2, процесс loop3 (аналогичен процессу loop 2) в интерактивном режиме в TTY3, исследуем работу процессов командой ps -ef в TTY4 (рисунок 14).

```
$ ps -ef | grep loop
user        604      525 99 20:57 tty1      00:01:37 sh loop
user        605      596 38 20:57 tty2      00:00:34 sh loop2
user        606      569 32 20:57 tty3      00:00:26 sh loop3
user        623      612  0 20:58 tty4      00:00:00 grep loop
$ ps -ef | grep loop
user        604      525 99 20:57 tty1      00:01:46 sh loop
user        605      596 38 20:57 tty2      00:00:37 sh loop2
user        606      569 32 20:57 tty3      00:00:29 sh loop3
user        625      612  0 20:58 tty4      00:00:00 grep loop
$ _
```

Рисунок 13 – Исследование работы процессов

Команда ps -ef | grep loop:

ps -ef: Отображает список всех запущенных процессов в системе в формате полного списка (-e для всех процессов, -f для полного формата с деталями, такими как PID, пользователь, команда, время старта и т.д.).

grep loop: Фильтрует вывод команды ps -ef, показывая только те строки, которые содержат слово "loop".

2.2. Перевести одну из задач, выполняющихся в интерактивном режиме, в фоновый режим.

Остановим процесс loop2 (CTRL + Z), возобновим в фоновом режиме командой bg %<job number> (рисунок 15).

```
Hello
Hello
^Z[1] + Stopped          sh loop3
$ bg %1
$ ps -ef | grep loop
user        604      525 99 20:57 tty1      00:11:47 sh loop
user        605      596 36 20:57 tty2      00:04:15 sh loop2
user        606      569 33 20:57 tty3      00:03:49 sh loop3
user        639      612  0 21:08 tty4      00:00:00 grep loop
$
```

Рисунок 14 – Перевод процесса в фоновый режим

2.3. Провести эксперименты по переводу задач из фонового режима в интерактивный и наоборот.

Остановим процесс loop3 (kill -STOP PID), возобновим в интерактивном режиме командой fg %<job number> (рисунок 16).

```
$ ps -ef | grep loop
user      604      525 99 20:57 tty1      00:11:47 sh loop
user      605      596 36 20:57 tty2      00:04:15 sh loop2
user      606      569 33 20:57 tty3      00:03:49 sh loop3
user      639      612  0 21:08 tty4      00:00:00 grep loop
$ kill -STOP 606
$ fg %1_
```

Рисунок 15 – Перевод процесса в интерактивный режим

2.4. Создать именованный канал для архивирования и осуществить передачу в канал списка файлов домашнего каталога вместе с подкаталогами и одного каталога вместе с файлами и подкаталогами.

Именованный канал — это специальный тип файла в Unix-подобных операционных системах, который используется для организации обмена данными между процессами. Командой mkfifo <name> создадим именованный канал, командой ls -R выведем список файлов с перенаправлением в именованный канал test (рисунок 17).

```
$ mkfifo test
$ ls -R ~ > test
-
```

Рисунок 16 – Создание именованного канала

Командой cat < test выведем содержимое канала test в ТТУ5 (рисунок 18).

```
$ cat < test
/home/user:
1.txt
2.txt
3.txt
loop
loop2
loop3
main.py
script.sh
test
$
```

Рисунок 17 – Вывод содержимого test

Именованный канал предоставляет механизм для передачи данных от одного процесса к другому, позволяя процессам взаимодействовать друг с другом.

Командой `touch` создадим файл `fail.txt`, передадим в именованный канал `test` список файлов домашнего каталога, перейдём в другой терминал и командой `cat < test > file.txt` считаем данные из канала и запишем их в `file.txt`, посмотрим содержимое `file.txt` (рисунок 19).



```
$ cat < test > file.txt
$ cat file.txt
/home/user:
1.txt
2.txt
3.txt
file.txt
loop
loop2
loop3
main.py
script.sh
test
$
```

Рисунок 19 – Запись данных из именованного канала в файл

3. Часть III

3.1. Получить следующую информацию о процессах текущего пользователя: идентификатор и имя владельца процесса, статус и приоритет процесса.

Пример получения информации о процессах текущего пользователя представлен на рисунке 20.

```
$ ps -o pid,user,stat,pri -u $(whoami)
  PID  USER   STAT  PRI
   485  user    Ss     19
   486  user    S      19
   501  user    S      19
   515  user    R+     19
$
```

Рисунок 20 – Вывод процессов

`ps` — выводит информацию о процессах.

`-o` — позволяет указать столбцы (поля), которые вы хотите увидеть в выводе. В данном случае это:

`pid` — идентификатор процесса.

`user` — имя владельца процесса.

`stat` — статус процесса.

`pri` — приоритет процесса.

`-u $(whoami)` — фильтрует процессы для текущего пользователя. `$(whoami)` подставляет имя текущего пользователя.

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

Командой `ps -fu` посмотрим процессы, запущенные текущим пользователем (рисунок 21).

```
$ ps -fu user
  UID    PID  PPID  C  STIME TTY          TIME CMD
  user    485     1  0  23:45 ?        00:00:00 /lib/systemd/systemd --user
  user    486    485  0  23:45 ?        00:00:00 (sd-pam)
  user    501    472  0  23:45 tty1    00:00:00 -sh
  user    519    501  99  23:57 tty1    00:00:51 sh loop
  user    526    520  0  23:57 tty2    00:00:00 -sh
  user    531    526  99  23:57 tty2    00:00:20 sh loop2
  user    538    532  0  23:57 tty3    00:00:00 -sh
  user    543    538  0  23:57 tty3    00:00:00 ps -fu user
```

Рисунок 21 – Процессы пользователя

Командой `pkill -SIGINT -f "loop2"` завершим выполнение процесса `loop2` (рисунок 22).

```
$ pkill -SIGINT -f "loop2"
$ ps -fu user
  UID      PID     PPID  C  STIME TTY          TIME CMD
user        485         1  0  23:45 ?        00:00:00 /lib/systemd/systemd --user
user        486        485  0  23:45 ?        00:00:00 (sd-pam)
user        501        472  0  23:45 tty1      00:00:00 -sh
user        519        501  99 23:57 tty1      00:01:57 sh loop
user        526        520  0  23:57 tty2      00:00:00 -sh
user        538        532  0  23:57 tty3      00:00:00 -sh
user        545        538  0  23:59 tty3      00:00:00 ps -fu user
$ _
```

Рисунок 22 – Завершение процесса `loop2`

Команда `pkill - SIGINT -f "loop2"` завершает все процессы, имя или командная строка которых содержат строку `"loop2"`, отправляя им сигнал `SIGINT`. Этот сигнал немедленно завершает процесс, не давая ему возможности обработать его или корректно завершиться.

Командой `kill -3 (SIGQUIT) PID` завершим выполнение процесса `loop` (рисунок 23).

```
$ kill -3 519
$ ps -fu
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user       538  0.0  0.0   2576    900 tty3      S   окт25    0:00 -sh
user       607  0.0  0.2  11084   4416 tty3      R+   00:00    0:00 \_ ps -fu
user       526  0.0  0.0   2576    896 tty2      S+   окт25    0:00 -sh
user       501  0.0  0.0   2576   1636 tty1      S+   окт25    0:00 -sh
$
```

Рисунок 18 – Завершение процесса `loop`

3.3. Определить идентификаторы и имена процессов, идентификатор группы которых не равен идентификатору группы текущего пользователя.

Вывод идентификаторов процессов, идентификатор группы которых не равен идентификатору группы текущего пользователя представлен на рисунке 24.

```
56 root
57 root
72 root
77 root
78 root
132 root
133 root
134 root
145 root
149 root
150 root
151 root
152 root
```

Рисунок 19 – Вывод идентификаторов процессов

Использованная команда: `ps -eo pid,user,group | awk -v gid=$(id -g) '$3 != gid {print $1, $2}'`.

`ps -eo pid,user,group:`

`-e` — отображает все процессы в системе.

`-o pid,user,group` — указывает, какие столбцы отобразить: идентификатор процесса (`pid`), имя владельца (`user`), и идентификатор группы (`group`).

`awk -v gid=$(id -g) '$3 != gid {print $1, $2}':`

`-v gid=$(id -g)` — присваивает переменной `gid` значение идентификатора группы текущего пользователя. Команда `id -g` выводит GID текущего пользователя.

`$3 != gid` — фильтрует процессы, у которых идентификатор группы (`$3`) отличается от `gid`.

`{print $1, $2}` — выводит идентификатор процесса (`$1`) и имя владельца (`$2`).

4. Часть IV

4.1. Вывести общую информацию о системе

4.1.1. Вывести информацию о текущем интерпретаторе команд

Командой `echo $SHELL` выведем путь к текущему интерпретатору команд (рисунок 25).

```
$ echo $SHELL
/bin/sh
$ _
```

Рисунок 20 – Путь к интерпретатору команд

4.1.2. Вывести информацию о текущем пользователе

Командой `whoami` выведем текущее имя пользователя (рисунок 26).

```
$ whoami
user
$ _
```

Рисунок 21 – Текущий пользователь

4.1.3. Вывести информацию о текущем каталоге

Командой `pwd` выведем путь от корневого каталога до текущего каталога (рисунок 27).

```
$ pwd
/home/user
$
```

Рисунок 22 – Команда `pwd`

4.1.4. Вывести информацию об оперативной памяти и области подкачки

Командой `free` выведем информацию об оперативной памяти и области подкачки (рисунок 28).

```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:          1,9Gi        207Mi        1,7Gi        568Ki         86Mi        1,7Gi
Swap:          975Mi           0B          975Mi
```

Рисунок 23 – Информация об оперативной памяти и области подкачки

4.1.5. Вывести информацию о дисковой памяти

Командой `df` выведем информацию о использовании дискового пространства (рисунок 29).

Файловая система	Размер	Использовано	Дост	Использовано%	Смонтировано в
udev	965M	0	965M	0%	/dev
tmpfs	197M	568K	197M	1%	/run
/dev/sda1	6,8G	2,2G	4,3G	34%	/
tmpfs	984M	0	984M	0%	/dev/shm
tmpfs	5,0M	0	5,0M	0%	/run/lock
/dev/sda6	12G	232K	12G	1%	/home
tmpfs	197M	0	197M	0%	/run/user/1001

Рисунок 24 – Информация о дисковой памяти

4.2. Выполнить команды получения информации о процессах

4.2.1. Получить идентификатор текущего процесса

Командой `echo $$` выведем идентификатор процесса оболочки, в котором выполняется команда (рисунок 30).

```
$ echo $$
521
$ ps -f
  UID          PID     PPID  C  STIME TTY          TIME CMD
user          521        516  0  22:36 tty3        00:00:00 -sh
user          596        521  0  23:22 tty3        00:00:00 ps -f
$
```

Рисунок 25 – Идентификатор текущего процесса

4.2.2. Получить идентификатор родительского процесса

Идентификатор родительского процесса хранится в переменной `PPID`: 516 (рисунок 30).

4.2.3. Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

Командой `ps -fu` выведем информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд (рисунок 31).

```
$ ps -fu
  USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user          521   0.0   0.0   2576   1612 tty3      S   22:36   0:00 -sh
user          599   0.0   0.2  11084   4440 tty3      R+  23:27   0:00 \_ ps -fu
user          512   0.0   0.0   2576    892 tty2      S+  22:36   0:00 -sh
user          501   0.0   0.0   2576    884 tty1      S+  22:35   0:00 -sh
$ _
```

Рисунок 26 – Информация о процессах текущего пользователя

4.2.4. Отобразить все процессы

Все процессы можно отобразить командой `ps -fe` (рисунок 32).

```

root      426      1  0 22:35 ?        00:00:00 /usr/sbin/cron -f
message+  429      1  0 22:35 ?        00:00:00 /usr/bin/dbus-daemon --system --adre
root      436      1  0 22:35 ?        00:00:00 /lib/systemd/systemd-logind
root      442      1  0 22:35 ?        00:00:00 dhclient -4 -v -i -pf /run/dhclient.e
root      464      1  0 22:35 tty1    00:00:00 /bin/login -p --
root      478      1  0 22:35 ?        00:00:00 sshd: /usr/sbin/sshd -D [listener] 0
user      485      1  0 22:35 ?        00:00:00 /lib/systemd/systemd --user
user      486     485  0 22:35 ?        00:00:00 (sd-pam)
user      501     464  0 22:35 tty1    00:00:00 -sh
root      507      1  0 22:36 tty2    00:00:00 /bin/login -p --
user      512     507  0 22:36 tty2    00:00:00 -sh
root      516      1  0 22:36 tty3    00:00:00 /bin/login -p --
user      521     516  0 22:36 tty3    00:00:00 -sh
root      549      2  0 22:56 ?        00:00:00 [kworker/3:0]
root      560      2  0 22:58 ?        00:00:00 [kworker/1:2]
root      582      2  0 23:12 ?        00:00:01 [kworker/0:0+events]
root      592      2  0 23:21 ?        00:00:00 [kworker/u8:2-events_unbound]
root      594      2  0 23:22 ?        00:00:00 [kworker/0:1-ata_sff]
root      600      2  0 23:27 ?        00:00:00 [kworker/0:2-ata_sff]
user      602     521  0 23:29 tty3    00:00:00 ps -fe
$ _

```

Рисунок 27 – Вывод всех процессов

4.3. Выполнить команды управления процессами

4.3.1. Определить текущее значение nice по умолчанию

Командой nice узнаем значение nice по умолчанию (рисунок 33).

```

$ nice
0
$ _

```

Рисунок 28 – Значение nice по умолчанию

4.3.2. Запустить интерпретатор sh с понижением приоритета nice -n 10 bash

Командой nice -n 10 sh понизим приоритет интерпретатора (рисунок 34).

```

$ nice -n 10 sh
$ ps -o pid,ni,cmd
  PID  NI CMD
   521   0 -sh
   607  10 bash
   610  19 sh
   613  19 sh
   614  19 ps -o pid,ni,cmd
$ _

```

Рисунок 29 – Понижение приоритета интерпретатора

4.3.3. Определить PID запущенного интерпретатора

Командой echo \$\$ определим PID запущенного интерпретатора (рисунок 35).

```

$ echo $$
613
$ _

```

Рисунок 30 – PID запущенного интерпретатора

4.3.4 Установить приоритет запущенного интерпретатора равным 5

Командой `sudo renice -n 5 <PID>` установим приоритет запущенного интерпретатора равным 5 (рисунок 36).

```
$ sudo renice -n 5 613
[sudo] пароль для user:
613 (process ID) old priority 19, new priority 5
$ _
```

Рисунок 36 – Изменение приоритета запущенного интерпретатора

4.3.5. Получить информацию о процессах bash: `ps lax | grep bash`

Командой `ps lax | grep sh` получим информацию о процессах (рисунок 37).

```
$ ps -lax | grep sh
1  0      5      2  0 -20      0  0 -      I<  ?      0:00 [slub_flushwq]
1  0      9      2  20  0      0  0 -      I  ?      0:00 [kworker/u8:0-flush-8:0]
5  0     36      2  20  0      0  0 -      I  ?      0:00 [kworker/u8:2-flush-8:0]
1  0     77      2  0 -20      0  0 -      I<  ?      0:00 [zswap-shrink]
4  0    477      1  20  0    15432 8852 -      Ss  ?      0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
4 1001   501     472  20  0     2576 1636 do_wai S    tty1    0:00 -sh
0 1001   508     501  20  0     6356 2204 pipe_r S+   tty1    0:00 grep sh
$
```

Рисунок 31 – Информация о процессах

`ps -lax:`

`l` — опция, которая указывает `ps` отображать процессы в «длинном» формате. В этом формате выводится больше информации, включая такие поля, как идентификатор пользователя (UID), приоритет процесса (PRI), и другие.

`a` — опция, показывающая информацию обо всех процессах, запущенных от текущего пользователя и имеющих доступ к терминалу, а не только о тех, что запущены в текущем терминале.

`x` — опция, включающая в вывод процессы, не привязанные к терминалу (например, фоновые демоны).

`grep bash:`

`grep` — утилита для поиска строк, соответствующих определенному шаблону.

`sh` — шаблон, по которому `grep` будет искать строки. В данном случае это строки, содержащие `sh`.

Вывод будет включать информацию о каждом процессе:

UID — идентификатор пользователя, запустившего процесс.

PID — идентификатор процесса.

PPID — идентификатор родительского процесса.

PRI и NI — приоритет и "nice" уровень (приоритет на уровне планировщика).

SZ и RSS — объем используемой памяти (в страницах памяти).

COMMAND — команда, которой был запущен процесс, в данном случае bash.

Контрольные вопросы

1. Перечислите состояния задачи в ОС Linux.

- TASK_RUNNING: выполняется или готово к выполнению.
- TASK_INTERRUPTIBLE: ожидает события и может быть прервано сигналом.
- TASK_UNINTERRUPTIBLE: ожидает события, но не прерывается сигналом.
- TASK_STOPPED: приостановлено.
- TASK_ZOMBIE: завершено, но еще не освобождено.
- TASK_DEAD: полностью завершено и удалено из системы.

2. Как создаются задачи в ОС Linux?

Задачи создаются с помощью системных вызовов fork(), vfork() и clone():

fork() создаёт новый процесс, являющийся точной копией родительского.

Оба процесса начинают выполняться независимо.

vfork() также создаёт копию, но блокирует родителя до завершения выполнения команды exec.

clone() используется для создания потоков, указывая параметры совместного использования памяти и других ресурсов.

3. Назовите классы потоков ОС Linux.

- SCHED_OTHER (обычный).
- SCHED_FIFO (реального времени, с FIFO-планировкой).
- SCHED_RR (реального времени, с планировкой Round Robin).
- SCHED_IDLE (низкий приоритет).
- SCHED_BATCH (для долгих вычислений).

4. Как используется приоритет планирования при запуске задачи?

Приоритет определяет порядок выполнения задач. Чем ниже значение, тем выше приоритет. Используется nice для задания приоритета.

5. Объясните, что произойдет, если запустить программу в фоновом режиме без подавления потока вывода.

Если запустить команду в фоне с символом `&` и не перенаправить вывод, он будет выводиться в терминал, мешая другим командам.

6. Объясните разницу между действием сочетаний клавиш `Ctrl^Z` и `Ctrl^C`.

`Ctrl+Z`: приостанавливает задачу.

`Ctrl+C`: прерывает (завершает) задачу.

7. Опишите, что значит каждое поле вывода команды `jobs`.

Отображает идентификатор задания, статус (работает, приостановлено), и команду, связанную с заданием.

8. Назовите главное отличие утилиты `top` от `jobs`.

– `top` показывает активные процессы системы; `jobs` — только задания текущей оболочки.

9. В чем отличие результата выполнения команд `top` и `htop`?

– `htop` — более продвинутый инструмент, позволяет сортировать процессы, поддерживает цветовую индикацию и более удобный интерфейс.

– `top` — базовый монитор процессов, отображающий только основные сведения.

10. Какую комбинацию клавиш нужно использовать для принудительного завершения задания, запущенного в интерактивном режиме?

– `Ctrl+C`.

11. Какую комбинацию клавиш нужно использовать для приостановки задания, запущенного в интерактивном режиме?

– `Ctrl+Z`.

12. Какая команда позволяет послать сигнал конкретному процессу?

Команда `kill` используется для отправки сигналов. Пример: `kill -9 PID` завершит процесс.

13. Какая команда позволяет поменять поправку к приоритету уже запущенного процесса?

Для изменения приоритета используется `renice`. Пример: `renice 5 -p 1234` изменит приоритет процесса с PID 1234.

14. Какая команда позволяет запустить задание с пониженным приоритетом?

Команда `nice` запускает команду с указанным приоритетом. Пример: `nice -n 10 ./script.sh`.

15. Какая команда позволяет запустить задание с защитой от прерывания при выходе из системы пользователя?

Команда `nohup` позволяет продолжить выполнение после выхода. Пример: `nohup ./script.sh &`.

16. Какой процесс всегда присутствует в системе и является предком всех процессов?

Процесс `init` (или `systemd`) всегда присутствует и является предком для всех других процессов.

17. Каким образом можно запустить задание в фоновом режиме?

Добавить `&` после команды: `./script.sh &`.

18. Каким образом задание, запущенное в фоновом режиме, можно перевести в интерактивный режим?

Использовать команду `fg %номер_задания`, например `fg %1`.

19. Каким образом приостановленное задание можно перевести в интерактивный режим?

Выполнить `fg` для перевода в интерактивный режим.

20. Что произойдет с заданием, выполняющимся в фоновом режиме, если оно попытается обратиться к терминалу?

Задание остановится до освобождения терминала.

21. Сколько терминалов может быть открыто в одной системе? Как перемещаться между терминалами (какие комбинации клавиш необходимо использовать)?

Количество виртуальных терминалов в системе обычно по умолчанию составляет 6, но может быть увеличено в зависимости от конфигурации. Перемещаться между ними можно с помощью комбинаций клавиш `CTRL + Alt + Fn`, где `n` – номер виртуального терминала.

22. В чем отличие идентификаторов PID и PPID? При каких условиях возможна ситуация, когда PPID равен нулю или отсутствует?

PID — идентификатор процесса; PPID — идентификатор родительского процесса. У init PPID равен 0.

23. Поясните, от чего зависит максимальное значение PID?

Максимальное значение PID в системе зависит от настроек ядра и может быть изменено с помощью файла `/proc/sys/kernel/pid_max`.

24. В каком случае, при создании нового процесса, его идентификатор (PID) будет меньше, чем у процесса, запущенного ранее?

Если PID достигнет максимума, система начнет переназначение с минимального значения.

Вывод

В ходе выполнения задачи были изучены основные понятия процесса в операционной системе и приобретены практические навыки управления процессами в Linux.