Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по курсу «ОС Linux»

Вариант 5

Студент _____ Морозов Д.С.

Группа ПИ-21-1

Руководитель _____ Кургасов В.В.

Липецк 2023 г.

Задание кафедры

Порядок выполнения работы

Часть I

1. Войти в систему под пользовательской учётной записью (не root).

2. Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.

3. Посмотреть процессы ps –f. Прокомментировать, изучив предварительно справку командой man ps.

4. Написать с помощью редактора vi два сценария loop и loop2. Текст сценария Loop:

while true; do true; done.

Текст сценария Loop2:

while true; do true; echo ‘Hello’; done.

5. Запустить loop2 на переднем плане: sh loop2.

6. Остановить, послав сигнал STOP.

7. Посмотреть последовательно несколько раз ps –f. Записать сообщение, объяснить.

8. Убить процесс loop2, послав сигнал kill -9 PID. Записать сообщение.

Прокомментировать.

9. Запустить в фоне процесс loop: sh loop&. Не останавливая, посмотреть несколькораз: ps –f. Записать значение, объяснить.

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

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

12. Запустить еще один экземпляр оболочки: bash.

13. Запустить несколько процессов в фоне. Останавливать их и снова запускать.Записать результаты просмотра командой ps –f.

Часть II

1. Запустить в консоли на выполнение три задачи: две в интерактивном режиме, одну - в фоновом.

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

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

4. Создать именованный канал для архивирования и осуществить передачу в канал:

a) списка файлов домашнего каталога вместе с подкаталогами (ключ -R);

b) одного каталога вместе с файлами и подкаталогами.

5. В отчете предоставьте все шаги ваших действий. То есть следует

привести следующее: текст задания, а следом за ним снимок экрана консоли с результатами выполнения задания. Кроме того, перед скриншотом следует при- вести текстовую запись использованных команд.

Часть III

Вариант 5

1. Отобразить информацию о процессах указанного пользователя в видеиерархии, вывод отсортировать по значениям PID.

2. С помощью сигнала SIGSTOP приостановить выполнение процесса, владельцем которого является текущий пользователь. Через несколько секунд возобновить выполнение процесса.

3. Определить идентификаторы и имена процессов, не связанных с указанным терминалом.

4. В отчете предоставьте все шаги ваших действий. То есть следует привести следующее: текст задания, а следом за ним снимок экрана консоли с результатами выполнения задания. Кроме того, перед скриншотом следует при- вести текстовую запись использованных команд. Кратко поясните результаты выполнения всех команд.

Часть IV

1. Открыть окно интерпретатора команд.

2. 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.

3. 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.

4. Выполнить команды управления процессами:

a) получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе;

b) определить текущее значение nice по умолчанию;

c) запустить интерпретатор bash с понижением приоритета nice –n 10 bash;

d) определить PID запущенного интерпретатора;

e) установить приоритет запущенного интерпретатора равным 5

renice –n 5 <PID процесса>;

f) получить информацию о процессах bash ps lax | grep bash.

Ход работы

Часть 1

Просмотр версии ядра Linux:

```
daniilpi@ubunto:/home$ cd ../../..
daniilpi@ubunto:/$ ls -la
total 4194380
drwxr-xr-x  19 root root       4096 окт  9 17:44 .
drwxr-xr-x  19 root root       4096 окт  9 17:44 ..
lrwxrwxrwx   1 root root          7 авг 10 00:17 bin -> usr/bin
drwxr-xr-x   4 root root       4096 сен 24 19:13 boot
drwxr-xr-x  20 root root       4080 окт  9 20:32 dev
drwxr-xr-x  99 root root       4096 окт  9 20:04 etc
drwxr-xr-x   6 root root       4096 окт  9 19:44 home
lrwxrwxrwx   1 root root          7 авг 10 00:17 lib -> usr/lib
lrwxrwxrwx   1 root root          9 авг 10 00:17 lib32 -> usr/lib32
lrwxrwxrwx   1 root root          9 авг 10 00:17 lib64 -> usr/lib64
lrwxrwxrwx   1 root root         10 авг 10 00:17 libx32 -> usr/libx32
drwx------   2 root root      16384 сен 24 19:04 lost+found
drwxr-xr-x   2 root root       4096 авг 10 00:17 media
drwxr-xr-x   2 root root       4096 авг 10 00:17 mnt
drwxr-xr-x   2 root root       4096 авг 10 00:17 opt
dr-xr-xr-x 233 root root          0 окт  9 10:22 proc
drwx------   8 root root       4096 окт  9 19:28 root
drwxr-xr-x  31 root root        900 окт  9 17:07 run
lrwxrwxrwx   1 root root          8 авг 10 00:17 sbin -> usr/sbin
drwxr-xr-x   6 root root       4096 авг 10 00:22 snap
drwxr-xr-x   2 root root       4096 авг 10 00:17 srv
-rw-------   1 root root 4294967296 сен 24 19:12 swap.img
dr-xr-xr-x  13 root root          0 окт  9 10:22 sys
drwxrwxrwt  14 root root       4096 окт  9 19:28 tmp
drwxr-xr-x  14 root root       4096 авг 10 00:17 usr
drwxr-xr-x  13 root root       4096 авг 10 00:20 var
daniilpi@ubunto:/$ _
```

```
daniilpi@ubunto:/$ cd boot
daniilpi@ubunto:/boot$ ls -la
total 124748
drwxr-xr-x  4 root root      4096 сен 24 19:13 .
drwxr-xr-x 19 root root      4096 окт  9 17:44 ..
-rw-r--r--  1 root root    262053 сен  5 13:31 config-5.15.0-84-generic
drwxr-xr-x  5 root root      4096 сен 24 19:13 grub
lrwxrwxrwx  1 root root        28 сен 24 19:12 initrd.img -> initrd.img-5.15.0-84-generic
-rw-r--r--  1 root root 109556727 сен 24 19:13 initrd.img-5.15.0-84-generic
lrwxrwxrwx  1 root root        28 сен 24 19:12 initrd.img.old -> initrd.img-5.15.0-84-generic
drwx------  2 root root     16384 сен 24 19:04 lost+found
-rw-------  1 root root   6273857 сен  5 13:31 System.map-5.15.0-84-generic
lrwxrwxrwx  1 root root        25 сен 24 19:12 vmlinuz -> vmlinuz-5.15.0-84-generic
-rw-------  1 root root  11612712 сен  5 16:57 vmlinuz-5.15.0-84-generic
lrwxrwxrwx  1 root root        25 сен 24 19:12 vmlinuz.old -> vmlinuz-5.15.0-84-generic
daniilpi@ubunto:/boot$ _
```

Просмотр процессов командой ps -f:

```
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2693    2653  0 22:50 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$ _
```

С помощью редактора vi создаем два файла:

```
daniilpi@ubunto:~$ vi loop
```

```
#!/bin/bash
while true
do true
done_
```

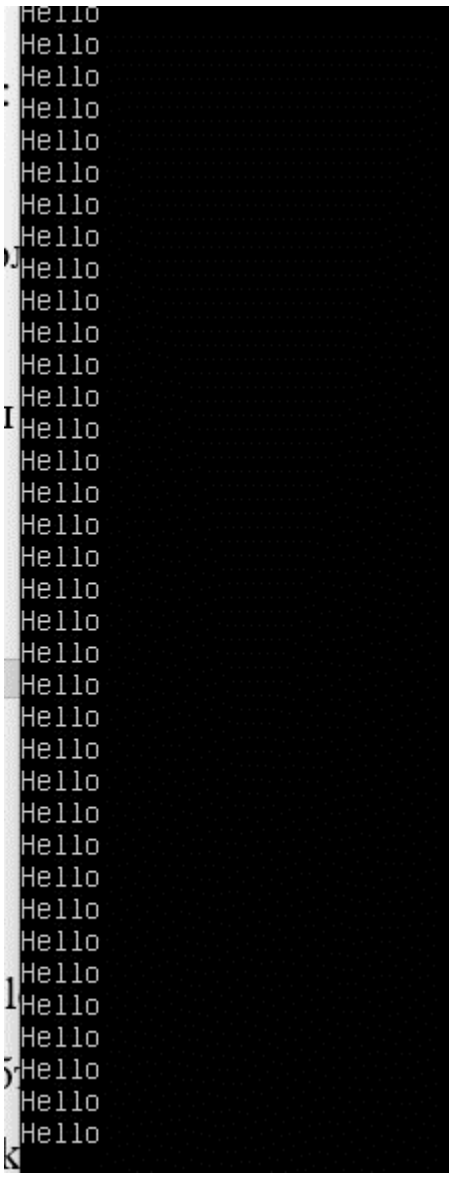
```
daniilpi@ubunto:~$ vi loop2_
```

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

```
daniilpi@ubunto:~$ ls -la
total 36
drwxr-x--- 2 daniilpi daniilpi 4096 окт  9 22:57 .
drwxr-xr-x 6 root     root     4096 окт  9 19:44 ..
-rw-r--r-- 1 daniilpi daniilpi  220 окт  9 19:42 .bash_logout
-rw-r--r-- 1 daniilpi daniilpi 3771 окт  9 19:42 .bashrc
-rwxrwxrwx 1 daniilpi daniilpi  125 окт  9 20:15 daniilpi
-rw-rw-r-- 1 daniilpi daniilpi   36 окт  9 22:54 loop
-rw-rw-r-- 1 daniilpi daniilpi   49 окт  9 22:57 loop2
-rw-r--r-- 1 daniilpi daniilpi  807 окт  9 19:42 .profile
-rw-r--r-- 1 daniilpi daniilpi    0 окт  9 20:06 .sudo_as_admin_successful
-rw------- 1 daniilpi daniilpi 2202 окт  9 22:57 .viminfo
daniilpi@ubunto:~$ _
```

Запускаем второй файл на переднем плане:

```
daniilpi@ubunto:~$ sh loop2
```



И с помощью ctrl+z останавливаем:

```
Hello
Hel^Z
[2]+  Stopped                 sh loop2
daniilpi@ubunto:~$ _
```

```
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2703    2653  0 23:03 pts/3    00:00:00 sh loop2
daniilpi    2704    2653  0 23:03 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$
```

Несколько раз выполняем команду ps -f:

```
[2]+  Stopped                 sh loop2
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2703    2653  0 23:03 pts/3    00:00:00 sh loop2
daniilpi    2704    2653  0 23:03 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2703    2653  0 23:03 pts/3    00:00:00 sh loop2
daniilpi    2705    2653  0 23:03 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2703    2653  0 23:03 pts/3    00:00:00 sh loop2
daniilpi    2706    2653  0 23:04 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$ _
```

Убиваем этот процесс:

```
daniilpi@ubunto:~$ kill -9 2703
[2]+  Killed                  sh loop2
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2714    2653  0 23:11 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$
```

Запускаем loop на фоне:

```
daniilpi@ubunto:~$ sh loop&
[2] 2715
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2715    2653 99 23:12 pts/3    00:00:13 sh loop
daniilpi    2717    2653  0 23:12 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$
```

Убиваем этот процесс:

```
daniilpi@ubunto:~$ kill -15 2715
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2724    2653  0 23:22 pts/3    00:00:00 ps -f
[2]-  Terminated              sh loop
daniilpi@ubunto:~$
```

Запускаем третий раз и тут же убиваем:

```
daniilpi@ubunto:~$ sh loop&
[2] 2726
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2726    2653 87 23:22 pts/3    00:00:05 sh loop
daniilpi    2727    2653  0 23:23 pts/3    00:00:00 ps -f
daniilpi@ubunto:~$ kill -9 2726
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    2653    2652  0 22:34 pts/3    00:00:00 -bash
daniilpi    2678    2653  0 22:49 pts/3    00:00:00 man ps
daniilpi    2686    2678  0 22:49 pts/3    00:00:00 pager
daniilpi    2728    2653  0 23:23 pts/3    00:00:00 ps -f
[2]-  Killed                  sh loop
daniilpi@ubunto:~$
```

Запускаем три процесса на фоне:

```
[2]+  Killed                  sh loop
daniilpi@ubunto:~$ sh loop&
[1] 1047
daniilpi@ubunto:~$ sh loop&
[2] 1048
daniilpi@ubunto:~$ sh loop&
[3] 1049
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1033     733  0 00:11 tty1     00:00:00 -bash
daniilpi    1047    1033 99 00:13 tty1     00:01:06 sh loop
daniilpi    1048    1033 99 00:13 tty1     00:01:02 sh loop
daniilpi    1049    1033 99 00:13 tty1     00:01:00 sh loop
daniilpi    1051    1033  0 00:14 tty1     00:00:00 ps -f
daniilpi@ubunto:~$
```

Останавливаем их:

```
daniilpi@ubunto:~$ sh loop&
[1] 1094
daniilpi@ubunto:~$ sh loop&
[2] 1095
daniilpi@ubunto:~$ sh loop&
[3] 1096
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1057    1056  0 00:50 pts/0    00:00:00 -bash
daniilpi    1094    1057 99 01:05 pts/0    00:00:13 sh loop
daniilpi    1095    1057 99 01:05 pts/0    00:00:09 sh loop
daniilpi    1096    1057 99 01:05 pts/0    00:00:06 sh loop
daniilpi    1097    1057  0 01:05 pts/0    00:00:00 ps -f
daniilpi@ubunto:~$ kill -19 1094

[1]+  Stopped                 sh loop
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1057    1056  0 00:50 pts/0    00:00:00 -bash
daniilpi    1094    1057 81 01:05 pts/0    00:00:31 sh loop
daniilpi    1095    1057 99 01:05 pts/0    00:00:32 sh loop
daniilpi    1096    1057 99 01:05 pts/0    00:00:29 sh loop
daniilpi    1100    1057  0 01:05 pts/0    00:00:00 ps -f
daniilpi@ubunto:~$ kill -19 1095
daniilpi@ubunto:~$ kill -19 1096

[2]+  Stopped                 sh loop
daniilpi@ubunto:~$ _
```

Часть 2

С помощью bg запускаем процессы на фоне, аналогично работает fg, но запускает процессы на переднем плане:

```
daniilpi@ubunto:~$ bg
[3]+ sh loop &
daniilpi@ubunto:~$ jobs -l
[1]-  1094 Stopped (signal)       sh loop
[2]+  1095 Stopped (signal)       sh loop
[3]   1096 Running                sh loop &
daniilpi@ubunto:~$ bg
[2]+ sh loop &
daniilpi@ubunto:~$ bg
[1]+ sh loop &
daniilpi@ubunto:~$ jobs
[1]   Running                 sh loop &
[2]-  Running                 sh loop &
[3]+  Running                 sh loop &
daniilpi@ubunto:~$ _
```

Убиваем процессы:

```
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1057    1056  0 00:50 pts/0    00:00:00 -bash
daniilpi    1094    1057 13 01:05 pts/0    00:01:13 sh loop
daniilpi    1095    1057 31 01:05 pts/0    00:02:49 sh loop
daniilpi    1096    1057 39 01:05 pts/0    00:03:28 sh loop
daniilpi    1107    1057  0 01:14 pts/0    00:00:00 ps -f
daniilpi@ubunto:~$ kill -9 1094 1095 1096
daniilpi@ubunto:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1057    1056  0 00:50 pts/0    00:00:00 -bash
daniilpi    1108    1057  0 01:14 pts/0    00:00:00 ps -f
[1]   Killed                  sh loop
[2]-  Killed                  sh loop
[3]+  Killed                  sh loop
daniilpi@ubunto:~$
```

Создаем именованный канал и осуществить передачу в канал каталога:

```
daniilpi@ubunto:~$ mkfifo katal7
daniilpi@ubunto:~$ ls -la
total 48
drwxr-x--- 5 daniilpi daniilpi 4096 окт 10 02:05 .
drwxr-xr-x 6 root     root     4096 окт  9 19:44 ..
-rw-r--r-- 1 daniilpi daniilpi  220 окт  9 19:42 .bash_logout
-rw-r--r-- 1 daniilpi daniilpi 3771 окт  9 19:42 .bashrc
drwx------ 2 daniilpi daniilpi 4096 окт  9 23:58 .cache
drwx------ 3 daniilpi daniilpi 4096 окт 10 01:03 .config
-rwxrwxrwx 1 daniilpi daniilpi  125 окт  9 20:15 daniilpi
drwxrwxr-x 2 daniilpi daniilpi 4096 окт 10 02:05 k1
prw-rw-r-- 1 daniilpi daniilpi    0 окт 10 02:05 katal7
-rw-rw-r-- 1 daniilpi daniilpi   36 окт  9 22:54 loop
-rw-rw-r-- 1 daniilpi daniilpi   49 окт  9 22:57 loop2
-rw-r--r-- 1 daniilpi daniilpi  807 окт  9 19:42 .profile
-rw-r--r-- 1 daniilpi daniilpi    0 окт  9 20:06 .sudo_as_admin_successful
-rw------- 1 daniilpi daniilpi 2202 окт  9 22:57 .viminfo
daniilpi@ubunto:~$ ls -l k1 >> katal7
```

Выводим полученный результат:

```
daniilpi@ubunto:~$ cat katal7
```

```
-rw-rw-r-- 1 daniilpi daniilpi 0 окт 10 02:05 1.txt
-rw-rw-r-- 1 daniilpi daniilpi 0 окт 10 02:05 2.txt
-rw-rw-r-- 1 daniilpi daniilpi 0 окт 10 02:05 3.txt
```

Осуществить передачу в канал списка файлов домашнего каталога вместе с подкаталогами:

```
daniilpi@ubunto:~$ ls -R > katal7_
```

```
daniilpi@ubunto:~$ cat katal7
```

```
.:
daniilpi  k1  katal7  loop  loop2

./k1:
1.txt  2.txt  3.txt
```

Часть 3

Выводим все процессы и останавливаем один из них:

```
daniilpi@ubunto:/home$ ps -fa
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1067     724  0 01:52 tty1     00:00:00 -bash
daniilpi    1088    1067  0 01:56 tty1     00:00:00 -bash
daniilpi    1115    1067  0 02:06 tty1     00:00:00 -bash
daniilpi    1123    1067 99 02:12 tty1     00:02:07 sh loop
daniilpi    1130    1067  0 02:14 tty1     00:00:00 ps -fa
daniilpi@ubunto:/home$ kill -19 1123
daniilpi@ubunto:/home$ ps -fa
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1067     724  0 01:52 tty1     00:00:00 -bash
daniilpi    1088    1067  0 01:56 tty1     00:00:00 -bash
daniilpi    1115    1067  0 02:06 tty1     00:00:00 -bash
daniilpi    1123    1067 96 02:12 tty1     00:02:21 sh loop
daniilpi    1131    1067  0 02:14 tty1     00:00:00 ps -fa

[5]+  Stopped                 sh loop  (wd: ~)
(wd now: /home)
daniilpi@ubunto:/home$ bg
```

После чего повторно запускаем:

```
daniilpi@ubunto:/home$ bg
[5]+ sh loop &  (wd: ~)
daniilpi@ubunto:/home$ ps -af
UID          PID    PPID  C STIME TTY          TIME CMD
daniilpi    1067     724  0 01:52 tty1     00:00:00 -bash
daniilpi    1088    1067  0 01:56 tty1     00:00:00 -bash
daniilpi    1115    1067  0 02:06 tty1     00:00:00 -bash
daniilpi    1123    1067 76 02:12 tty1     00:02:31 sh loop
daniilpi    1132    1067  0 02:15 tty1     00:00:00 ps -af
daniilpi@ubunto:/home$
```

С указанным терминалом у нас лишь не связан первый процесс.

Часть 4

Определяем какой пользователь у нас на данный момент:

```
daniilpi@ubunto:~$ who
daniilpi tty1         2023-10-10 01:52
```

Информация о текущем каталоге:

```
daniilpi@ubunto:~$ pwd
/home/daniilpi
```

Вся информация о дисковой памяти, оперативной памяти и области подкачки:

```
daniilpi@ubunto:~$ df -h
Filesystem                         Size  Used Avail Use% Mounted on
tmpfs                              578M  1,1M  577M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv   48G  7,4G   39G  17% /
tmpfs                              2,9G     0  2,9G   0% /dev/shm
tmpfs                              5,0M     0  5,0M   0% /run/lock
/dev/sda2                          2,0G  129M  1,7G   8% /boot
tmpfs                              578M  4,0K  578M   1% /run/user/1002
daniilpi@ubunto:~$
```

Получаем PID текущих процессов и PPID родительского процесса :

```
daniilpi@ubunto:~$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1002    1067     724  0  80   0 -  2185 do_wai tty1     00:00:00 bash
1 T  1002    1088    1067  0  80   0 -  2185 do_sig tty1     00:00:00 bash
1 T  1002    1115    1067  0  80   0 -  2185 do_sig tty1     00:00:00 bash
0 R  1002    1123    1067 96  80   0 -   722 -      tty1     00:21:57 sh
0 R  1002    1164    1067  0  80   0 -  2517 -      tty1     00:00:00 ps
daniilpi@ubunto:~$
```

Рапустить интерпретатор bash с понижением приоритета nice –n 10 bash:

```
daniilpi@ubunto:~$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1002    1067     724  0  80   0 -  2185 do_wai tty1     00:00:00 bash
1 T  1002    1088    1067  0  80   0 -  2185 do_sig tty1     00:00:00 bash
1 T  1002    1115    1067  0  80   0 -  2185 do_sig tty1     00:00:00 bash
0 R  1002    1123    1067 96  80   0 -   722 -      tty1     00:23:21 sh
0 S  1002    1165    1067  0  90  10 -  2166 do_wai tty1     00:00:00 bash
0 R  1002    1171    1165  0  90  10 -  2517 -      tty1     00:00:00 ps
daniilpi@ubunto:~$ _
```

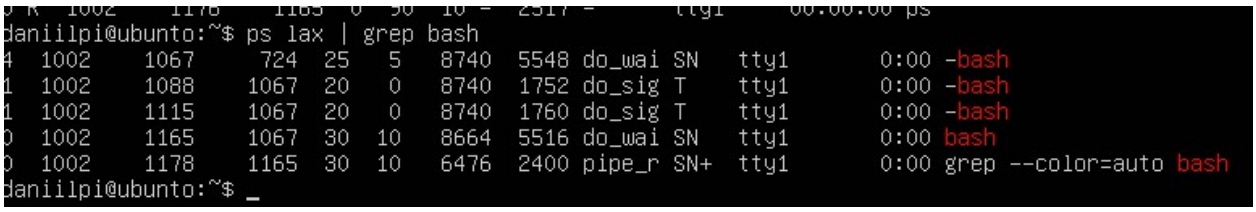
Установить приоритет запущенного интерпретатора равным 5

renice –n 5 <PID процесса>:

```
daniilpi@ubunto:~$ renice -n 5 1067
1067 (process ID) old priority 0, new priority 5
daniilpi@ubunto:~$ _
```

```
daniilpi@ubunto:~$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1002    1067     724  0  85   5 -  2185 do_wai tty1     00:00:00 bash
1 T  1002    1088    1067  0  80   0 -  2185 do_sig tty1     00:00:00 bash
1 T  1002    1115    1067  0  80   0 -  2185 do_sig tty1     00:00:00 bash
0 R  1002    1123    1067 97  80   0 -   722 -      tty1     00:26:30 sh
0 S  1002    1165    1067  0  90  10 -  2166 do_wai tty1     00:00:00 bash
0 R  1002    1176    1165  0  90  10 -  2517 -      tty1     00:00:00 ps
daniilpi@ubunto:~$ _
```

Получить информацию о процессах bash ps lax | grep bash:


0 R 1002 1178 1165 0 30 10 - 2317 - tty1 00:00:00 ps
daniilpi@ubunto:~$ ps lax | grep bash
4 1002 1067 724 25 5 8740 5548 do_wai SN tty1 0:00 -bash
1 1002 1088 1067 20 0 8740 1752 do_sig T tty1 0:00 -bash
1 1002 1115 1067 20 0 8740 1760 do_sig T tty1 0:00 -bash
0 1002 1165 1067 30 10 8664 5516 do_wai SN tty1 0:00 bash
0 1002 1178 1165 30 10 6476 2400 pipe_r SN+ tty1 0:00 grep --color=auto bash
daniilpi@ubunto:~$ _

Контрольные вопросы:

1. Перечислите состояния задачи в ОС Linux.
2. Как создаются задачи в ОС Linux?
3. Назовите классы потоков ОС Linux.
4. Как используется приоритет планирования при запуске задачи?
5. Объясните, что произойдет, если запустить программу в фоновом режиме без подавления потока вывода.
6. Объясните разницу между действием сочетаний клавиш Ctrl^Z и Ctrl^C.
7. Опишите, что значит каждое поле вывода команды jobs.
8. Назовите главное отличие утилиты top от ps.
9. В чем отличие результата выполнения команд top и htop?
10. Какую комбинацию клавиш нужно использовать для принудитель- ного завершения задания, запущенного в интерактивном режиме?
11. Какую комбинацию клавиш нужно использовать для приостановки задания, запущенного в интерактивном режиме?
12. Какая команда позволяет послать сигнал конкретному процессу?
13. Какая команда позволяет поменять поправку к приоритету уже запу- щенного процесса?
14. Какая команда позволяет запустить задание с пониженным приоритетом?
15. Какая команда позволяет запустить задание с защитой от прерывания при выходе из системы пользователя?
16. Какой процесс всегда присутствует в системе и является предком всех процессов?
17. Каким образом можно запустить задание в фоновом режиме?
18. Каким образом задание, запущенное в фоновом режиме, можно перевести в интерактивный режим?
19. Каким образом приостановленное задание можно перевести в интерактивный режим?

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

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

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

23. Поясните, от чего зависит максимальное значение PID.

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

1. Состояния задачи в ОС Linux:
    - Запущено (Running): Задача выполняется в данный момент.
    - Готово к выполнению (Ready): Задача готова к выполнению, но еще не запущена.
    - Ожидание (Waiting): Задача ожидает какое-то событие или ресурс, такое как ввод/вывод, блокировка и другие условия.
    - Остановлено (Stopped): Задача приостановлена и не выполняет действий.
    - Зомби (Zombie): Задача завершила выполнение, но ее статус остался до тех пор, пока родительский процесс не получит информацию о ее завершении.
2. Создание задач в ОС Linux:
    - Задачи можно создавать с помощью системных вызовов, таких как **fork()** и **exec()**.
    - Из командной строки можно создать задачу, добавив амперсанд **&** в конец команды для запуска в фоновом режиме.
3. Классы потоков ОС Linux:
    - Real-time (реального времени): Для приложений с жесткими временными ограничениями.
    - User-space (пользовательского пространства): Обычные приложения, работающие в пользовательском режиме.
4. Приоритет планирования используется при запуске задачи для определения того, какой задаче предоставляется больше процессорного

времени. Задачи с более высоким приоритетом получают больше времени на выполнение.

5. Если программа запущена в фоновом режиме без подавления потока вывода, то вывод будет направлен в терминал, который может быть заблокирован, пока задача не завершит свою работу. Терминал будет недоступен для ввода до тех пор, пока задача не завершится.
6. Разница между Ctrl^Z и Ctrl^C:
    - Ctrl^Z (SIGTSTP): Приостанавливает выполнение текущей задачи и отправляет ее в фоновый режим.
    - Ctrl^C (SIGINT): Прерывает выполнение текущей задачи, вынуждая ее завершиться.
7. Поля вывода команды **jobs**:
    - [номер] + или - : Статус задачи (запущена в переднем плане или фоне).
    - PID: Идентификатор процесса задачи.
    - Статус: Состояние задачи (например, Running, Stopped).
    - Команда: Команда, связанная с задачей.
8. Главное отличие утилиты **top** от **ps** заключается в том, что **top** предоставляет непрерывное отслеживание активности процессов в реальном времени, в то время как **ps** выводит снимок текущего состояния процессов.
9. **htop** является улучшенной версией **top** с более интерактивным интерфейсом и дополнительными функциями мониторинга.
10. Для принудительного завершения задачи, запущенной в интерактивном режиме, можно использовать комбинацию клавиш Ctrl^C (SIGINT).
11. Для приостановки задачи, запущенной в интерактивном режиме, можно использовать комбинацию клавиш Ctrl^Z (SIGTSTP).
12. Команда **kill** позволяет послать сигнал конкретному процессу.
13. Команда **renice** позволяет изменить приоритет уже запущенного процесса.
14. Для запуска задания с пониженным приоритетом можно использовать команду **nice**.

15. Для запуска задания с защитой от прерывания при выходе из системы пользователя можно использовать команду **nohup**.
16. Исключением является процесс с PID равным 1, который является инициализационным процессом (init или systemd) и предком всех процессов в системе.
17. Запуск задачи в фоновом режиме можно осуществить, добавив амперсанд **&** в конец команды при ее выполнении.
18. Задание, запущенное в фоновом режиме, можно перевести в интерактивный режим с помощью команды **fg**.
19. Приостановленное задание можно перевести в интерактивный режим с помощью команды **fg**.
20. Задание, выполняющееся в фоновом режиме, будет заблокировано при попытке обращения к терминалу и не сможет прочитать ввод с клавиатуры.
21. В Linux может быть открыто несколько терминалов. Для перемещения между ними можно использовать комбинации клавиш Ctrl+Alt+F1 до Ctrl+Alt+F6 для текстовых терминалов и Ctrl+Alt+F7 для графической среды.
22. Идентификатор PID (Process ID) - это уникальный номер процесса. PPID (Parent Process ID) - это идентификатор родительского процесса. PPID равен нулю в случае процесса init или systemd.
23. Максимальное значение PID зависит от настройки ядра и архитектуры системы. В 32-битных системах оно обычно равно 32767, в 64-битных системах - гораздо больше.
24. Если создается новый процесс, то его PID будет увеличен на 1 по сравнению с PID процесса, который был запущен ранее.