

**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Отчет по лабораторной работе № 3

«Управление процессами в Linux»

по курсу «ОС Linux»

Студент  
Группа ПМ-18

Полухина Е.Д.

Руководитель

Кургасов В.В.

Липецк 2020 г.

## СОДЕРЖАНИЕ

Цель работы .....	3
Задание .....	4
Ход работы .....	5
1. Команда cat. ....	5
2. Команда head. ....	6
3. Команда more.....	7
4. Команда tail.....	9
5. Команда less.....	10
6. Команда grep.....	11
7. Команда find. ....	12
8. Конвейер, перенаправление ввода-вывода. ....	14
9. Команды chmod, chown. ....	16
10. Процессы.....	18
11. Supervisor. ....	27
12. Cron. ....	30
Вывод .....	31

## Цель работы

Ознакомиться на практике с понятием процесса в операционной системе. Научиться пользоваться перенаправлением ввода-вывода, «Supervisor», планировщиком задач. Приобрести опыт и навыки управления процессами в операционной системе Linux.

### Задание

1. Повторить команды `cat`, `head`, `tail`, `more`, `less`, `grep`, `find`.
2. Разобраться с понятиями конвейер, перенаправление ввода-вывода.
3. Ознакомиться с информацией из рекомендованных источников и других про конвейеризации.
4. Повторить назначение прав доступа. Команды `chmod`, `chown`.
5. Ознакомиться с информацией по теме процессы, посмотреть и опробовать примеры наиболее распространенных команд, изучить возможность запуска процессов в `supervisor`.
6. Изучить возможность автоматического запуска программ по расписанию.

## Ход работы

### 1. Команда cat.

Команда cat предназначена для просмотра содержимого файла, а также для создания нового файла.

Используя команду cat > book.txt, создадим файл, введем необходимые символы, и после этого сочетанием клавиш Ctrl+D сохраним изменения.

Команда cat «имя файла» читает данные из файла или стандартного ввода и выводит их на экран. Мы посмотрим содержание файла 3.txt (рис. 1).

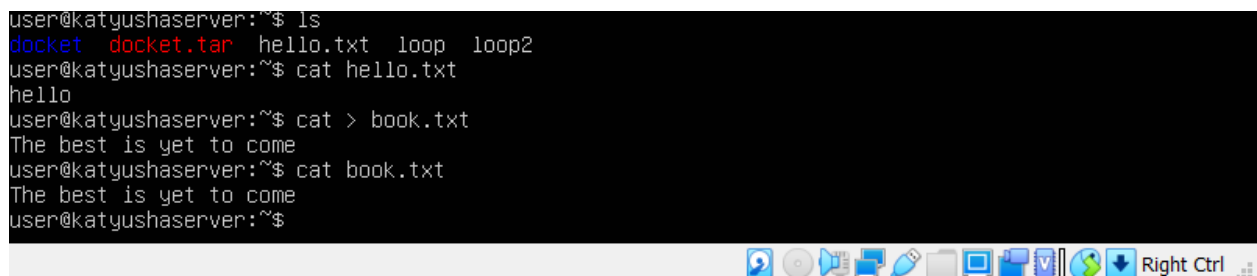
A screenshot of a terminal window with a black background and white text. The terminal shows a series of commands and their outputs. The prompt is 'user@katyushaserver:~\$'. The first command is 'ls', which lists 'docket', 'docket.tar', 'hello.txt', 'loop', and 'loop2'. The second command is 'cat hello.txt', which outputs 'hello'. The third command is 'cat > book.txt', which outputs 'The best is yet to come'. The fourth command is 'cat book.txt', which also outputs 'The best is yet to come'. The prompt returns to '~\$'. At the bottom of the terminal window, there is a taskbar with various icons and the text 'Right Ctrl'.

Рисунок 1 – Использование команды cat

## 2. Команда head.

Команда head выводит начальные строки (по умолчанию - 10) из одного или нескольких документов.

Используя команду head me.txt посмотрим первые 10 строк файла.

С помощью команды head hello.txt me.txt получим вывод с нескольких файлов.

Если десяти строк, по умолчанию выводимых командой, окажется мало или много, то их количество можно изменить с помощью опции -n. Выведем только первые 5 строк файла me.txt (рис. 2)

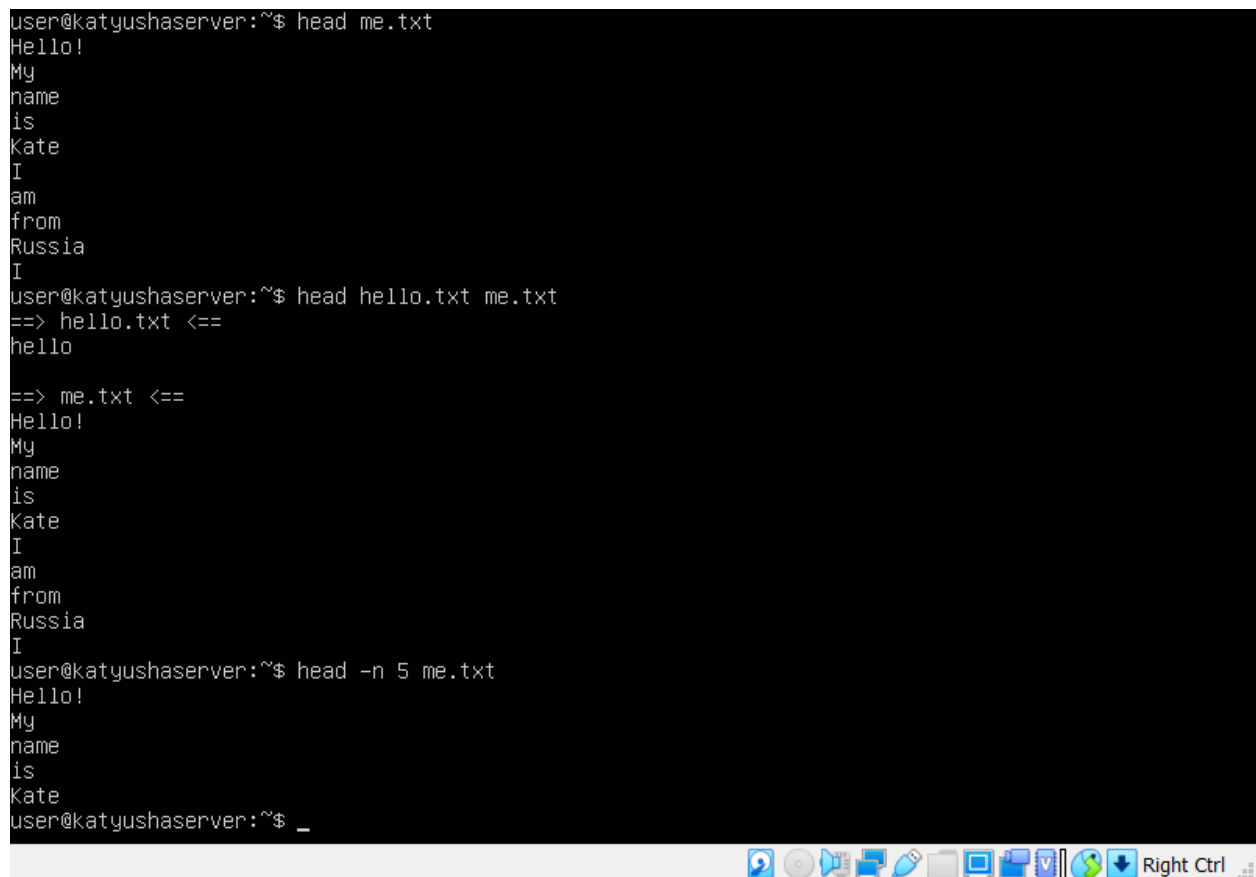
A screenshot of a terminal window with a black background and white text. The prompt is 'user@katyushaserver:~\$'. The first command is 'head me.txt', which outputs ten lines of text: 'Hello!', 'My', 'name', 'is', 'Kate', 'I', 'am', 'from', 'Russia', 'I'. The second command is 'head hello.txt me.txt', which outputs the first line of 'hello.txt' ('hello') followed by the first ten lines of 'me.txt' (the same text as above). The third command is 'head -n 5 me.txt', which outputs the first five lines of 'me.txt': 'Hello!', 'My', 'name', 'is', 'Kate'. The prompt returns to 'user@katyushaserver:~\$'. At the bottom of the terminal, there is a taskbar with various icons and the text 'Right Ctrl'.

Рисунок 2 – Использование команды head

### 3. Команда more.

Утилита more предназначена для постраничного просмотра файлов в терминале Linux.

На рисунке 3 показан просмотр текста из файла с помощью команды more me.txt.

Также в команде можно перечислить несколько имён файлов одно за другим, разделяя их пробелом. Содержимое этих файлов будет выведено в том же порядке. Посмотрим файлы hello.txt и book.txt.

Если файл находится не в текущей директории, нужно указывать его полный адрес.

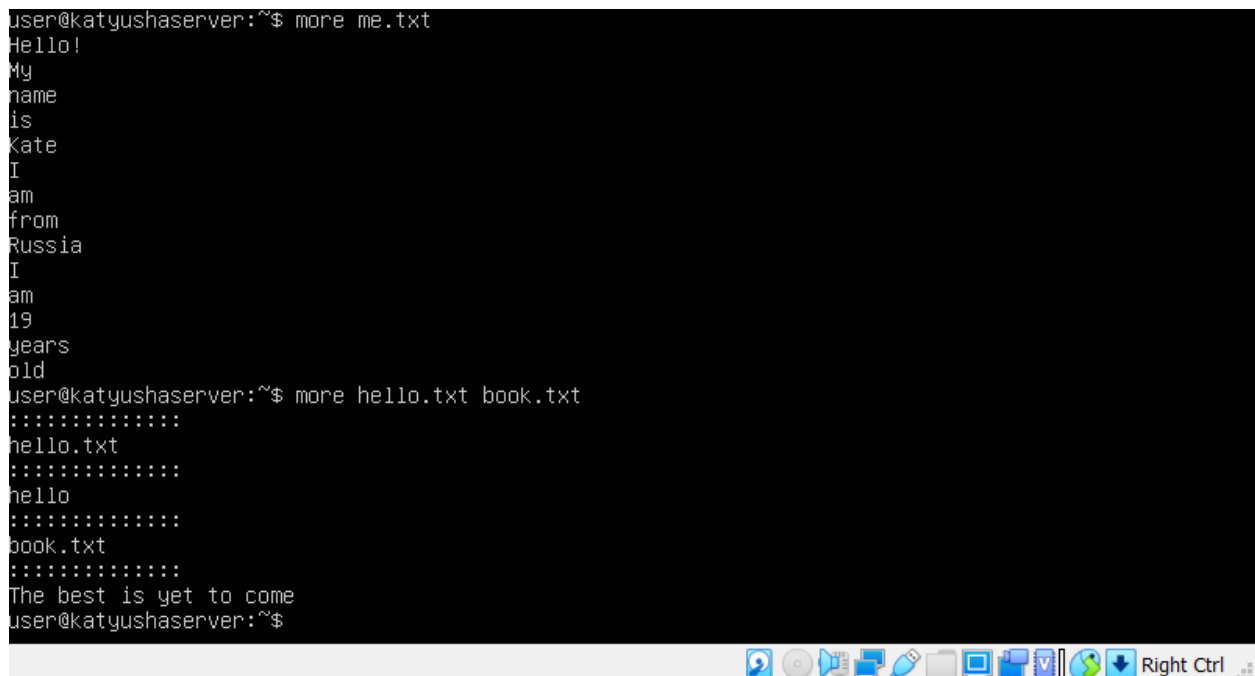
A screenshot of a terminal window with a black background and white text. The prompt is 'user@katyushaserver:~\$'. The first command is 'more me.txt', which displays the contents of 'me.txt' line by line: 'Hello!', 'My', 'name', 'is', 'Kate', 'I', 'am', 'from', 'Russia', 'I', 'am', '19', 'years', 'old'. The second command is 'more hello.txt book.txt'. This command lists the files 'hello.txt' and 'book.txt' with separator lines of dots. It then displays the contents of 'hello.txt' ('hello') and 'book.txt' ('The best is yet to come'). The prompt returns to 'user@katyushaserver:~\$'. At the bottom of the terminal window, there is a taskbar with various icons and the text 'Right Ctrl'.

Рисунок 3 – Просмотр содержимого файлов

Чтобы удобно посмотреть список всех процессов, запущенных в системе, используем команду `ps -ef | more`. Экран будет заполняться со списком данных, но остановится в конце страницы с сообщением «more». Для того, чтобы перейти к следующей странице необходимо нажать пробел на клавиатуре (рис. 4).

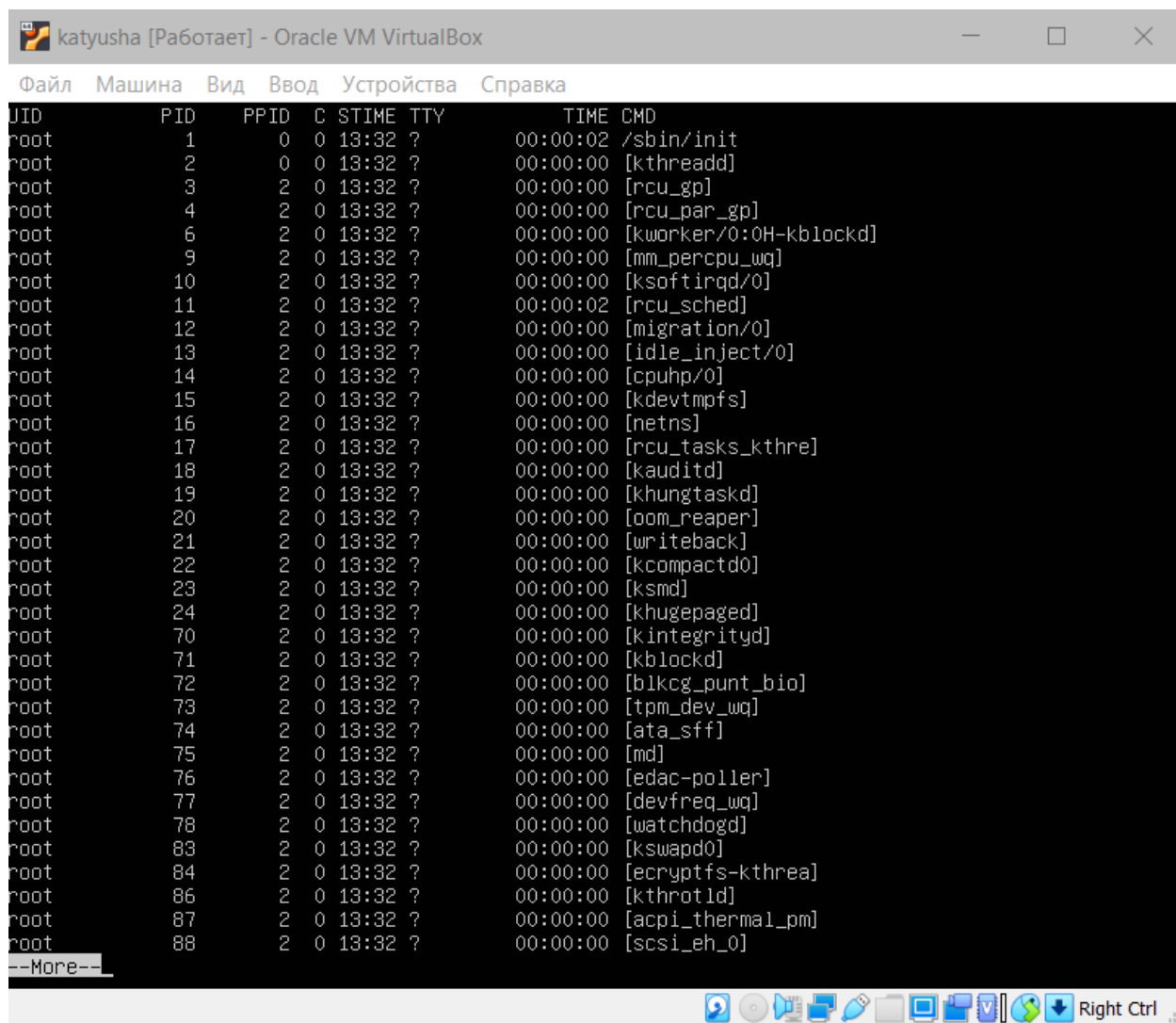


Рисунок 4 – Постраничный просмотр файлов



#### 4. Команда tail.

Команда `tail` по умолчанию выводит десять последних строк из файла, но ее поведение можно настроить с помощью опций.

Введем команду `tail me.txt`, чтобы посмотреть последние 10 строк файла. А затем введем команду `tail -n 5 me.txt`, чтобы посмотреть последние 5 строк файла (рис. 5).

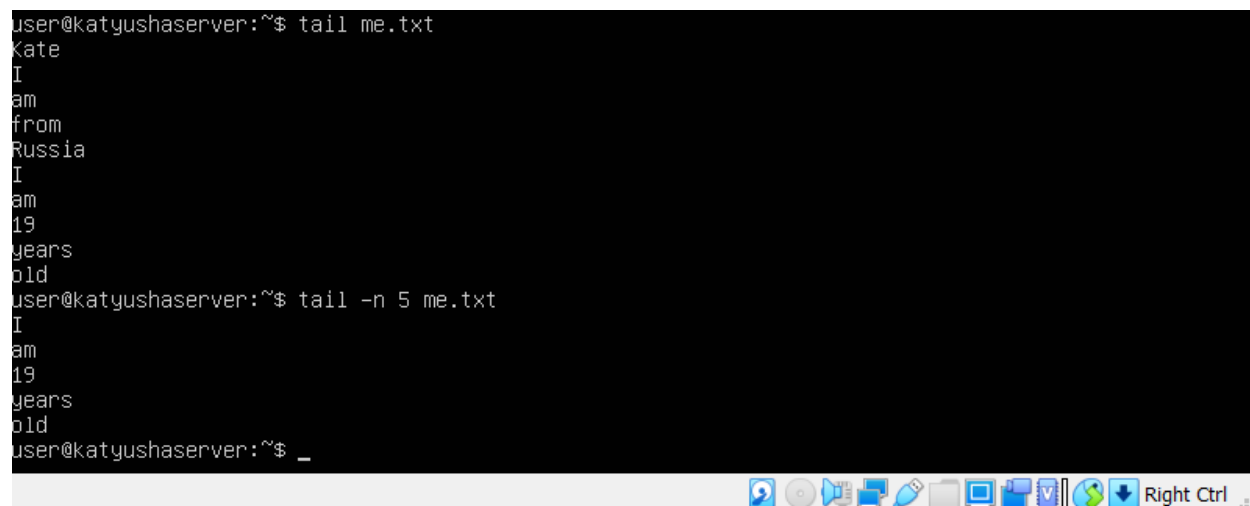
A screenshot of a terminal window with a black background and white text. The prompt is 'user@katyushaserver:~\$'. The first command entered is 'tail me.txt', which outputs ten lines of text: 'Kate', 'I', 'am', 'from', 'Russia', 'I', 'am', '19', 'years', 'old'. The second command entered is 'tail -n 5 me.txt', which outputs the last five lines of the file: 'I', 'am', '19', 'years', 'old'. The prompt returns to 'user@katyushaserver:~\$'. At the bottom of the terminal, there is a taskbar with various icons and the text 'Right Ctrl'.

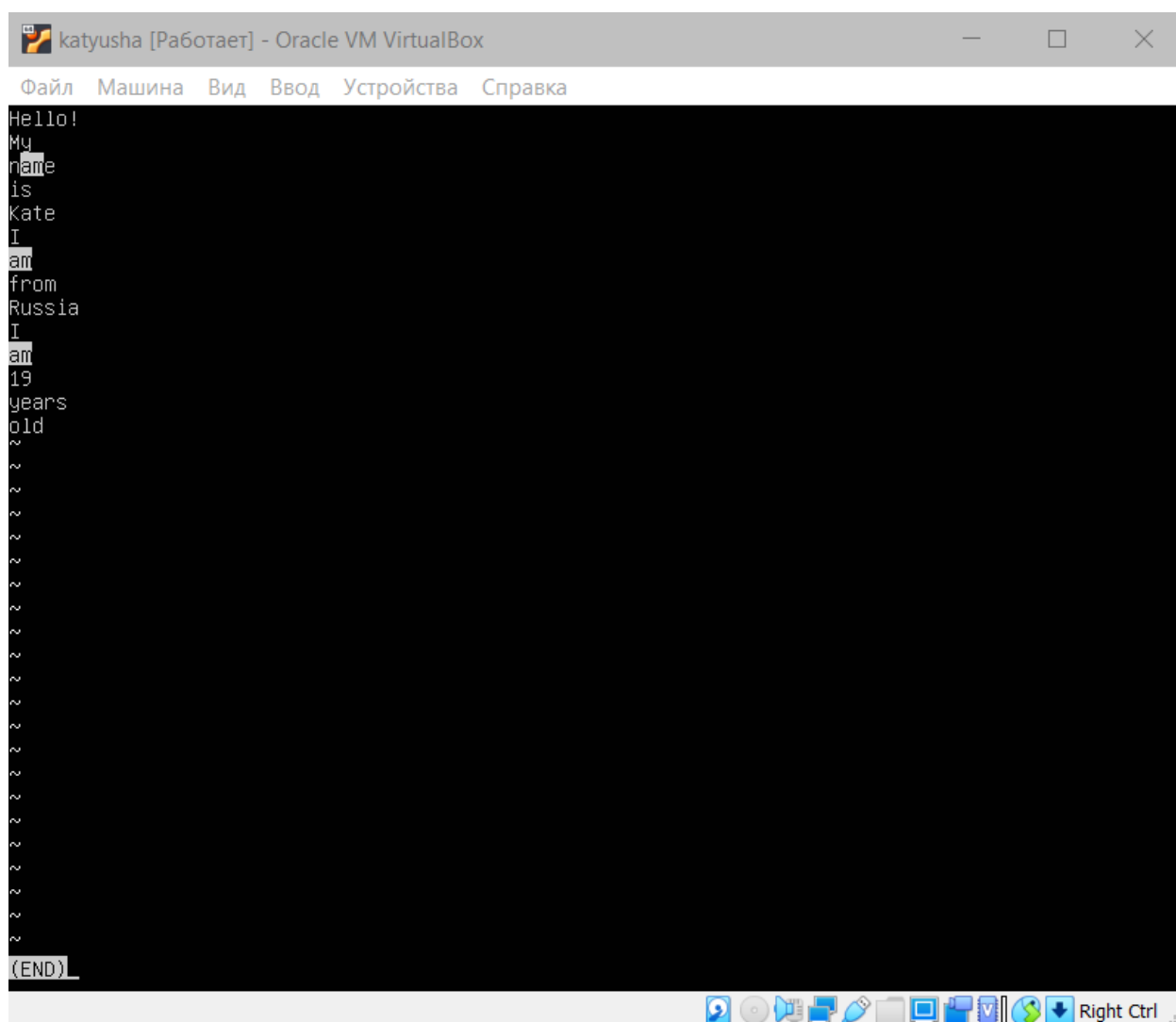
Рисунок 5 – Использование команды `tail`

## 5. Команда less.

Less – существенно более развитая команда для пролистывания текста. При чтении данных со стандартного ввода она создает буфер, который позволяет листать текст как вперед, так и назад.

На рисунке 6 показан просмотр файла `me.txt` с использованием команды `less me.txt`. Командная строка исчезает, а в окне терминала открывается указанный документ.

Утилиту less можно использовать не только для чтения текста, но и для поиска определенных участков в документе. Чтобы найти слово, нужно напечатать /текст и нажать Enter. Все участки текста, которые соответствуют условиям поиска, будут выделены. Найдем «am» в файле me.txt.




### Рисунок 6 – Использование команды less

## 6. Команда `grep`.

`Grep` – это утилита, которая даёт возможность вести поиск строки. С ее помощью можно искать конкретные слова в файл, а также можно передать вывод любой команды в `grep`, что упрощает работу во время поиска.

На рисунке 7 показан рекурсивный поиск в каталоге `/home` соответствующей строки с помощью команды «`ls -R /home | grep user`» и поиск строк «`am`» в файле `me.txt`.

A screenshot of a terminal window with a black background and white text. The prompt is 'user@katyushaserver:~\$'. The first command entered is 'ls -R /home | grep user', which outputs 'user', 'user2', '/home/user:', '/home/user/docket:', '/home/user/docket/docket2:', and '/home/user2:'. The second command is 'grep am /home/user/me.txt', which outputs 'name', 'am', and 'am'. The prompt returns to 'user@katyushaserver:~\$'. At the bottom of the terminal, there is a taskbar with various icons including a globe, a CD, a folder, a printer, a USB drive, a monitor, a document, a mail icon, and a 'Right Ctrl' button.

```
user@katyushaserver:~$ ls -R /home | grep user
user
user2
/home/user:
/home/user/docket:
/home/user/docket/docket2:
/home/user2:
user@katyushaserver:~$ grep am /home/user/me.txt
name
am
am
user@katyushaserver:~$
```

Рисунок 7 – Использование команды `grep`

## 7. Команда find.

Команда `find` позволяет искать файлы и каталоги, например, для поиска файлов по разрешениям, владельцам, группам, типу, размеру и другим критериям.

Выведем файлы в текущей директории с помощью команды `find`, найдем файлы с помощью команды «`find . -name "*.txt"`», а также найдем файлы, принадлежащие `user`, с помощью команды «`find -user user`» (рис. 8 и рис. 9).

```
user@katyushaserver:~$ find
.
./.lessht
./book.txt
./loop2
./docket
./docket/docket2
./docket/d1.txt
./docket/d2.txt
./me.txt
./hello.txt
./sudo_as_admin_successful
./.local
./.local/share
./.local/share/nano
./viminfo
./.cache
./.cache/motd.legal-displayed
./docket.tar
./profile
./bashrc
./bash_history
./config
./config/procps
./loop
./bash_logout
user@katyushaserver:~$ find . -name "*.txt"
./book.txt
./docket/d1.txt
./docket/d2.txt
./me.txt
./hello.txt
user@katyushaserver:~$
```

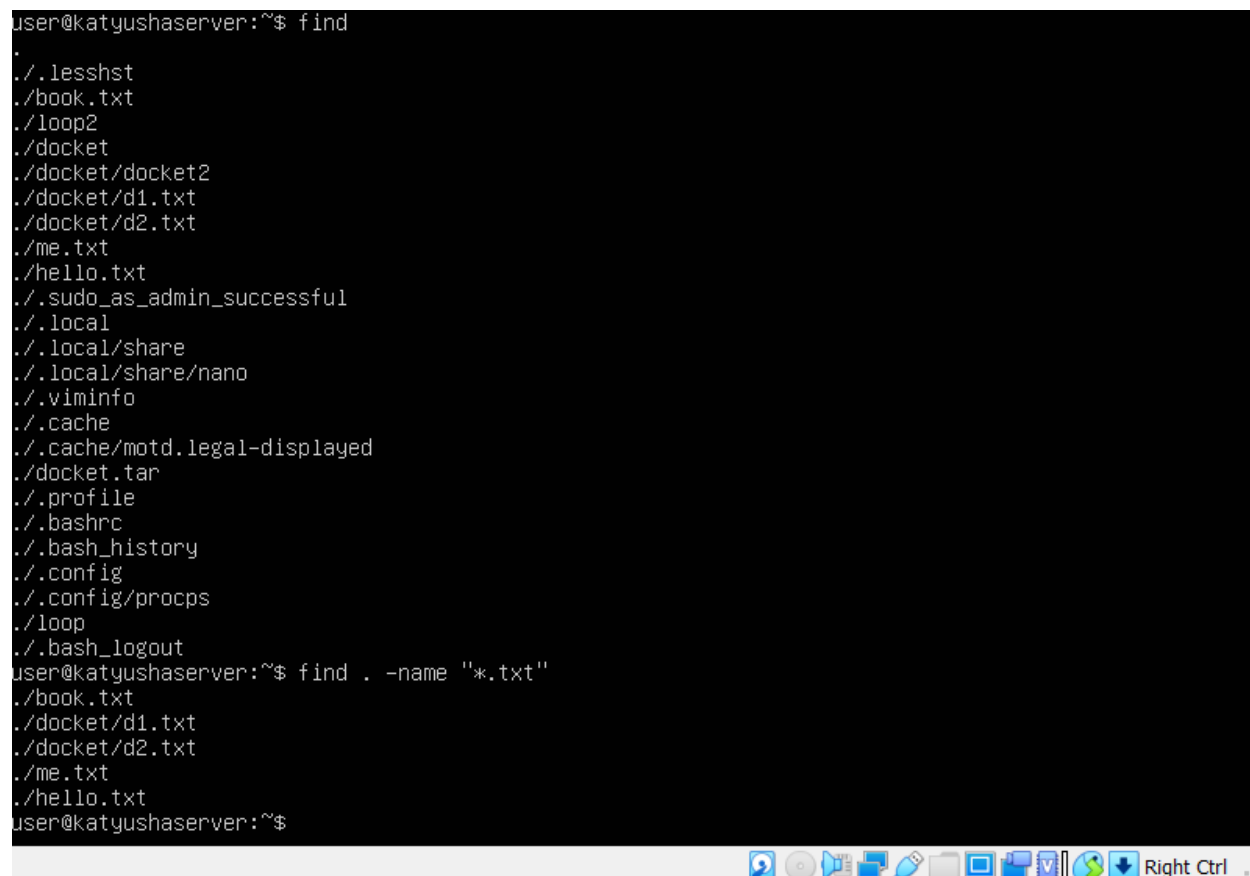


Рисунок 8 – Использование команды `find`

```
user@katyushaserver:~$ find -user user
.
./.lessht
./book.txt
./loop2
./docket
./docket/docket2
./docket/d1.txt
./docket/d2.txt
./me.txt
./hello.txt
./.sudo_as_admin_successful
./.local
./.local/share
./.local/share/nano
./.viminfo
./.cache
./.cache/motd.legal-displayed
./docket.tar
./profile
./bashrc
./bash_history
./config
./config/procps
./loop
./bash_logout
user@katyushaserver:~$
```



Рисунок 9 – Файлы, принадлежащие user

## 8. Конвейер, перенаправление ввода-вывода.

Существует возможность, не выходя из терминала в графический режим, записать в файл заметку. Для этого не обязательно использовать консольный текстовый редактор. Это можно сделать с помощью программы `cat`, перенаправив ее вывод в файл.

В работе с командной строкой Linux есть понятия стандартных устройств ввода, вывода и вывода ошибок.

`stdin` – стандартное устройство ввода. Имеет файловый указатель №0. Автоматически открывается всеми процессами.

`stdout` – стандартное устройство вывода. Имеет файловый указатель №1. Автоматически открывается всеми процессами.

`stderr` – стандартный поток ошибок (специальное устройство вывода для сообщений об ошибках. Имеет файловый указатель №2. Автоматически открывается всеми процессами.

По умолчанию практически все команды Linux используют для ввода информации `stdin`, а для вывода `stdout` и `stderr`, если их параметрами не указано обратное.

Операторы перенаправления способны изменять направление вывода и ввода информации. Так оператор:

`>` - перенаправляет стандартный поток в файл (другой поток). При этом если файл существует, то он перезаписывается, если не существует – создается.

`>>` - перенаправляет стандартный поток в файл. При этом если файл существует, то информация добавляется в конец, если не существует – файл создается.

`<` - перенаправляет содержимое указанного файла на стандартный ввод программы.

`>&` - перенаправляет стандартные потоки вывода и ошибок друг в друга.

На рисунке 10 показано перенаправление стандартного потока в файл с помощью команды `cat new1.txt > new2.txt`. После этого смотрим содержимое файла `new2.txt` и видим, что команда сработала.

```
user@katyushaserver:~$ ls
book.txt  docket  docket.tar  hello.txt  loop  loop2  me.txt
user@katyushaserver:~$ cat > new1.txt
what are you doing
i am dancing
user@katyushaserver:~$ cat new1.txt > new2.txt
user@katyushaserver:~$ cat new2.txt
what are you doing
i am dancing
user@katyushaserver:~$ _
```

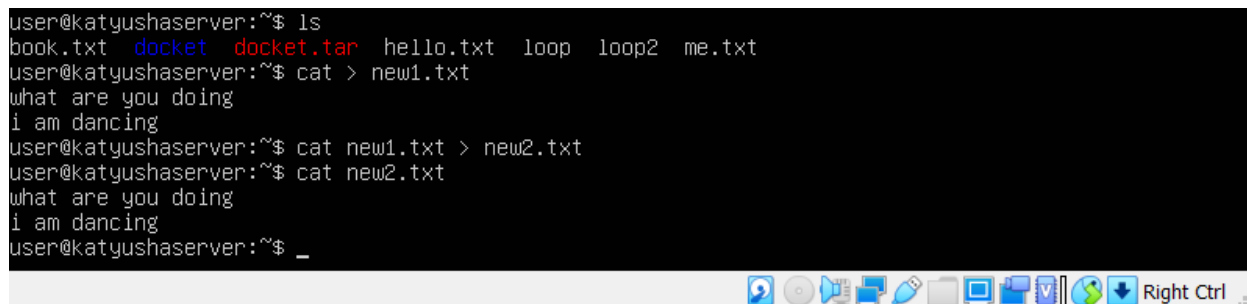


Рисунок 10 – Перенаправление потока в файл

На рисунке 11 показано добавление сегодняшней даты к файлу `new2.txt`. Перенаправим вывод команды `date` в тот же файл, используя команду «`date >> new2.txt`».

```
user@katyushaserver:~$ date >> new2.txt
user@katyushaserver:~$ cat new2.txt
what are you doing
i am dancing
Tue Nov 10 17:55:44 UTC 2020
user@katyushaserver:~$
```

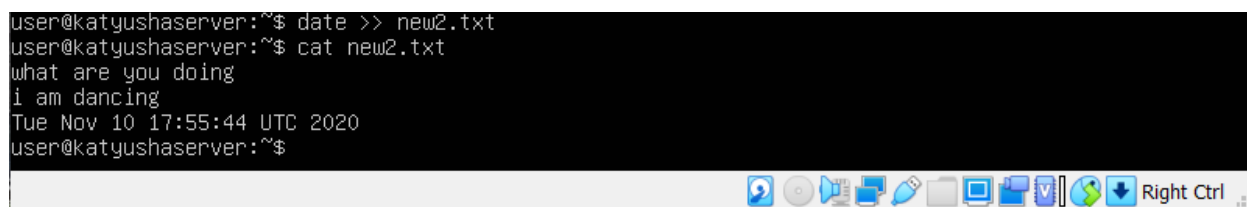


Рисунок 11 – Перенаправление вывода команды `date` в файл

На рисунке 12 показан способ перенаправления стандартного потока ошибок в файл с помощью команды «`cat nofile.txt > result.txt 2>&1`».

```
user@katyushaserver:~$ cat nofile.txt > result.txt 2>&1
user@katyushaserver:~$ cat result.txt
cat: nofile.txt: No such file or directory
user@katyushaserver:~$ _
```

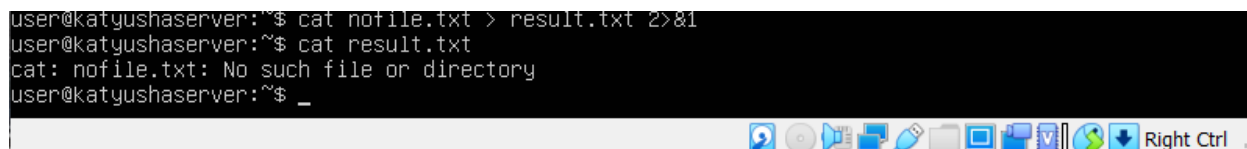


Рисунок 12 – Перенаправление вывода ошибок в файл

Канал – программный интерфейс, позволяющий процессам обмениваться данными (односторонний поток). Организацией канала занимается `shell`. Для управления каналом существует оператор «`|`».

Чтобы удобно посмотреть список всех процессов, запущенных в системе, используем команду `ps -ef | more`. «More» подразумевает постраничный просмотр файлов.

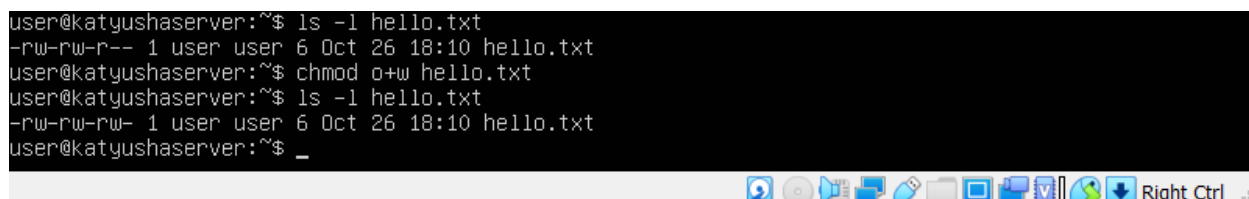
## 9. Команды chmod, chown.

Для распределения прав доступа в Linux существует множество команд. Основные из них – это chmod, chown и chgrp. Команда chmod изменяет права доступа к файлу. Для использования этой команды также необходимо иметь права владельца файла или права root. Синтаксис команды таков «chmod mode filename», где filename – имя файла, у которого изменяются права доступа, а mode – права доступа, устанавливаемые на файл.

Так как у каждого объекта в Linux имеется владелец, то права доступа применяются относительно владельца файла. Они состоят из набора 3 групп по три атрибута:

- чтение(r), запись(w), выполнение(x) для владельца;
- чтение, запись, выполнение для группы владельца;
- чтение, запись, выполнение для всех остальных.

На рисунке 13 показано предоставление права доступа на запись в файл hello.txt для пользователей, не являющихся пользователем и не входящим в группу пользователей.



```
user@katyushaserver:~$ ls -l hello.txt
-rw-rw-r-- 1 user user 6 Oct 26 18:10 hello.txt
user@katyushaserver:~$ chmod o+w hello.txt
user@katyushaserver:~$ ls -l hello.txt
-rw-rw-rw- 1 user user 6 Oct 26 18:10 hello.txt
user@katyushaserver:~$ _
```

Рисунок 13 – Изменение прав доступа

Команда chown позволяет сменить владельца файла. Для использования этой команды необходимо либо иметь права владельца текущего файла, либо права root. Синтаксис команды прост «chown username:groupname filename», где username – имя пользователя, groupname – имя группы, filename – имя файла, у которого сменяется владелец.

Имя группы в синтаксисе команды можно не указывать, тогда будет изменен только владелец файла.

На рисунке 14 показано изменение владельца файла и группы пользователей new1.txt с user на user2.



```
user@katyushaserver:~$ ls -l new1.txt
-rw-rw-r-- 1 user user 32 Nov 10 17:50 new1.txt
user@katyushaserver:~$ sudo chown user2:user2 new1.txt
user@katyushaserver:~$ ls -l new1.txt
-rw-rw-r-- 1 user2 user2 32 Nov 10 17:50 new1.txt
user@katyushaserver:~$
```



Рисунок 14 – Изменение владельца файла

## 10. Процессы.

Команда `top` обеспечивает вывод изменяющейся в реальном времени информации о запущенных процессах.

Вывод утилиты `top` включает в себя большое количество информации: средняя загрузка, количество запущенных процессов, состояние процессора, информация о свободной памяти.

На рисунке 15 показано использование команды `top` без параметров. Для выхода из данной команды следует нажать «q».

```
top - 16:14:43 up 4 min, 1 user, load average: 0.20, 0.35, 0.18
Tasks: 96 total, 1 running, 95 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1987.8 total, 1519.4 free, 132.7 used, 335.7 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 1700.7 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 502 root        rt   0 280288 18096  8184 S   0.3   0.9   0:00.06 multipathd
 926 user        20   0   7916   3716  3184 R   0.3   0.2   0:00.05 top
    1 root        20   0 101984 11400  8320 S   0.0   0.6   0:01.84 systemd
    2 root        20   0         0      0      0 S   0.0   0.0   0:00.00 kthreadd
    3 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    5 root        20   0         0      0      0 I   0.0   0.0   0:00.34 kworker/0:0-ata_sff
    6 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
    7 root        20   0         0      0      0 I   0.0   0.0   0:00.08 kworker/0:1-rcu_gp
    8 root        20   0         0      0      0 I   0.0   0.0   0:00.03 kworker/u2:0-events_power_e+
    9 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
   10 root        20   0         0      0      0 S   0.0   0.0   0:00.09 ksoftirqd/0
   11 root        20   0         0      0      0 I   0.0   0.0   0:00.32 rcu_sched
   12 root        rt   0         0      0      0 S   0.0   0.0   0:00.00 migration/0
   13 root       -51   0         0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
   14 root        20   0         0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
   15 root        20   0         0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
   16 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 netns
   17 root        20   0         0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_kthre
   18 root        20   0         0      0      0 S   0.0   0.0   0:00.00 kauditd
   19 root        20   0         0      0      0 S   0.0   0.0   0:00.00 khungtaskd
   20 root        20   0         0      0      0 S   0.0   0.0   0:00.00 oom_reaper
   21 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 writeback
   22 root        20   0         0      0      0 S   0.0   0.0   0:00.00 kcompactd0
   23 root        25   5         0      0      0 S   0.0   0.0   0:00.00 ksm
   24 root        39  19         0      0      0 S   0.0   0.0   0:00.00 khugepaged
   70 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 kintegrityd
   71 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 kblockd
   72 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 blkcg_punt_bio
   73 root         0 -20         0      0      0 I   0.0   0.0   0:00.00 tpm_dev_wq
```

Рисунок 15 – Использование команды `top`

PID — уникальный идентификатор процесса.

USER — имя пользователя, являющегося владельцем задачи.

PR — приоритет задачи в расписании.

NI — значение nice задачи. Отрицательное значение означает более высокий приоритет, а положительное значение nice означает более низкий приоритет

VIRT — общее количество используемой задачей виртуальной памяти, включает все коды, данные, совместные библиотеки, плюс страницы, которые были перенесены в раздел подкачки, и страницы, которые были размечены, но не используются.

RES — это используемая оперативная память, является подмножеством VIRT, представляет физическую память, не помещённую в раздел подкачки, которую в текущий момент использует задача.

SHR — размер совместной памяти, подмножество используемой памяти RES, которая может использоваться другими процессами.

S — статус процесса.

%CPU — использование центрального процессора, доля задачи в потреблённом процессорном времени с момента последнего обновления экрана, выражается в процентах от общего времени CPU.

%MEM — доля задачи в использовании памяти (RES).

TIME+ — общее время центрального процессора, которое использовала задача с момента запуска.

COMMAND — показывает строку команды, используемую для запуска задачи или имя ассоциированной программы.

Для вывода процессов, запущенных конкретным пользователем, используется параметр «-u». Посмотрим процессы пользователя user с помощью команды «top -u user» (рис. 16).

```

top - 16:40:10 up 29 min, 1 user, load average: 0.01, 0.04, 0.04
Tasks: 92 total, 1 running, 91 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1987.8 total, 1454.1 free, 132.3 used, 401.4 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 1699.3 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
  905 user       20   0  18552   9868  8196 S   0.0   0.5   0:00.16 systemd
  907 user       20   0 103200   3448   12 S   0.0   0.2   0:00.00 (sd-pam)
  912 user       20   0   7072   5020 3316 S   0.0   0.2   0:00.07 bash
 1456 user       20   0   7916   3772 3192 R   0.0   0.2   0:00.01 top
  
```

Рисунок 16 – Использование команды top

Для того, чтобы изобразить конкретный процесс с заданным идентификатором PID, используем параметр «-r». С помощью команды «top -r 1460» посмотрим информацию для сценария loop, запущенного в фоновом режиме (рис. 17).

```

top - 16:52:27 up 42 min, 1 user, load average: 0.49, 0.13, 0.04
Tasks: 1 total, 1 running, 0 sleeping, 0 stopped, 0 zombie
%Cpu(s):100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1987.8 total, 1457.3 free, 129.1 used, 401.4 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 1702.5 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
 1460 user       20   0   2608    612   540 R  99.9   0.0   0:42.36 sh
  
```

Рисунок 17 – Использование команды top

По умолчанию команда top системы обновляет выходные данные каждые 3 секунды. Для того, чтобы дать запрос на обновление выходных данных, нужно нажать клавишу пробела.

Для того, чтобы изменить частоту обновления выходных данных, нажимаем в интерактивном режиме клавишу d и вводим время, указываемое в секундах. Сделаем время обновления входных данных равным 10 секундам (рис. 18).

Change delay from 3.0 to 10

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1460	user	20	0	2608	612	540	R	93.8	0.0	8:49.35	sh
1	root	20	0	101984	11400	8320	S	0.0	0.6	0:01.89	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	20	0	0	0	0	I	0.0	0.0	0:02.14	kworker/0:0-events
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblockd
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.15	ksoftirqd/0
11	root	20	0	0	0	0	I	0.0	0.0	0:00.52	rcu_sched
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.01	migration/0
13	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
16	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd0
23	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
24	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
70	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
71	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
72	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
73	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
74	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff
75	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
76	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller

Рисунок 18 – Изменение времени обновления входных данных

Чтобы выделить запущенные процессы другим цветом, следует нажать клавишу «z». (рис. 19).

```

top - 17:06:29 up 56 min, 1 user, load average: 1.00, 0.96, 0.64
Tasks: 93 total, 2 running, 91 sleeping, 0 stopped, 0 zombie
%Cpu(s):100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1987.8 total, 1457.3 free, 129.1 used, 401.4 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 1702.5 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
1460 user      20   0    2608    612   540  R  99.7   0.0   14:43.05  sh
   1 root       20   0   101984  11400  8320  S   0.0   0.6    0:01.89  systemd
   2 root       20   0        0     0      0  S   0.0   0.0    0:00.00  kthreadd
   3 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  rcu_gp
   4 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  rcu_par_gp
   5 root       20   0        0     0      0  I   0.0   0.0    0:02.41  kworker/0:0-events
   6 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  kworker/0:0H-kblockd
   9 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  mm_percpu_wq
  10 root       20   0        0     0      0  S   0.0   0.0    0:00.15  ksoftirqd/0
  11 root       20   0        0     0      0  I   0.0   0.0    0:00.56  rcu_sched
  12 root        rt   0        0     0      0  S   0.0   0.0    0:00.02  migration/0
  13 root      -51   0        0     0      0  S   0.0   0.0    0:00.00  idle_inject/0
  14 root       20   0        0     0      0  S   0.0   0.0    0:00.00  cpuhp/0
  15 root       20   0        0     0      0  S   0.0   0.0    0:00.00  kdevtmpfs
  16 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  netns
  17 root       20   0        0     0      0  S   0.0   0.0    0:00.00  rcu_tasks_kthre
  18 root       20   0        0     0      0  S   0.0   0.0    0:00.00  kauditd
  19 root       20   0        0     0      0  S   0.0   0.0    0:00.00  khungtaskd
  20 root       20   0        0     0      0  S   0.0   0.0    0:00.00  oom_reaper
  21 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  writeback
  22 root       20   0        0     0      0  S   0.0   0.0    0:00.00  kcompactd0
  23 root       25   5        0     0      0  S   0.0   0.0    0:00.00  ksmd
  24 root       39  19        0     0      0  S   0.0   0.0    0:00.00  khugepaged
  70 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  kintegrityd
  71 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  kblockd
  72 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  blkcg_punt_bio
  73 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  tpm_dev_wq
  74 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  ata_sff
  75 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  md
  76 root        0 -20     0     0      0  I   0.0   0.0    0:00.00  edac-poller
  
```

Рисунок 19 – Выделение работающих процессов

Чтобы полностью увидеть команды, которые были инициаторами процессов, со всеми опциями командной строки и аргументами, нужно нажать «с». Мы увидим абсолютные пути команд, а также все опции и аргументы. (рис. 20).

```

top - 17:12:15 up 1:01, 1 user, load average: 1.00, 1.00, 0.77
Tasks: 93 total, 2 running, 91 sleeping, 0 stopped, 0 zombie
%Cpu(s): 99.7 us, 0.3 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1987.8 total, 1457.3 free, 129.1 used, 401.4 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 1702.5 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 1460 user      20   0    2608    612   540  R   99.0   0.0   20:28.47 sh loop
    1 root      20   0 101984 11400  8320  S    0.0   0.6    0:01.89 /sbin/init
    2 root      20   0      0      0      0  S    0.0   0.0    0:00.00 [kthreadd]
    3 root       0 -20      0      0      0  I    0.0   0.0    0:00.00 [rcu_gp]
    4 root       0 -20      0      0      0  I    0.0   0.0    0:00.00 [rcu_par_gp]
    5 root      20   0      0      0      0  I    0.0   0.0    0:02.55 [kworker/0:0-events]
    6 root       0 -20      0      0      0  I    0.0   0.0    0:00.00 [kworker/0:0H-kblockd]
    9 root       0 -20      0      0      0  I    0.0   0.0    0:00.00 [mm_percpu_wq]
   10 root      20   0      0      0      0  S    0.0   0.0    0:00.15 [ksoftirqd/0]
  
```

Рисунок 20 – Отображение абсолютных путей команд

Для сортировки отображения во время работы команды `top` можно ввести одно из значений: «М» - сортировать по объёму используемой памяти, «Р» - отсортировать по загрузке процессора, «и» - сортировать по имени пользователя.

На рисунке 21 показана сортировка по использованию объема памяти.

```

top - 17:16:41 up 1:06, 1 user, load average: 1.02, 1.01, 0.84
Tasks: 93 total, 2 running, 91 sleeping, 0 stopped, 0 zombie
%Cpu(s):100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1987.8 total, 1457.3 free, 129.1 used, 401.4 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 1702.5 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 609 root        20   0 636684 27272 15472 S   0.0   1.3   0:01.27 snapd
 655 root        20   0 105096 20872 13220 S   0.0   1.0   0:00.09 unattended-upgr
 502 root        rt    0 280288 18096  8184 S   0.0   0.9   0:00.39 multipathd
 606 root        20   0  26292 17932 10180 S   0.0   0.9   0:00.10 networkd-dispat
 331 root       19  -1  51644 15604 14492 S   0.0   0.8   0:00.27 systemd-journal
   1 root        20   0 101984 11400  8320 S   0.0   0.6   0:01.89 systemd
 905 user        20   0  18552  9868  8196 S   0.0   0.5   0:00.16 systemd
 593 root        20   0 238152  9428  8320 S   0.0   0.5   0:00.12 accounts-daemon
 579 systemd+   20   0  21144  9284  8116 S   0.0   0.5   0:00.13 systemd-resolve
 679 root        20   0 236292  8964  8044 S   0.0   0.4   0:00.01 polkitd
 610 root        20   0  16808  8072  7096 S   0.0   0.4   0:00.11 systemd-logind
 577 systemd+   20   0  26740  7688  6728 S   0.0   0.4   0:00.11 systemd-network
 535 systemd+   20   0  90388  6412  5536 S   0.0   0.3   0:00.12 systemd-timesyn
 361 root        20   0  21752  5896  4036 S   0.0   0.3   0:00.62 systemd-udevkd
 912 user        20   0   7072  5020  3316 S   0.0   0.2   0:00.10 bash
 607 syslog     20   0 224324  4752  3720 S   0.0   0.2   0:00.01 rsyslogd
 597 message+   20   0   7464  4668  3960 S   0.0   0.2   0:00.08 dbus-daemon
 625 root        20   0   5976  3940  3168 S   0.0   0.2   0:00.02 login
1476 user        20   0   7916  3664  3136 R   0.0   0.2   0:00.00 top
 907 user        20   0 103200  3448    12 S   0.0   0.2   0:00.00 (sd-pam)
 596 root        20   0   5568  2908  2688 S   0.0   0.1   0:00.01 cron
 619 daemon     20   0   3792  2284  2108 S   0.0   0.1   0:00.00 atd
1460 user        20   0   2608    612   540 R  99.7   0.0 24:54.12 sh
   2 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kthreadd
   3 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_gp
   4 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
   5 root        20   0      0      0      0 I   0.0   0.0   0:02.72 kworker/0:0-events
   6 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
   9 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
  10 root        20   0      0      0      0 S   0.0   0.0   0:00.16 ksoftirqd/0
  
```

Рисунок 21 – Сортировка процессов по использованию памяти

По умолчанию `top` обновляет выводимые данные до нажатия «q», а параметр `-n` при запуске позволяет указать необходимое количество обновлений (итераций), после которых выход произойдет автоматически.

Сделаем так, чтобы выполнялось всего 2 обновления, с помощью команды «`top -n 2`» (рис. 22).



```
katyusha [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Tasks: 93 total,  2 running, 91 sleeping,  0 stopped,  0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 1987.8 total, 1457.1 free,  129.3 used,  401.4 buff/cache
MiB Swap: 2048.0 total, 2048.0 free,   0.0 used. 1702.3 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 1460 user      20   0   2608    612    540  R  99.9   0.0  42:32.62 sh
 1480 root      20   0     0      0      0   I  0.3   0.0   0:00.03 kworker/u2:0-events_power_e+
    1 root      20   0 101984 11400   8320  S   0.0   0.6   0:01.90 systemd
    2 root      20   0     0      0      0   S   0.0   0.0   0:00.00 kthreadd
    3 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 rcu_gp
    4 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 rcu_par_gp
    5 root      20   0     0      0      0   I  0.0   0.0   0:03.07 kworker/0:0-events
    6 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 kworker/0:0H-kblockd
    9 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 mm_percpu_wq
   10 root      20   0     0      0      0   S   0.0   0.0   0:00.17 ksoftirqd/0
   11 root      20   0     0      0      0   I  0.0   0.0   0:00.60 rcu_sched
   12 root      rt   0     0      0      0   S   0.0   0.0   0:00.02 migration/0
   13 root     -51   0     0      0      0   S   0.0   0.0   0:00.00 idle_inject/0
   14 root      20   0     0      0      0   S   0.0   0.0   0:00.00 cpuhp/0
   15 root      20   0     0      0      0   S   0.0   0.0   0:00.00 kdevtmpfs
   16 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 netns
   17 root      20   0     0      0      0   S   0.0   0.0   0:00.00 rcu_tasks_kthre
   18 root      20   0     0      0      0   S   0.0   0.0   0:00.00 kauditd
   19 root      20   0     0      0      0   S   0.0   0.0   0:00.00 khungtaskd
   20 root      20   0     0      0      0   S   0.0   0.0   0:00.00 oom_reaper
   21 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 writeback
   22 root      20   0     0      0      0   S   0.0   0.0   0:00.00 kcompactd0
   23 root      25   5     0      0      0   S   0.0   0.0   0:00.00 ksm
   24 root      39  19     0      0      0   S   0.0   0.0   0:00.00 khugepaged
   70 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 kintegrityd
   71 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 kblockd
   72 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 blkcg_punt_bio
   73 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 tpm_dev_wq
   74 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 ata_sff
   75 root      0 -20     0      0      0   I  0.0   0.0   0:00.00 md
user@katyushaserver:~$ _
```

Рисунок 22 – Выход после заданного числа итераций

Для сохранения выводимых командой `top` результатов в файл используем команду «`top -n 1 -b > top-output.txt`». После этого, используя команду `head`, посмотрим первые 10 строк файла `top-output.txt` и увидим, что вывод команды `top` действительно сохранился в файл. Благодаря параметру «`-b`» `top` не будет принимать входных команд, выполнив заданное параметром «`-n`» количество обновлений (рис. 23).

```
user@katyushaserver:~$ top -n 1 -b > top-output.txt
user@katyushaserver:~$ head top-output.txt
top - 17:48:02 up 1:37, 1 user, load average: 1.00, 1.00, 1.00
Tasks: 92 total,  2 running, 90 sleeping,  0 stopped,  0 zombie
%Cpu(s): 93.8 us,  6.2 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 1987.8 total, 1457.3 free,  129.0 used,  401.5 buff/cache
MiB Swap: 2048.0 total, 2048.0 free,   0.0 used. 1702.6 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 1460 user      20   0   2608    612    540  R  99.9   0.0  56:12.53 sh
    1 root      20   0 101984 11400   8320  S   0.0   0.6   0:01.91 systemd
    2 root      20   0     0      0      0   S   0.0   0.0   0:00.00 kthreadd
user@katyushaserver:~$ _
```

Рисунок 23 – Сохранение результатов команды `top` в файл

Чтобы уничтожить процесс, нужно нажать клавишу «k», будет запрошен идентификатор процесса PID и будет послан сигнал на уничтожение процесса. Если у нас достаточно привилегий для того, чтобы уничтожить конкретный PID, операция уничтожения будет выполнена успешно. Завершим процесс с PID = 1460 (рис. 24). После ввода необходимо нажать Enter.

```

top - 18:07:58 up 1:57, 1 user, load average: 1.00, 1.00, 1.00
Tasks: 93 total, 2 running, 91 sleeping, 0 stopped, 0 zombie
%Cpu(s):100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1987.8 total, 1457.3 free, 128.9 used, 401.6 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 1702.7 avail Mem
Send pid 1460 signal [15/sigterm]

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1460	user	20	0	2608	612	540	R	99.9	0.0	76:06.82	sh
1	root	20	0	101984	11400	8320	S	0.0	0.6	0:01.91	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd

Рисунок 24 – Завершение процесса в команде top

Для получения справки по основным командам утилиты top следует нажать клавишу «h» (рис. 25).

```

Help for Interactive Commands - procs-ng UNKNOWN
Window 1:Def: Cumulative mode Off. System: Delay 3.0 secs; Secure mode Off.

Z,B,E,e Global: 'Z' colors; 'B' bold; 'E'/'e' summary/task memory scale
l,t,m Toggle Summary: 'l' load avg; 't' task/cpu stats; 'm' memory info
0,1,2,3,I Toggle: '0' zeros; '1/2/3' cpus or numa node views; 'I' Irix mode
f,F,X Fields: 'f'/'F' add/remove/order/sort; 'X' increase fixed-width

L,&,<,> . Locate: 'L'/'&' find/again; Move sort column: '<'/'>' left/right
R,H,J,C . Toggle: 'R' Sort; 'H' Threads; 'J' Num justify; 'C' Coordinates
c,i,S,j . Toggle: 'c' Cmd name/line; 'i' Idle; 'S' Time; 'j' Str justify
x,y . Toggle highlights: 'x' sort field; 'y' running tasks
z,b . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
u,U,o,O . Filter by: 'u'/'U' effective/any user; 'o'/'O' other criteria
n,#,^O . Set: 'n'/'#' max tasks displayed; Show: Ctrl+'O' other filter(s)
V,v . Toggle: 'V' forest view; 'v' hide/show forest view children

k,r Manipulate tasks: 'k' kill; 'r' renice
d or s Set update interval
W,Y Write configuration file 'W'; Inspect other output 'Y'
q Quit
( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
Type 'q' or <Esc> to continue

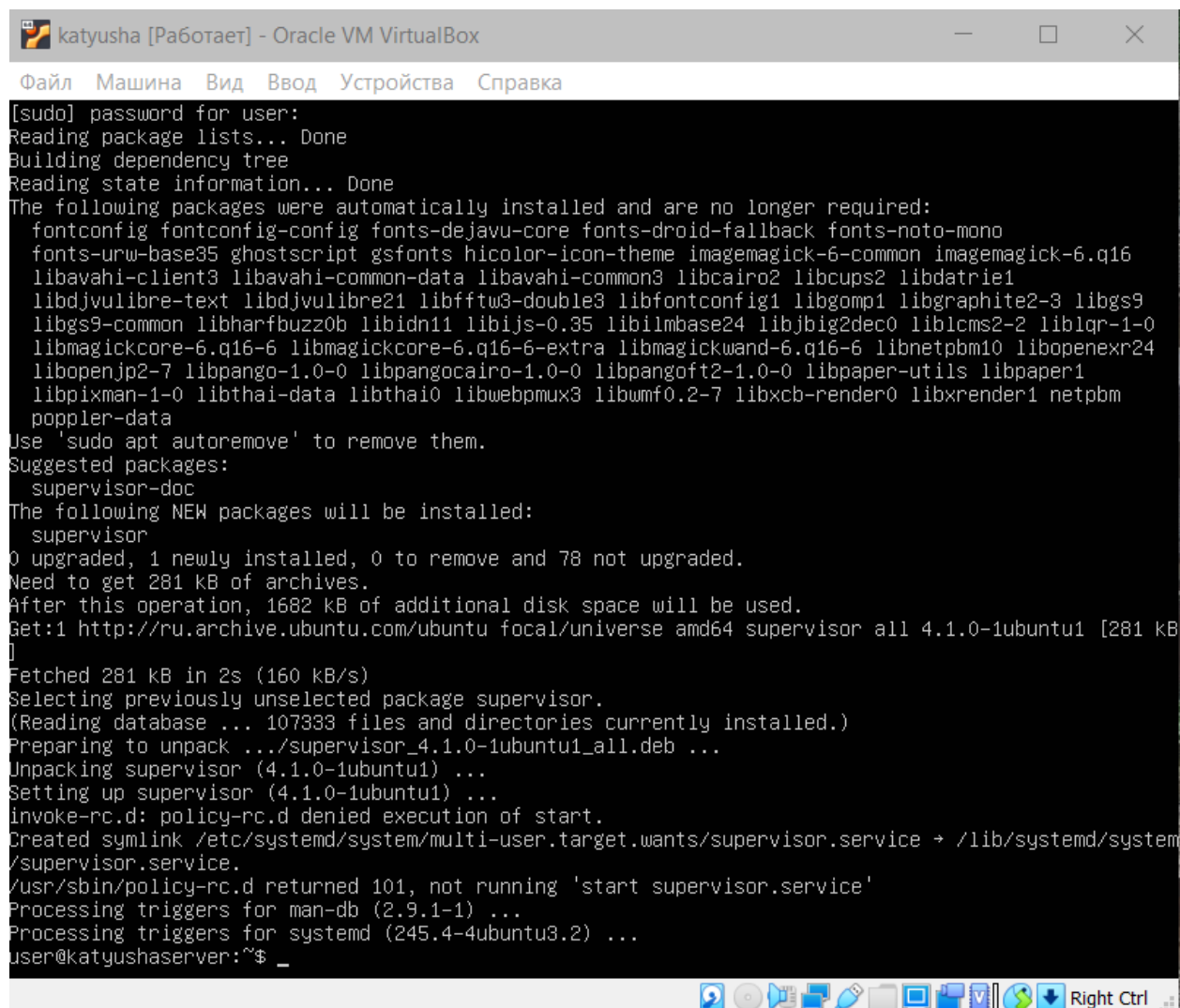
```

Рисунок 25 – Справка об основных командах

## 11. Supervisor.

Supervisor – это система клиент/сервер, при помощи которой пользователь (администратор) может контролировать подключенные процессы в системах типа UNIX. Инструмент создает процессы в виде под-процессов от своего иетсмени, поэтому имеет полный контроль над ними.

Для начала нужно установить supervisor. Для этого используем команду «apt-get install supervisor» (рис. 26).



```
[sudo] password for user:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fontconfig fontconfig-config fonts-dejavu-core fonts-droid-fallback fonts-terminus
  fonts-urw-base35 ghostscript gsfonts hicolor-icon-theme imagemagick-6.q16
  libavahi-client3 libavahi-common-data libavahi-common3 libcairo2 libcups2 libdat1
  libdjvulibre-text libdjvulibre21 libfftw3-double3 libfontconfig1 libgomp1 libgraphite2-3 libgs9
  libgs9-common libharfbuzz0b libidn11 libijs-0.35 liblmbase24 libjpeg2dec0 liblms2-2 liblqr-1-0
  libmagickcore-6.q16-6 libmagickcore-6.q16-6-extra libmagickwand-6.q16-6 libnetpbm10 libopenexr24
  libopenjp2-7 libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpaper-utils libpaper1
  libpixman-1-0 libthai-data libthai0 libwebpmux3 libwmf0.2-7 libxcb-render0 libxrender1 netpbm
  poppler-data
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  supervisor-doc
The following NEW packages will be installed:
  supervisor
0 upgraded, 1 newly installed, 0 to remove and 78 not upgraded.
Need to get 281 kB of archives.
After this operation, 1682 kB of additional disk space will be used.
Get:1 http://ru.archive.ubuntu.com/ubuntu focal/universe amd64 supervisor all 4.1.0-1ubuntu1 [281 kB]
Fetched 281 kB in 2s (160 kB/s)
Selecting previously unselected package supervisor.
(Reading database ... 107333 files and directories currently installed.)
Preparing to unpack .../supervisor_4.1.0-1ubuntu1_all.deb ...
Unpacking supervisor (4.1.0-1ubuntu1) ...
Setting up supervisor (4.1.0-1ubuntu1) ...
invoke-rc.d: policy-rc.d denied execution of start.
Created symlink /etc/systemd/system/multi-user.target.wants/supervisor.service → /lib/systemd/system/supervisor.service.
/usr/sbin/policy-rc.d returned 101, not running 'start supervisor.service'
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.2) ...
user@katyushaserver:~$
```

Рисунок 26 – Установка supervisor

После установки нужно сконфигурировать и добавить программы/процессы, которыми будет управлять supervisor. Файл конфигурации по умолчанию находится в /etc/supervisor/supervisord.conf.

Создаем сценарий «motivation» и пишем внутри скрипт. Затем переходим в каталог с помощью команды «cd /etc/supervisor/conf.d» Создаем файл запуска процесса, используя команду «nano test.conf» (рис. 27).

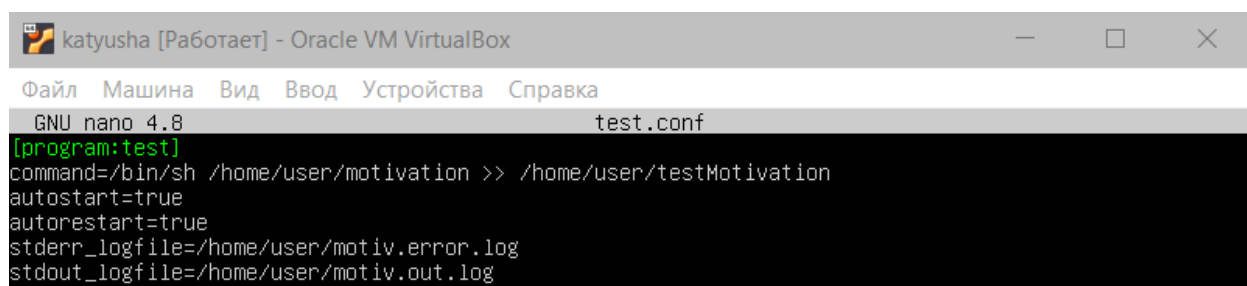


```
user@katyushaserver:~$ cat > motivation
while true; do true; echo 'Just do it'; sleep 3; done
user@katyushaserver:~$ cd /etc/supervisor/conf.d
user@katyushaserver:/etc/supervisor/conf.d$ nano test.conf_
```

Рисунок 27 – Создание скрипта и файла запуска

Для добавления нового процесса (воркера) нужно дополнить файл кодом:

[program:test] – название процесса;  
command= /bin/sh /home/user/motivation >>  
/home/user/testMotivation – команда на запуск скрипта;  
autostart=true – запуска воркера вместе с supervisor;  
autorestart = true – перезапуск воркера;  
stderr\_logfile=/home/user/motiv.error.log – файл с выводом лога  
ошибки;  
stdout\_logfile=/home/user/motiv.out.log – файл с выводом лога  
процесса при работе



```
katyusha [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
GNU nano 4.8                                test.conf
[program:test]
command=/bin/sh /home/user/motivation >> /home/user/testMotivation
autostart=true
autorestart=true
stderr_logfile=/home/user/motiv.error.log
stdout_logfile=/home/user/motiv.out.log
```

Рисунок 28 – Написание содержимого файла запуска процесса

Чтобы supervisor считал обновленные настройки пишем покоманду «supervisorctl reread». А после этого с помощью команды «supervisorctl update» нужно дать команду запуска все сконфигурированных процессов. Также используя команду ls, видим, что появились ли у нас файлы с выводом лога ошибок и выводом лога процесса при работе (рис. 29).



## 12. Cron.

Для выполнения конкретных задач по расписанию, существует утилита cron.

Cron – программа-демон, предназначенная для выполнения заданий в определенное время, или через определенные промежутки времени.

Описания регулярных действий, запускаемых утилитой – это так называемая crontab-таблица, которая имеет строго определенный формат. Она состоит из 6 колонок, разделённых табуляторами или пробелами, первые 5 из которых определяют время запуска действия: «minute(s) hour(s) day(s) month(s) weekday(s) command(s)». Сначала задаётся колонка минут, затем часов, дней, месяцев и дней недели. Для задания шага значений используется символ «/». Последняя колонка интерпретируется как команда запуска, то есть само действие.

Для редактирования файла расписания необходимо использовать команду `crontab -e`, а для удаления `crontab -r`.

На рисунке 31 показано добавление процесса, который добавляет фразу «I love you» в текстовый файл каждую минуту.

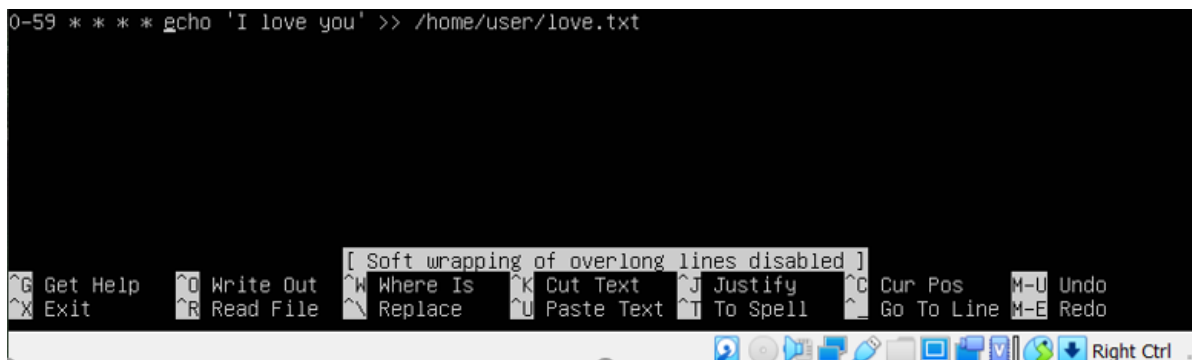


Рисунок 31 – Редактирование файла

На рисунке 32 видим содержимое файла, в который добавилась фраза «I love you».



Рисунок 32 – Содержимое файла

## Вывод

В результате выполнения лабораторной работы я получила знания по работе с процессами в ОС Linux Ubuntu. Научилась пользоваться перенаправлением ввода-вывода, «Supervisor», планировщиком задач. Выполнила основные команды просмотра файлов и изучила существующие у них параметры.

.