

**Липецкий государственный технический университет**

**Факультет автоматизации и информатики**

**Кафедра Автоматизированных систем управления**

**ЛАБОРАТОРНАЯ РАБОТА №3**

**по дисциплине «Операционная система Linux»**

**Процессы в операционной системе Linux**

Студент

Фетисов В. Д.

Группа ПИ-19

Руководитель

Кургасов В. В.

К.П.Н.

Липецк 2021

## Оглавление

Цель работы .....	3
Задание кафедры .....	4
Ход работы .....	6
Вывод .....	13
Контрольные вопросы .....	14

Цель работы

Ознакомиться на практике с понятием процесса в операционной системе.

Приобрести опыт и навыки управления процессами в операционной системе Linux.

Задание кафедры  
Индивидуальное задание

Вариант 11.

1. Вывести информацию о состоянии процессов системы в реальном режиме с сортировкой по убыванию значения приоритета. Отобразите информацию о состоянии процессов.
2. С помощью сигнала SIGKILL завершить все процессы, родителем которых является командный интерпретатор текущей сессии.
3. Вывести статистику использования памяти в байтах с обновлением каждые три секунды.
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 привести текстовую запись использованных команд. Кратко поясните результаты выполнения всех команд.

Задание к лабораторной

1. Запустить программу виртуализации Oracle VM VirtualBox.
2. Запустить виртуальную машину Ubuntu.
3. Открыть окно интерпретатора команд
4. Вывести общую информацию о системе
  - 4.1. Вывести информацию о текущем интерпретаторе команд
  - 4.2. Вывести информацию о текущем пользователе
  - 4.3. Вывести информацию о текущем каталоге
  - 4.4. Вывести информацию об оперативной памяти и области подкачки

- 4.5. Вывести информацию о дисковой памяти
- 5. Выполнить команды получения информации о процессах
  - 5.1. Получить идентификатор текущего процесса (PID)
  - 5.2. Получить идентификатор родительского процесса (PPID)
  - 5.3. Получить идентификатор процесса инициализации системы
  - 5.4. Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд
  - 5.5. Отобразить все процессы
- 6. Выполнить команды управления процессами
  - 6.1. Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе
  - 6.2. Определить текущее значение nice по умолчанию
  - 6.3. Запустить интерпретатор bash с понижением приоритета **nice -n 10 bash**
  - 6.4. Определить PID запущенного интерпретатора
  - 6.5. Установить приоритет запущенного интерпретатора равным 5 **renice -n 5 <PID процесса>**
  - 6.6. Получить информацию о процессах **bash ps lax | grep bash**

Ход работы

Индивидуальное задание.

Выведем информацию о состоянии процессов системы в реальном режиме

с сортировкой по убыванию значения приоритета. Отобразим

информацию о состоянии процессов. `ps -el --sort -nice | less`

`-el` – вывод расширенной информации

`-nice` – сортировка по убыванию приоритета

```
Server_Ubuntu (Процессы) [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
F S  UID      PID      PPID    C  PRI  NI ADDR SZ WCHAN  TTY      TIME CMD
1 S   0       24        2  0  99  19 -   0 -   ?    00:00:00 khugepaged
1 S   0       23        2  0  85   5 -   0 -   ?    00:00:00 ksm
4 S   0        1        0  1  80   0 - 25501 -   ?    00:00:01 systemd
1 S   0        2        0  0  80   0 -   0 -   ?    00:00:00 kthreadd
1 I   0        5        2  0  80   0 -   0 -   ?    00:00:00 kworker/0:0-events
1 I   0        7        2  0  80   0 -   0 -   ?    00:00:00 kworker/0:1-events
1 I   0        8        2  0  80   0 -   0 -   ?    00:00:00 kworker/u2:0-events_power_efficient
1 S   0       10        2  0  80   0 -   0 -   ?    00:00:00 ksoftirqd/0
1 I   0       11        2  0  80   0 -   0 -   ?    00:00:01 rcu_sched
1 S   0       12        2  0 -40   - -   0 -   ?    00:00:00 migration/0
5 S   0       13        2  0   9   - -   0 -   ?    00:00:00 idle_inject/0
1 S   0       14        2  0  80   0 -   0 -   ?    00:00:00 cpuhp/0
5 S   0       15        2  0  80   0 -   0 -   ?    00:00:00 kdevtmpfs
1 S   0       17        2  0  80   0 -   0 -   ?    00:00:00 rcu_tasks_kthre
1 S   0       18        2  0  80   0 -   0 -   ?    00:00:00 kauditd
1 S   0       19        2  0  80   0 -   0 -   ?    00:00:00 khungtaskd
1 S   0       20        2  0  80   0 -   0 -   ?    00:00:00 oom_reaper
1 S   0       22        2  0  80   0 -   0 -   ?    00:00:00 kcompactd0
1 S   0       78        2  0 -40   - -   0 -   ?    00:00:00 watchdogd
1 I   0       79        2  0  80   0 -   0 -   ?    00:00:00 kworker/u2:1-scsi_tmf_0
1 S   0       81        2  0  80   0 -   0 -   ?    00:00:00 kswapd0
1 S   0       82        2  0  80   0 -   0 -   ?    00:00:00 ecryptfs-kthrea
1 S   0       86        2  0  80   0 -   0 -   ?    00:00:00 scsi_eh_0
1 S   0       88        2  0  80   0 -   0 -   ?    00:00:00 scsi_eh_1
1 I   0       90        2  0  80   0 -   0 -   ?    00:00:00 kworker/u2:2-events_unbound
1 I   0       92        2  0  80   0 -   0 -   ?    00:00:00 kworker/u2:3-events_power_efficient
1 I   0      120        2  0  80   0 -   0 -   ?    00:00:00 kworker/0:2-events
1 S   0      160        2  0  80   0 -   0 -   ?    00:00:00 scsi_eh_2
1 I   0      161        2  0  80   0 -   0 -   ?    00:00:00 kworker/0:3-cgroup_destroy
1 S   0      199        2  0   9   - -   0 -   ?    00:00:00 irq/18-vmwgfx
1 S   0      287        2  0  80   0 -   0 -   ?    00:00:00 jbd2/dm-0-8
4 S   0      382        1  0  80   0 - 5346 -   ?    00:00:00 systemd-udevd
1 I   0      385        2  0  80   0 -   0 -   ?    00:00:00 kworker/0:4-events
vlad@vlad:~$ ps -el --sort -nice | less
```

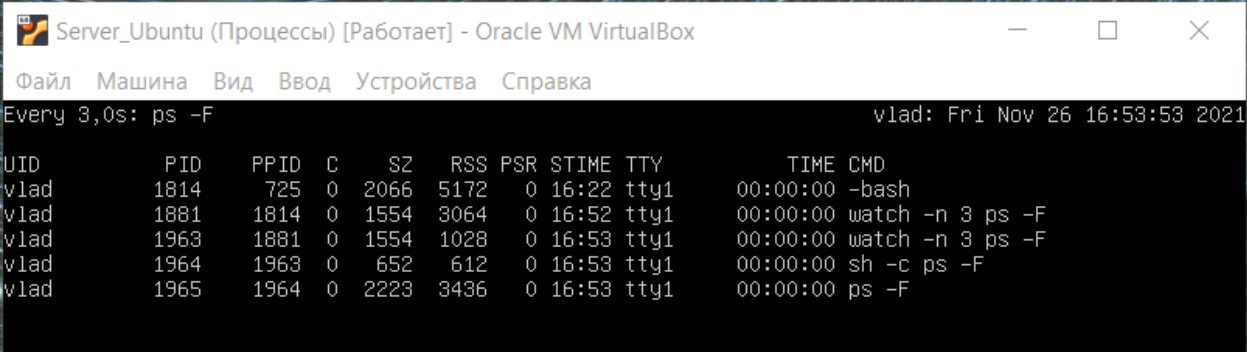
Рисунок 1 – Информация о состоянии процессов системы

Для начала узнаем PID процесса с помощью `ps -f`. В данном случае PID = 1814. Потом найдем процессы чей PPID = 1814, `ps -f --ppid 1814`. Далее завершим эти процессы командой `kill -9 1840 1841 1872`.

```
vlad@vlad:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
vlad         1814     725  0 16:22 tty1        00:00:00 -bash
vlad         1840     1814  0 16:26 tty1        00:00:00 sh loop
vlad         1841     1814  0 16:26 tty1        00:00:01 sh loop2
vlad         1871     1814  0 16:37 tty1        00:00:00 ps -f
vlad@vlad:~$ ps -f --ppid 1814
UID          PID    PPID  C STIME TTY          TIME CMD
vlad         1840     1814  0 16:26 tty1        00:00:00 sh loop
vlad         1841     1814  0 16:26 tty1        00:00:01 sh loop2
vlad         1872     1814  0 16:37 tty1        00:00:00 ps -f --ppid 1814
vlad@vlad:~$ kill -9 1840 1841 1872
-bash: kill: (1872) - No such process
[1]-  Killed                  sh loop
[2]+  Killed                  sh loop2
vlad@vlad:~$ ps -f --ppid 1814
UID          PID    PPID  C STIME TTY          TIME CMD
vlad         1873     1814  0 16:38 tty1        00:00:00 ps -f --ppid 1814
vlad@vlad:~$ _
```

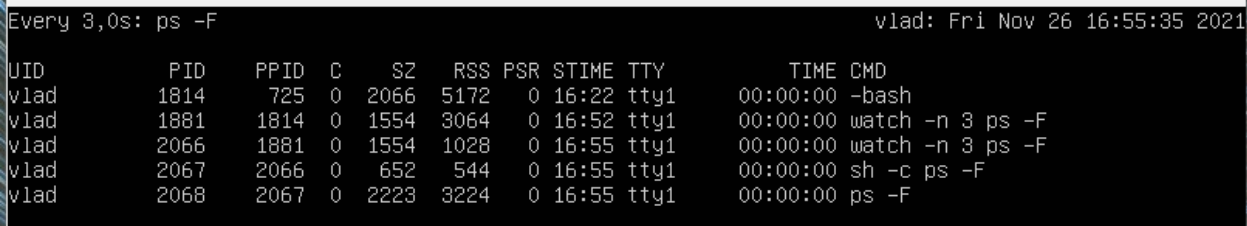
Рисунок 2 – С помощью сигнала SIGKILL завершили процессы

Выведем статистику использования памяти в байтах с обновлением каждые три секунды. `watch -n 3 ps -F`. `watch -3` чтобы команда повторялась по времени каждые 3 секунды. `ps -F` вывод статистики.



```
Server_Ubuntu (Процессы) [Работает] - Oracle VM VirtualBox
Файл  Машина Вид Ввод  Устройства Справка
Every 3,0s: ps -F                                vlad: Fri Nov 26 16:53:53 2021
UID          PID    PPID  C   S2  RSS PSR STIME TTY          TIME CMD
vlad         1814     725  0  2066 5172  0 16:22 tty1        00:00:00 -bash
vlad         1881     1814  0  1554 3064  0 16:52 tty1        00:00:00 watch -n 3 ps -F
vlad         1963     1881  0  1554 1028  0 16:53 tty1        00:00:00 watch -n 3 ps -F
vlad         1964     1963  0   652  612  0 16:53 tty1        00:00:00 sh -c ps -F
vlad         1965     1964  0  2223 3436  0 16:53 tty1        00:00:00 ps -F
```

Рисунок 3 – До “обновления”



```
Every 3,0s: ps -F                                vlad: Fri Nov 26 16:55:35 2021
UID          PID    PPID  C   S2  RSS PSR STIME TTY          TIME CMD
vlad         1814     725  0  2066 5172  0 16:22 tty1        00:00:00 -bash
vlad         1881     1814  0  1554 3064  0 16:52 tty1        00:00:00 watch -n 3 ps -F
vlad         2066     1881  0  1554 1028  0 16:55 tty1        00:00:00 watch -n 3 ps -F
vlad         2067     2066  0   652  544  0 16:55 tty1        00:00:00 sh -c ps -F
vlad         2068     2067  0  2223 3224  0 16:55 tty1        00:00:00 ps -F
```

Рисунок 4 – После “обновления”

## Задание к лабораторной

Запустили программу виртуализации Oracle VM VirtualBox и запустим виртуальную машину Ubuntu, откроем окно интерпретатора команд и выведем общую информацию о системе командой `echo $SHELL`.

```
vlad@vlad:~$ echo $SHELL
/bin/bash
vlad@vlad:~$
```

### Рисунок 5 – Информация о текущем интерпретаторе команд

Выведем информацию о текущем пользователе с помощью команды `whoami`.

```
vlad@vlad:~$ whoami
vlad
```

### Рисунок 6 – Информация о текущем пользователе

Выведем информацию о текущем каталоге с помощью команды `pwd`.

```
vlad@vlad:~$ pwd
/home/vlad
```

### Рисунок 7 – Информация о текущем каталоге

Выведем информацию об оперативной памяти и области подкачки `free`.

```
vlad@vlad:~$ free
              total        used        free      shared  buff/cache   available
Mem:           2035456       196140       1244148          1152        595168       1683592
Swap:          1776636           0        1776636
```

### Рисунок 8 – Информация об оперативной памяти и области подкачки

Выведем информацию о дисковой памяти.



```

vlad@vlad:~$ df
Filesystem            1K-blocks    Used Available Use% Mounted on
udev                  972560         0   972560    0% /dev
tmpfs                 203548      1132   202416    1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 9219412 5746876 2984500   66% /
tmpfs                1017728         0   1017728    0% /dev/shm
tmpfs                  5120         0     5120    0% /run/lock
tmpfs                1017728         0   1017728    0% /sys/fs/cgroup
/dev/loop2             63360     63360         0 100% /snap/core20/1242
/dev/sda2             999320     306212   624296   33% /boot
/dev/loop3             63360     63360         0 100% /snap/core20/1169
/dev/loop4             56832     56832         0 100% /snap/core18/2253
/dev/loop5             56832     56832         0 100% /snap/core18/2128
/dev/loop1            101888    101888         0 100% /snap/core/11993
/dev/loop6             68864     68864         0 100% /snap/lxd/21545
/dev/loop0            101888    101888         0 100% /snap/core/11798
/dev/loop7            119424    119424         0 100% /snap/docker/1125
/dev/loop8             27136     27136         0 100% /snap/heroku/4078
/dev/loop9             43264     43264         0 100% /snap/snapd/14066
/dev/loop10            68864     68864         0 100% /snap/lxd/21835
/dev/loop11            26880     26880         0 100% /snap/heroku/4076
/dev/loop12            33280     33280         0 100% /snap/snapd/13640
tmpfs                 203544         0   203544    0% /run/user/1000
vlad@vlad:~$ _

```

Рисунок 9 – Информация о дисковой памяти

Введем команды для получения информации о процессах.

Получим информацию о текущем процессе (PID) с помощью команды `echo $$`.

```

vlad@vlad:~$ echo $$
1312

```

Рисунок 10 – информация о текущем процессе

Получим идентификатор родительского процесса (PPID) с помощью команды.

```

vlad@vlad:~$ echo $PPID
730

```

Рисунок 11 – информация о PPID

Получим идентификатор процесса инициализации системы с помощью команды `pidof bash`.

```

vlad@vlad:~$ pidof bash
1312

```

Рисунок 12 – Информация о текущем процессе

Получим информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд с помощью команды ps.

```
vlad@vlad:~$ ps
  PID TTY          TIME CMD
 1312 tty1      00:00:00 bash
 1368 tty1      00:00:00 ps
```

Рисунок 13 – Информация о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

Отобразим информацию о всех процессах с помощью команды ps -e.

```
  PID TTY          TIME CMD
    1 ?           00:00:00 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
    9 ?           00:00:00 mm_percpu_wq
   10 ?           00:00:00 ksoftirqd/0
   11 ?           00:00:02 rcu_sched
   12 ?           00:00:00 migration/0
   13 ?           00:00:00 idle_inject/0
   14 ?           00:00:00 cpuhp/0
   15 ?           00:00:00 kdevtmpfs
   16 ?           00:00:00 netns
   17 ?           00:00:00 rcu_tasks_kthre
   18 ?           00:00:00 kauditd
   19 ?           00:00:00 khungtaskd
   20 ?           00:00:00 oom_reaper
   21 ?           00:00:00 writeback
   22 ?           00:00:00 kcompactd0
   23 ?           00:00:00 ksmd
   24 ?           00:00:00 khugepaged
   70 ?           00:00:00 kintegrityd
   71 ?           00:00:00 kblockd
   72 ?           00:00:00 blkcg_punt_bio
   73 ?           00:00:00 tpm_dev_wq
   74 ?           00:00:00 ata_sff
   75 ?           00:00:00 md
   76 ?           00:00:00 edac-poller
   77 ?           00:00:00 devfreq_wq
   78 ?           00:00:00 watchdogd
   81 ?           00:00:00 kswapd0
   82 ?           00:00:00 ecryptfs-kthrea
   84 ?           00:00:00 kthrotld
   85 ?           00:00:00 acpi_thermal_pm
   86 ?           00:00:00 scsi_eh_0
```

Рисунок 14 – Информация о всех процессах

Выполним команды для управления процессами.

Получим информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе с помощью команды ps.

```

vlad@vlad:~$ ps
  PID TTY          TIME CMD
 1312 tty1      00:00:00 bash
 1396 tty1      00:00:00 ps

```

Рисунок 15 – Информация о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

Определим текущее значение nice по умолчанию с помощью команды nice.

```

1396 tty1      00:00:00 ps
vlad@vlad:~$ nice
0

```

Рисунок 16 – Информация о текущем значении nice

Запустим интерпретатор bash с понижением приоритета с помощью команды nice -n 10 bash.

```

vlad@vlad:~$ nice -n 10 bash
vlad@vlad:~$ nice
10

```

Рисунок 17 – Запустили bash с понижением приоритета

Определим PID запущенного интерпретатора с помощью команды echo \$\$.

```

vlad@vlad:~$ echo $$
1404

```

Рисунок 18 – PID запущенного интерпретатора

Установим приоритет запущенного интерпретатора равным 5 с помощью команды renice -n 5 <PID процесса>.

```

vlad@vlad:~$ renice -n 5 1404
renice: failed to set priority for 1404 (process ID): Permission denied
vlad@vlad:~$ sudo renice -n 5 1404
[sudo] password for vlad:
^[[A^[[A^[[ASorry, try again.
[sudo] password for vlad:
Sorry, try again.
[sudo] password for vlad:
1404 (process ID) old priority 10, new priority 5

```

Рисунок 19 – Изменили приоритет запущенного интерпретатора

Получим информацию о процессах bash с помощью команды `ps lax | grep bash`.

```
vlad@vlad:~$ ps lax | grep bash
4 1000 1312 730 20 0 8264 5092 do_wai S tty1 0:00 -bash
0 1000 1404 1312 25 5 8272 5088 do_wai SN tty1 0:00 bash
0 1000 1438 1404 25 5 6300 664 - RN+ tty1 0:00 grep --color=auto bash
vlad@vlad:~$ _
```

Рисунок 20 – Информация о процессах bash

## Вывод

В ходе лабораторной работы я ознакомился со средами управления процессами ОС Ubuntu.

## Контрольные вопросы

1. Перечислите состояния задачи в ОС Ubuntu.

Задача переходит в состояние `running` (выполнения) после выделения ей процессора. При блокировке задача переходит в состояние `sleeping` (спячки), а при остановке работы в состояние `stopped` (останов). Состояние `zombie` (зомби) показывает, что выполнение задачи прекратилось, однако она еще не была удалена из системы. Например, если процесс состоит из нескольких потоков, он будет пребывать в состоянии зомби, пока все потоки не получат уведомление о завершении работы основного процесса. Задача в состоянии `dead` (смерти) может быть удалена из системы. Состояния `active` (активный) и `expired` (неактивный) используются при планировании выполнения процесса, и поэтому они не сохраняются в переменной `state`.

2. Как создаются задачи в ОС Ubuntu?

Задачи создаются путем вызова функции `clone`.

3. Назовите классы потоков ОС Ubuntu

1. Потоки реального времени, обслуживаемые по алгоритму FIFO.

2. Потоки реального времени, обслуживаемые в порядке циклической очереди.

3. Потоки разделения времени

4. Как используется приоритет планирования при запуске задачи

Для Linux приоритет планирования процесса реального времени варьируется от 1 (самый низкий приоритет) до 99 (самый высокий приоритет). Обычные процессы также имеют приоритет планирования, равный 0.

## 5. Как можно изменить приоритет для выполняющейся задачи?

С помощью `renice`

`renice` — UNIX-утилита, позволяющая изменить приоритет запущенных задач. Привилегированный пользователь (`root`) может указать отрицательное смещение. Команда `renice` может смещать приоритет в диапазоне от -20 (наивысший приоритет) до 19 (низший приоритет) от текущего. Для изменения значения приоритета отдельных процессов достаточно перечислить их идентификаторы.