

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Отчет по лабораторной работе № 4

«Управление процессами ОС Ubuntu»

по курсу «ОС Linux»

Студент
Группа ПМ-18

Полухина Е.Д.

Руководитель

Кургасов В.В.

Липецк 2020 г.

СОДЕРЖАНИЕ

Цель работы	4
Задание.....	5
Ход работы	6
1. Запуск программы виртуализации Oracle VM VirtualBox.	6
2. Запуск виртуальной машины Ubuntu.	7
3. Открытие окна интерпретатора команд.	8
4. Вывод информации о текущем интерпретаторе команд.	9
5. Вывод информации о текущем пользователе.	10
6. Вывод информации о текущем каталоге.	11
7. Вывод информации об оперативной памяти и области подкачки.	12
8. Вывод информации о дисковой памяти.	13
9. Получение идентификатора текущего процесса.	14
10. Получение идентификатора родительского процесса.	15
11. Получение идентификатора процесса инициализации системы.	16
12. Получение информации о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд.	17
13. Отображение всех процессов.	18
14. Определение текущего значения <code>nice</code> по умолчанию.	19
15. Запуск интерпретатора <code>bash</code> с понижением приоритета.	20
16. Определение PID запущенного интерпретатора.	21
17. Установление приоритета запущенного интерпретатора равным 5.	22
18. Получение информации о процессах <code>bash</code>	23
Ответы на контрольные вопросы	24

Вывод	26
-------------	----

Цель работы

Целью работы является знакомство со средствами управления процессами ОС Ubuntu.

Задание

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 aux | grep bash`

Ход работы

1. Запуск программы виртуализации Oracle VM VirtualBox.

На рисунке 1 представлено окно программы Oracle VM VirtualBox после запуска программы.

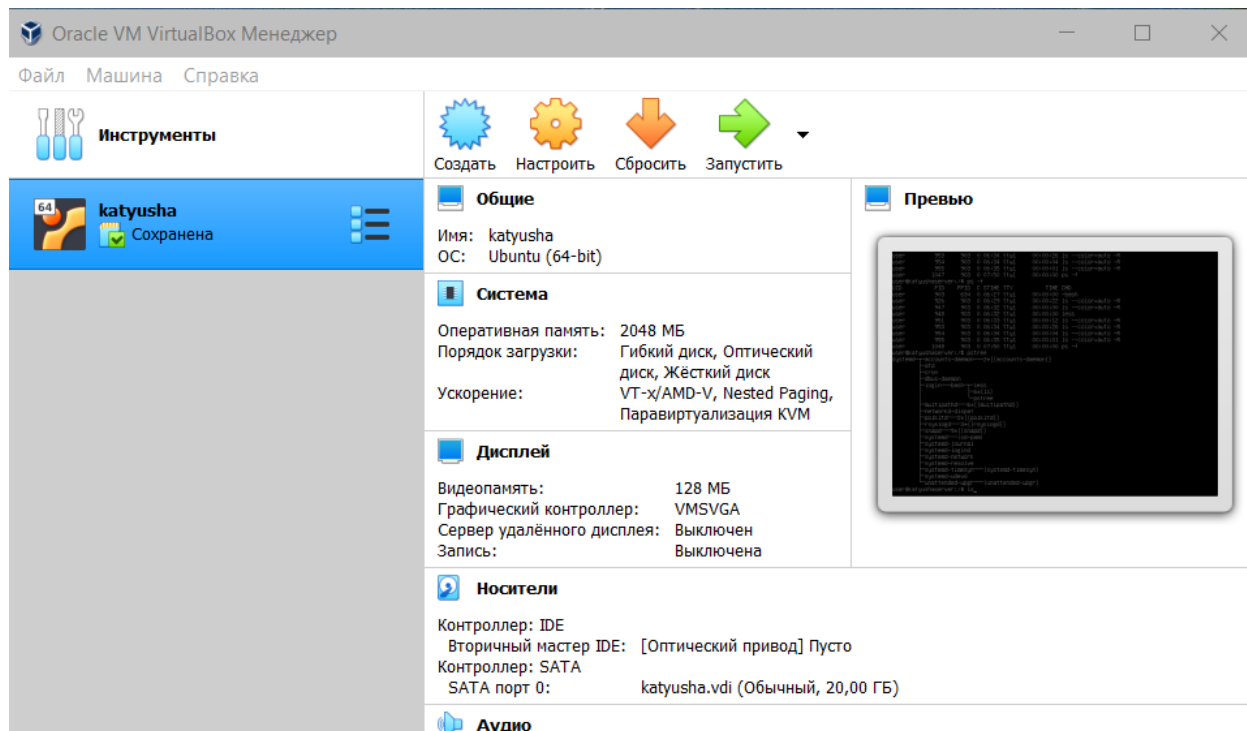


Рисунок 1 – Oracle VM VirtualBox

2. Запуск виртуальной машины Ubuntu.

На рисунке 2 показан запуск виртуальной машины с Linux Ubuntu.

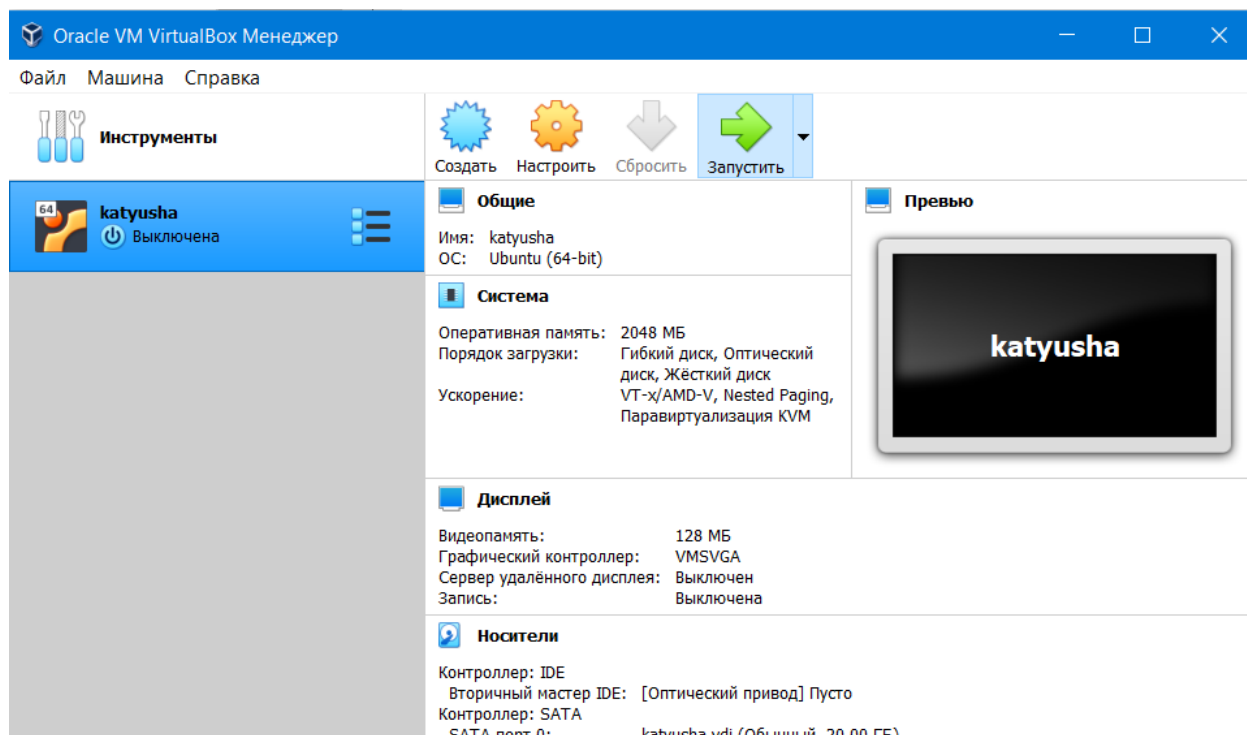
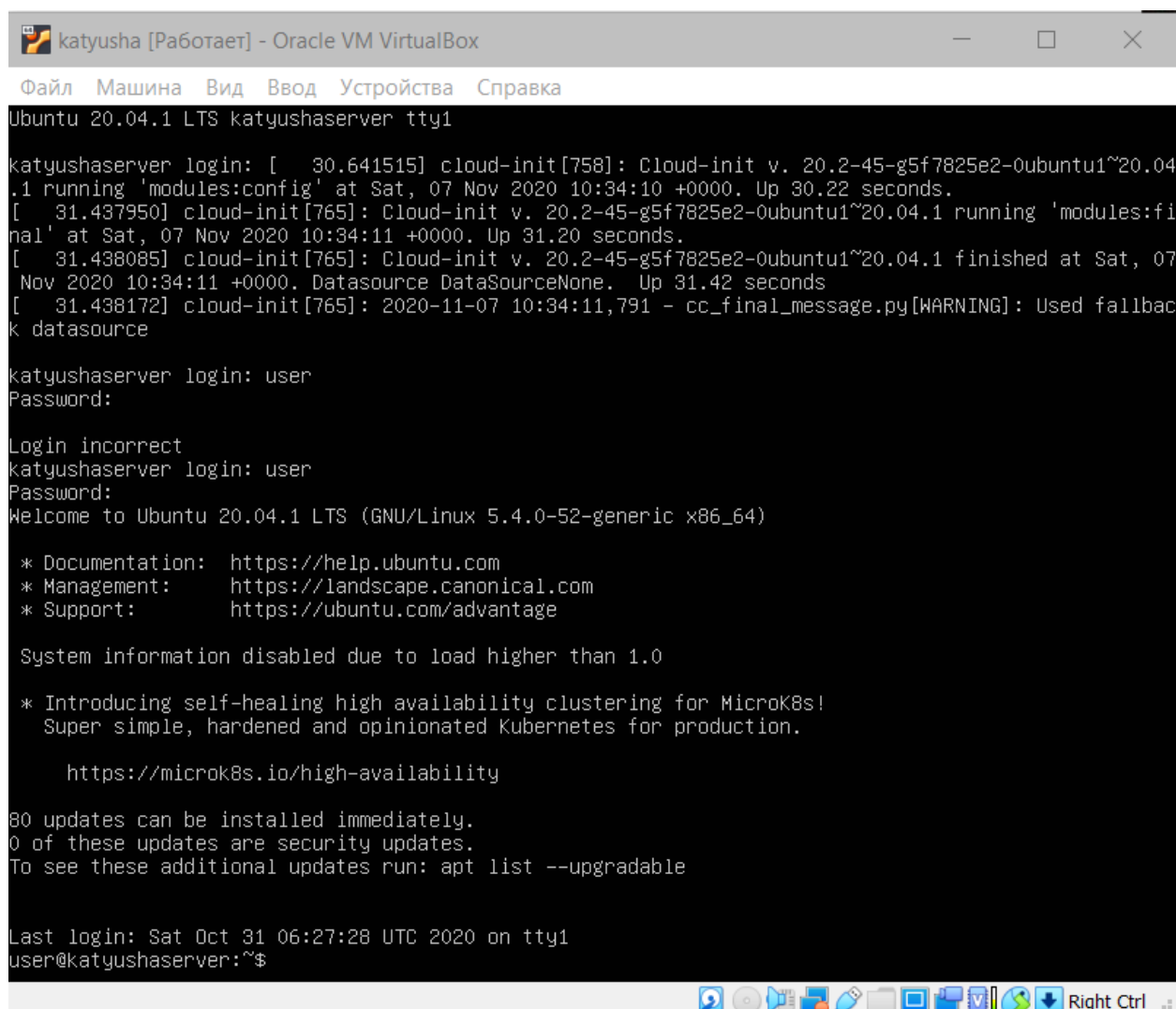


Рисунок 2 – Запуск виртуальной машины

3. Открытие окна интерпретатора команд.

На рисунке 3 представлено окно интерпретатора команд после запуска виртуальной машины с Linux Ubuntu и ввода логина и пароля



```
katyusha [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Ubuntu 20.04.1 LTS katyushaserver tty1

katyushaserver login: [ 30.641515] cloud-init[758]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1~20.04
.1 running 'modules:config' at Sat, 07 Nov 2020 10:34:10 +0000. Up 30.22 seconds.
[ 31.437950] cloud-init[765]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1~20.04.1 running 'modules:fi
nal' at Sat, 07 Nov 2020 10:34:11 +0000. Up 31.20 seconds.
[ 31.438085] cloud-init[765]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1~20.04.1 finished at Sat, 07
Nov 2020 10:34:11 +0000. Datasource DataSourceNone. Up 31.42 seconds
[ 31.438172] cloud-init[765]: 2020-11-07 10:34:11,791 - cc_final_message.py[WARNING]: Used fallback
k datasource

katyushaserver login: user
Password:

Login incorrect
katyushaserver login: user
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

 * Introducing self-healing high availability clustering for MicroK8s!
   Super simple, hardened and opinionated Kubernetes for production.

   https://microk8s.io/high-availability

80 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Sat Oct 31 06:27:28 UTC 2020 on tty1
user@katyushaserver:~$
```

Рисунок 3 – Терминал после ввода логина и пароля

4. Вывод информации о текущем интерпретаторе команд.

На рисунке 4 показано получение информации о текущем интерпретаторе с помощью команды `echo $SHELL`.



```
user@katyushaserver:~$ echo $SHELL
/bin/bash
user@katyushaserver:~$ _
```

The screenshot shows a terminal window with a black background. The prompt is 'user@katyushaserver:~\$'. The command 'echo \$SHELL' is entered, and the output is '/bin/bash'. The prompt changes to 'user@katyushaserver:~\$ _' after the command is executed. The terminal window has a taskbar at the bottom with various icons and the text 'Right Ctrl'.

Рисунок 4 – получение информации о текущем интерпретаторе

Переменная окружения `SHELL` хранит путь до исполняемого файла оболочки. Из вывода команды, мы видим, что используется оболочка `bash`.

5. Вывод информации о текущем пользователе.

Выведена информация о текущем пользователе с помощью команды `whoami`. Эта команда нужна, чтобы отобразить имя пользователя, который в данный момент вошел в систему (рис. 5).

A screenshot of a terminal window. The prompt is 'user@katyushaserver:~\$'. The command 'whoami' has been entered, and the output 'user' is displayed on the next line. The prompt 'user@katyushaserver:~\$' appears again. The terminal window has a black background and a light gray title bar. The taskbar at the bottom shows various icons including a globe, a CD, a folder, a USB drive, a printer, a network icon, and a 'Right Ctrl' button.

```
user@katyushaserver:~$ whoami
user
user@katyushaserver:~$
```

Рисунок 5 – Получение информации о текущем пользователе

6. Вывод информации о текущем каталоге.

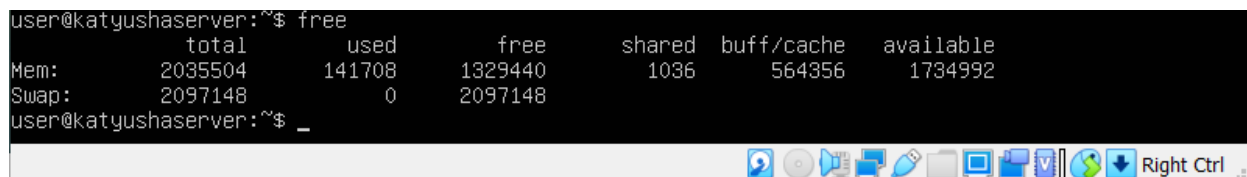
На рисунке 6 показано получение информации о текущем каталоге с использованием команды `pwd`. Эта команда выводит полный путь до текущей рабочей директории, в которой находится пользователь. Как видно, мы находимся в домашнем каталоге.

A screenshot of a terminal window. The prompt is 'user@katyushaserver:~\$'. The command 'pwd' has been entered, and the output is '/home/user'. The prompt is now 'user@katyushaserver:~\$ _'. The terminal window has a black background and a light gray title bar. The system tray at the bottom shows various icons including a network icon, a volume icon, a power icon, and a 'Right Ctrl' button.

Рисунок 6 – Получение информации о текущем каталоге

7. Вывод информации об оперативной памяти и области подкачки.

На рисунке 7 показано получение информации об оперативной памяти и файле подкачки с помощью команды `free`. Будучи запущенной без ключей, она отобразит статистику в кибибайтах (2 в степени $10 = 1024$).



```
user@katyushaserver:~$ free
              total        used         free       shared    buff/cache   available
Mem:           2035504       141708       1329440         1036        564356       1734992
Swap:          2097148           0         2097148
user@katyushaserver:~$ _
```

Рисунок 7 – Список запущенных процессов

Вывод содержит данные о физической памяти `Mem` и файле подкачки `Swap`. В операционной системе Linux, как и в других ОС, файл подкачки нужен для страховки оперативной памяти. Когда установленный объем ОЗУ заканчивается, используется именно выделенная область из файла подкачки.

В столбцах указаны следующие параметры:

`Total` – эта цифра представляет всю существующую память.

`Used` – вычисление общего значения оперативной памяти системы за вычетом выделенной свободной, разделяемой, буферной и кэш-памяти.

`Free` – это память, которая не используется ни для каких целей.

`Shared`, `Buffer`, и `Cache` – идентифицируют память, используемую для нужд ядра или операционной системы. Буфер и кэш складываются вместе, а сумма указывается в разделе «`buff/cache`».

`Available` – память появляется в более новых версиях `free` и предназначена для того, чтобы дать конечному пользователю оценку того, сколько ресурсов памяти все еще открыто для использования.

8. Вывод информации о дисковой памяти.

Получить информацию о дисковой памяти можно с помощью команды `df -h`. Параметр `-h` выведет данные в более читаемом формате – в мегабайтах и гигабайтах (рис. 8).

```
user@katyushaserver:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            952M   0    952M   0% /dev
tmpfs           199M  1.1M   198M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 19G   5.3G   13G  30% /
tmpfs           994M   0    994M   0% /dev/shm
tmpfs           5.0M   0     5.0M   0% /run/lock
tmpfs           994M   0    994M   0% /sys/fs/cgroup
/dev/loop0       56M   56M     0 100% /snap/core18/1932
/dev/sda2        976M  197M   713M  22% /boot
/dev/loop2       56M   56M     0 100% /snap/core18/1885
/dev/loop3       71M   71M     0 100% /snap/lxd/16922
/dev/loop4       31M   31M     0 100% /snap/snapd/9607
/dev/loop5       31M   31M     0 100% /snap/snapd/9721
tmpfs           199M   0    199M   0% /run/user/1000
/dev/loop6       68M   68M     0 100% /snap/lxd/18150
user@katyushaserver:~$
```

Рисунок 8 – Вывод информации о дисковой памяти

Filesystem – файловая система.

Size – размер в мегабайтах, показывается вся емкость точки монтирования.

Used – количество используемого дискового пространства.

Available – количество свободного пространства в мегабайтах.

Use% – процент использования файловой системы.

Mounted on – точка монтирования, где установлена файловая система.

Например, мы видим, что в каталоге `/boot` уже занято 22% места.

9. Получение идентификатора текущего процесса.

На рисунке 9 показано получение идентификатора текущего процесса (PID) с использованием команды `echo $$`. `$$` – это идентификатор используемого в данный момент процесса командной оболочки. Итак, в нашем случае число 904 является PID экземпляра `bash`.

A screenshot of a terminal window with a black background and white text. The prompt is 'user@katyushaserver:~\$'. The user has entered the command 'echo \$\$'. The output of the command is '904'. Below the output, the prompt is again 'user@katyushaserver:~\$'. At the bottom of the terminal window, there is a taskbar with various icons including a globe, a folder, a document, and a network icon. The text 'Right Ctrl' is visible on the right side of the taskbar.

Рисунок 9 – Получение PID текущего процесса

10. Получение идентификатора родительского процесса.

На рисунке 10 показано получение идентификатора родительского процесса (PPID) с помощью команды `echo $PPID`. Переменная `$PPID` – идентификатор соответствующего родительского процесса. В нашем случае число 624 является PPID родительского процесса экземпляра `bash`.



11. Получение идентификатора процесса инициализации системы.

Получаем идентификатора процесса по имени `init` с помощью команды `pidof init`. `Init` - система инициализации в Unix-подобных системах, которая запускает все остальные процессы. Первый пользовательский процесс работает как демон и обычно имеет PID 1.

Мы видим, что PID процесса инициализации системы равен 1 (рис. 11).

```
user@katyushaserver:~$ pidof init
1
user@katyushaserver:~$
```

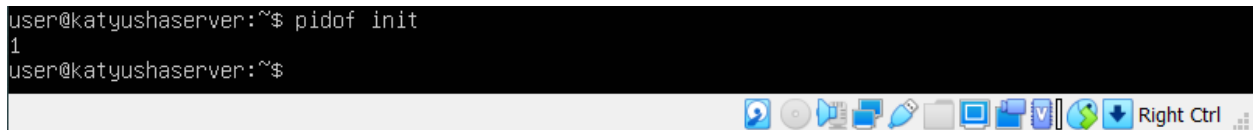


Рисунок 11 – Получение PID процесса инициализации системы

12. Получение информации о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд.

На рисунке 12 показано получение информации о выполняющихся процессах с помощью команды «ps T -fu user». Параметр «T» позволяет увидеть только процессы, связанные с этим терминалом, а параметр «-u <user>» нужен, чтобы ограничить список только процессами, действительно запущенными user.

```
user@katyushaserver:~$ ps T -fu user
UID      PID     PPID  C  STIME TTY      STAT   TIME CMD
root      624       1  0  10:34 tty1    Ss      0:00 /bin/login -p --
user      889       1  0  10:34 ?        Ss      0:00 /lib/systemd/systemd --user
user      896     889  0  10:34 ?        S       0:00 (sd-pam)
user      904     624  0  10:34 tty1    S       0:00 -bash
user     1397     904  0  12:20 tty1    R+      0:00 ps T -fu user
user@katyushaserver:~$
```

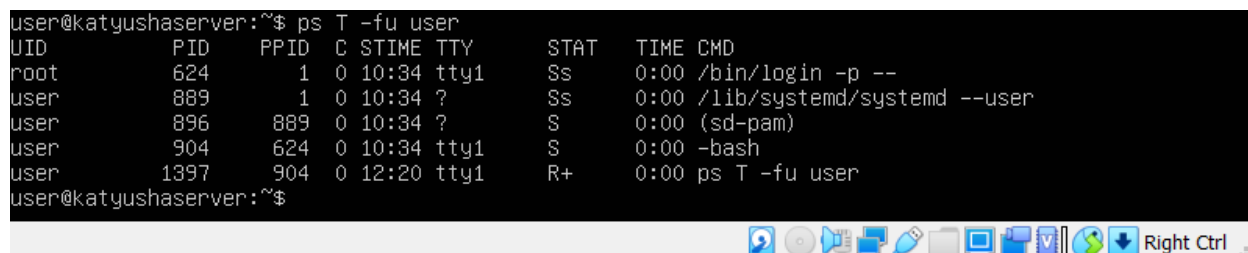


Рисунок 12 – Получение информации о процессах

13. Отображение всех процессов.

Для отображения всех процессов используем команду `ps -e`, где параметр `-e` нужен, чтобы просмотреть все запущенные процессы (рис. 13).



Рисунок 13 – Ввод команды `ps -e`

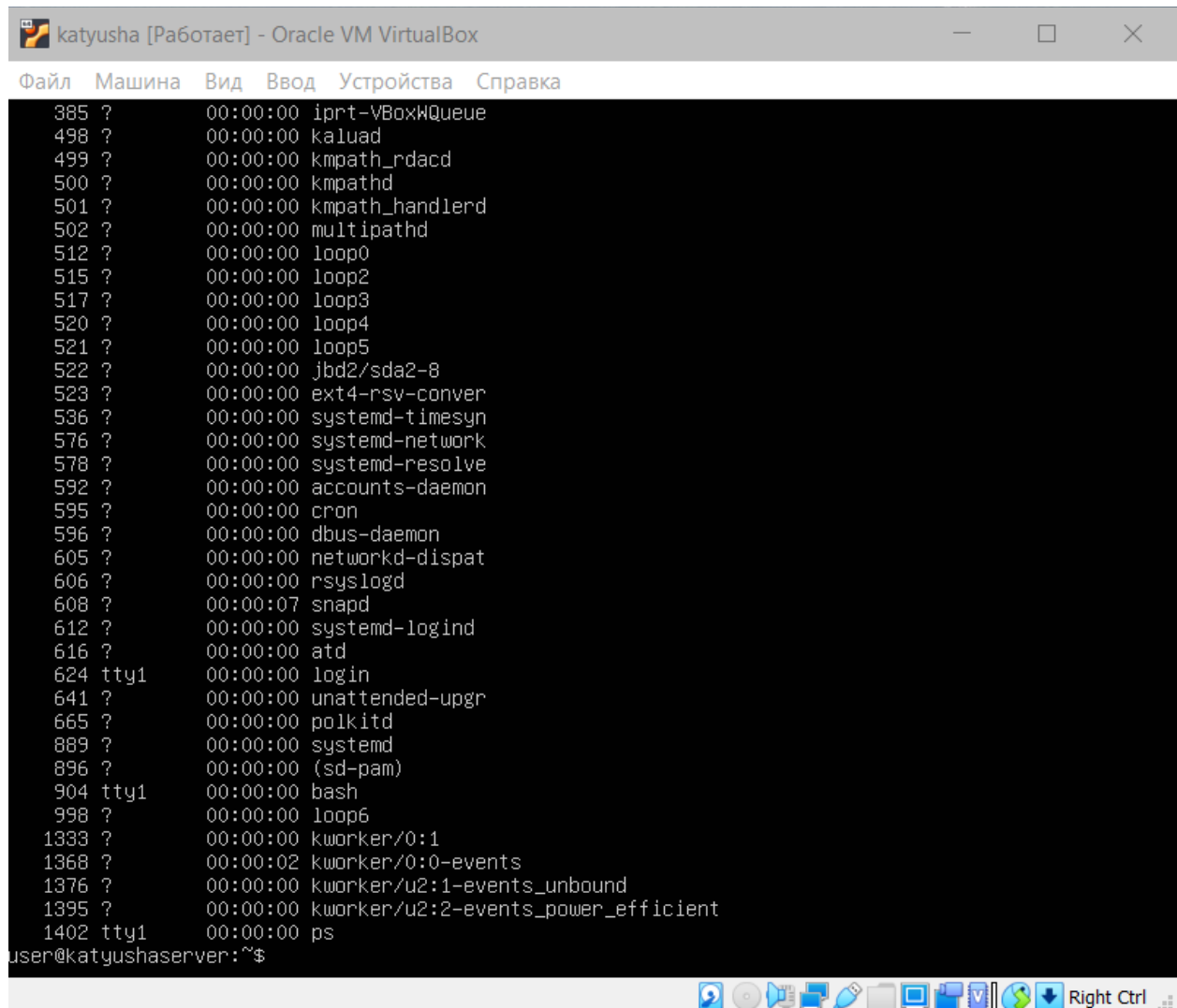


Рисунок 14 – Просмотр всех запущенных процессов

14. Определение текущего значения nice по умолчанию.

Определить текущее значение nice по умолчанию можно с помощью команды «nice» (рис. 15).


A terminal window with a black background. The prompt is 'user@katyushaserver:~\$'. The command 'nice' has been entered, and the output '0' is displayed on the next line. The prompt is now 'user@katyushaserver:~\$ _'. The terminal window is part of a desktop environment with a taskbar at the bottom showing various icons and the text 'Right Ctrl'.

Рисунок 15 – Значение nice

В нашем случае значение nice по умолчанию оказалось равным 0.

Во время создания каждой задаче присваивается статический приоритет (static priority), называемый также правильным значением (nice value). При обычном запуске команд или программ принимается равным приоритету родительского процесса.

Значение nice находится в диапазоне от -20 до 19. Большее значение означает меньший приоритет.

15. Запуск интерпретатора bash с понижением приоритета.

Используя команду `ps -l`, смотрим информацию о запущенных процессах и видим, что значение приоритета bash равно 0.

С помощью команды «`nice -n 10 bash`» запускаем интерпретатор bash с понижением приоритета на 10.

Опять смотрим информацию о запущенных процессах и видим, что действительно запустили bash с понижением приоритета на 10 (рис. 16).

```
user@katyushaserver:~$ ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000      904       624  0  80   0 - 1801 do_wai tty1      00:00:00 bash
0 R   1000     1405       904  0  80   0 - 1888 -      tty1      00:00:00 ps
user@katyushaserver:~$ nice -n 10 bash
user@katyushaserver:~$ ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000      904       624  0  80   0 - 1801 do_wai tty1      00:00:00 bash
0 S   1000     1406       904  1  90  10 - 1759 do_wai tty1      00:00:00 bash
0 R   1000     1412     1406  0  90  10 - 1888 -      tty1      00:00:00 ps
user@katyushaserver:~$
```

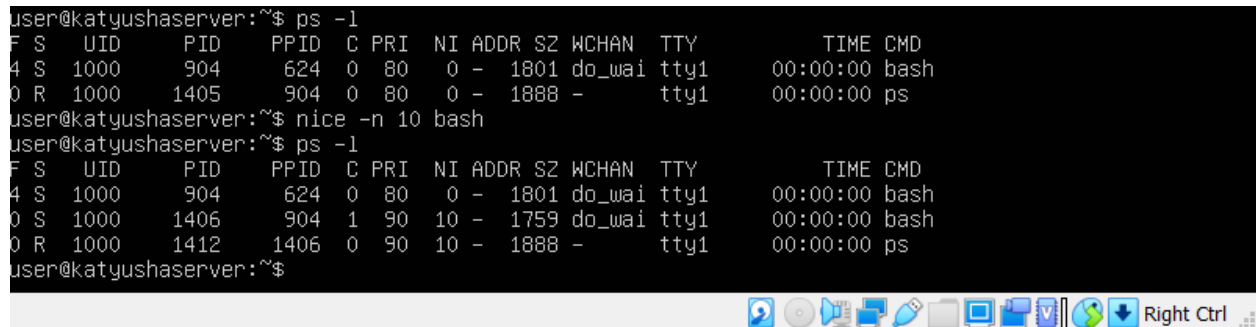


Рисунок 16 – Запуск bash с понижением приоритета

NI – значение `nice`, которое находится в диапазоне от -20 до 19. Большее значение означает меньший приоритет.

16. Определение PID запущенного интерпретатора.

На рисунке 17 показано определение PID запущенного интерпретатора с помощью команды `pidof bash`. У нас PID bash оказался равным 904. Также это можно сделать с помощью команды `ps -f`.

```
user@katyushaserver:~$ pidof bash
904
user@katyushaserver:~$ ps -f
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
user	904	624	0	10:34	tty1	00:00:00	-bash
user	1423	904	0	13:01	tty1	00:00:00	ps -f

```
user@katyushaserver:~$
```



Рисунок 17 – Определение PID запущенного интерпретатора.

17. Установление приоритета запущенного интерпретатора равным 5.

Чтобы установить приоритет запущенного интерпретатора равным 5 используем команду «renice -n 5 <PID процесса>». Команда renice позволяет изменять приоритет уже выполняемого процесса (рис. 18).

```
user@katyushaserver:~$ ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000      904      624  0  80   0 -  1801 do_wai  tty1        00:00:00 bash
0 R   1000     1424      904  0  80   0 -  1888 -      tty1        00:00:00 ps
user@katyushaserver:~$ renice -n 5 904
904 (process ID) old priority 0, new priority 5
user@katyushaserver:~$ ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000      904      624  0  85   5 -  1801 do_wai  tty1        00:00:00 bash
0 R   1000     1426      904  0  85   5 -  1888 -      tty1        00:00:00 ps
user@katyushaserver:~$ _
```

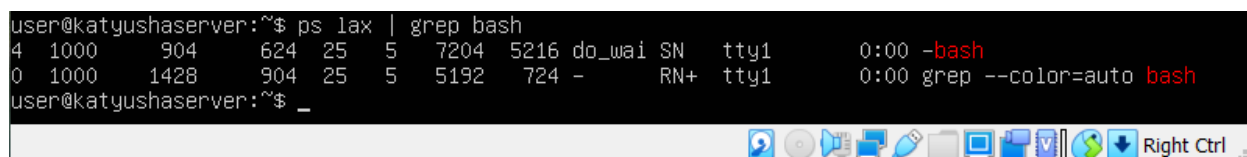


Рисунок 18 – Установление приоритета процесса

Используя команду ps -l смотрим информацию о запущенных процессах и видим, что приоритет действительно стал равным 5.

18. Получение информации о процессах bash.

На рисунке 19 показано получение информации о процессах bash с использованием команды «ps lax | grep bash».



```
user@katyushaserver:~$ ps lax | grep bash
4 1000    904    624 25   5   7204  5216 do_wai SN   tty1      0:00 -bash
0 1000    1428    904 25   5   5192   724  -    RN+  tty1      0:00 grep --color=auto bash
user@katyushaserver:~$ _
```

Рисунок 19 – Получение информации о процессах bash

Ответы на контрольные вопросы

1.Перечислите состояния задачи в ОС Ubuntu.

Running (выполнение) – переходит в это состояние после выделения ей процессора.

Sleeping (спячка) – переходит в это состояние при блокировке задачи.

Stopped (останов) – переходит в это состояние после остановки работы.

Zombie (зомби) – данное состояние показывает, что выполнение задачи прекратилось, но ещё не удалена из системы.

Dead (смерть) – Задача в этом состоянии, может быть удалена из системы

Active (активный) и expired (неактивный) используются при планировании выполнения процесса, поэтому они не сохраняются в переменной state.

2.Как создаются задачи задачи в ОС Ubuntu?

Задачи создаются путем вызова системной функции clone. Любые обращения к fork или vfork преобразуются в системные вызовы clone во время компиляции. Функция fork создает дочернюю задачу, виртуальная память для которой выделяется по принципу копирования при записи (copy-on-write). Когда дочерний или же родительский процесс пытается выполнить запись в страницу памяти, записывающая программа создает собственную копию страницы в памяти.

3.Назовите классы потоков ОС Ubuntu.

В операционной системе Linux алгоритмом диспетчеризации различаются три класса потоков:

- 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.

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?

Команда `renice` служит для изменения значения `nice` для уже выполняющихся процессов. Ее формат таков:

`renice priority [[-p] PID] [[-g] grp] [[-u] user]`

Вывод

В результате выполнения лабораторной работы я получила знания по работе с процессами в ОС Linux Ubuntu. Научилась выводить информацию о текущем пользователе, каталоге, об оперативной памяти и области подкачки. Также выполнила команды управления процессами и получила информацию о выполняющихся процессах текущего пользователя.