

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа № 3

По дисциплине «OS Linux»

Процессы в операционной системе Linux

Студент

Бахмутский М.В.

Группа АС-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Цель работы

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

Ход работы

1 Задание 1

Название команды - это сокращения от слова *catenate*. По сути, задача команды *cat* очень проста - она читает данные из файла или стандартного ввода и выводит их на экран. Это все, чем занимается утилита. Но с помощью ее опций и операторов перенаправления вывода можно сделать очень многое. Сначала рассмотрим синтаксис утилиты:

`$ cat опции файл1 файл2 ...`

Можно передать утилите несколько файлов и тогда их содержимое будет выведено поочередно, без разделителей. Опции позволяют очень сильно видоизменить вывод и сделать именно то, что вам нужно. Рассмотрим основные опции:

- b - нумеровать только непустые строки;
- E - показывать символ \$ в конце каждой строки;
- n - нумеровать все строки;
- s - удалять пустые повторяющиеся строки;
- T - отображать табуляции в виде ^I;
- h - отобразить справку;
- v - версия утилиты.

```
nuviky@nuviky:~$ cat test.txt
fsd
vxd
fgs
drydrtYu45

sdfr
gwerytg3
456v
345tertgvdrth
vdrf
ghvdfgh
dfghv
dfgvhdfghvd
fghv
dfg
hvd
fg
hvdr
th
ertuy
e456
745
6u
546nuby6uj
rety
hvds
rth
vert
hvrt
yj
nuviky@nuviky:~$ _
```

Рисунок 1 – Пример выполнения команды cat

Синтаксис у команды head следующий:

\$ head опции файл

Здесь:

Опции — это параметр, который позволяет настраивать работу команды таким образом, чтобы результат соответствовал конкретным потребностям пользователя.

Файл — это имя документа (или имена документов, если их несколько). Если это значение не задано либо вместо него стоит знак «-», команда будет брать данные из стандартного вывода.

Чаще всего к команде head применяются такие опции:

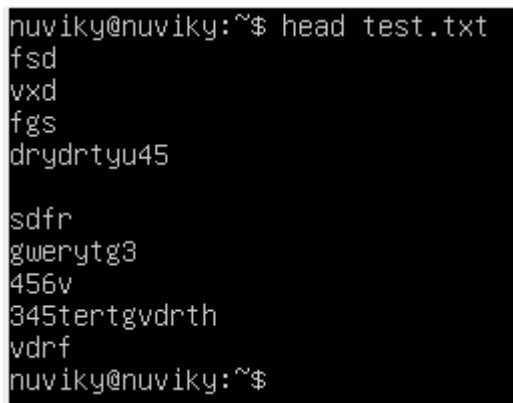
-c (--bytes) — позволяет задавать количество текста не в строках, а в байтах. При записи в виде --bytes=[-]NUM выводит на экран все содержимое файла, кроме NUM байт, расположенных в конце документа.

-n (--lines) — показывает заданное количество строк вместо 10, которые выводятся по умолчанию. Если записать эту опцию в виде --lines=[-]NUM, будет показан весь текст кроме последних NUM строк.

-q (--quiet, --silent) — выводит только текст, не добавляя к нему название файла.

-v (--verbose) — перед текстом выводит название файла.

-z (--zero-terminated) — символы перехода на новую строку заменяет символами завершения строк.



```
nuviky@nuviky:~$ head test.txt
fsd
vxd
fgs
drydrtyu45

sdf
gwerytg3
456v
345tertgvdrth
vdrf
nuviky@nuviky:~$
```

Рисунок 2 – Пример выполнения команды head

Синтаксис у команды tail следующий:

\$ tail опции файл

По умолчанию утилита выводит десять последних строк из файла, но ее поведение можно настроить с помощью опций:

-c - выводить указанное количество байт с конца файла;

-f - обновлять информацию по мере появления новых строк в файле;

-n - выводить указанное количество строк из конца файла;

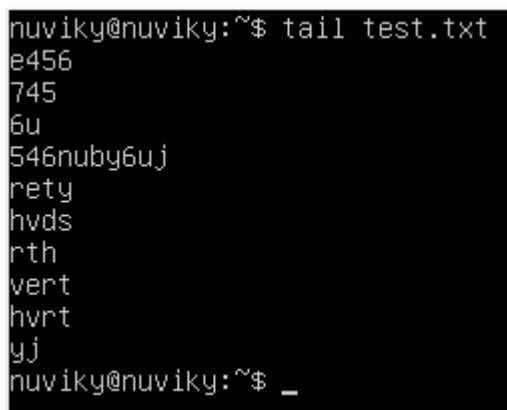
--pid - используется с опцией -f, позволяет завершить работу утилиты, когда завершится указанный процесс;

-q - не выводить имена файлов;

--retry - повторять попытки открыть файл, если он недоступен;

-v - выводить подробную информацию о файле;

В качестве значения параметра -s можно использовать число с приставкой b, kB, K, MB, M, GB, G T, P, E, Z, Y. Еще есть одно замечание по поводу имен файлов. По умолчанию утилита не отслеживает изменение имен, но вы можете указать что нужно отслеживать файл по дескриптору, подробнее в примерах.



```
nuviky@nuviky:~$ tail test.txt
e456
745
6u
546nuby6uj
rety
hvds
rth
vert
hvrt
yj
nuviky@nuviky:~$ _
```

Рисунок 3 – Пример выполнения команды tail

Утилита more предназначена для постраничного просмотра файлов в терминале Linux. Своим названием она обязана надписи more (в русскоязычном варианте — дальше), появляющейся внизу каждой страницы.

Команда more linux — одна из самых примитивных команд для работы с текстом. Её ближайшая родственница — команда less — обладает куда большим набором опций и дополнительных возможностей.

Синтаксис у команды more следующий:

\$ more опции файл

Список опций команды:

- d — вывод информации в конце страницы о клавишах, использующихся для продолжения работы, завершения её или получения инструкций;
- l — игнорирование в тексте символа разрыва страницы;
- f — подсчёт числа логических строк вместо экранных;
- r — очистка экрана терминала для того, чтобы пользователю не пришлось пользоваться прокруткой перед выводом следующей порции текста;

-c — устранение потребности в прокрутке (как и -p) — отображение текста, начиная с верха экрана, и стирание при этом предыдущего вывода построчно;

-s — замена нескольких пустых строк, расположенных подряд, одной пустой строкой;

-u — удаление подчёркивания;

-n — отображение n-го количества строк;

+n — отображение текста, начиная со строки с номером n;

+/строка — поиск в файле указанной строки и начало вывода текста именно с неё;

--help — вызов справки;

-v (--version) — вывод на экран текущей версии утилиты.

Также у команды more есть собственные горячие клавиши и интерактивные команды:

h (?) — помощь (вывод информации только об интерактивных командах);

ПРОБЕЛ — отображение следующей порции текста (по умолчанию количество строк зависит от текущего размера окна терминала);

z — то же, что и ПРОБЕЛ;

ENTER — вывод текста построчно (шаг команды — одна строка);

d (^D) — прокрутка текста на количество строк, соответствующее размеру терминала;

q (Q) — выход из утилиты;

s — переход на одну строку вперёд;

f — переход на одну экранную страницу вперёд;

b (^B) — переход на одну экранную страницу назад;

' — возвращение к месту начала поиска;

= — отображение текущего количества строк;

/pattern — поиск с использованием регулярных выражений;

n — поиск слов и фраз, соответствующих последнему использованному регулярному выражению;

!command (:command) — выполнение команды в субоболочке;

v — открытие файла в текстовом редакторе, назначенном по умолчанию, а если таковой не найден, использование консольного текстового редактора для открытия файла;

^L — удаление с экрана всего, кроме содержимого файла;

:n — переход к следующему файлу;

:p — переход к предыдущему файлу;

:f — вывод названия текущего файла и количества строк в нём;

. - повторное выполнение предыдущей команды.



```
nuviky@nuviky:~$ more test.txt
fsd
vxd
fgs
drydrtyu45

sdfr
gwerytg3
456v
345tertgvdrth
vdrf
ghvdfgh
dfghv
dfgvhdfghvd
fghv
dfg
hvd
fg
hvdr
th
ertuy
e456
745
6u
546nuby6uj
rety
hvds
rth
vert
hvrt
yj
nuviky@nuviky:~$ _
```

Рисунок 4 – Пример выполнения команды more

Команда `less` — она позволяет перематывать текст не только вперёд, но и назад, осуществлять поиск в обоих направлениях, переходить сразу в конец или в начало файла.

Особенность `less` заключается в том, что команда не считывает текст полностью, а загружает его небольшими фрагментами.

Синтаксис у команды `less` следующий:

`less` опции файл

Наиболее популярные опции:

`-a, --search-skip-screen` — не осуществлять поиск в тексте, который в данный момент отображен на экране;

`-bn, --buffers=n` — задать размер буфера памяти;

`-c, --clear-screen` — листать текст, полностью стирая содержимое экрана (построчная прокрутка работать не будет);

`-Dxcolor, --color=xcolor` — задать цвет отображаемого текста;

`-E, --QUIT-AT-EOF` — выйти, когда утилита достигнет конца файла;

`-e, --quit-at-eof` — выйти, когда утилита второй раз достигнет конца файла;

`-F, --quit-if-one-screen` — выйти, если содержимое файла помещается на одном экране;

`-f, --force` — открыть специальный файл;

`-hn, --max-back-scroll=n` — задать максимальное количество строк для прокрутки назад;

`-yn, --max-forw-scroll=n` — задать максимальное количество строк для прокрутки вперёд;

`-i, --ignore-case` — игнорировать регистр;

`-I, --IGNORE-CASE` — игнорировать регистр, даже если паттерн для поиска содержит заглавные буквы;

`-jn, --jump-target=n` — указать, в какой строке должна быть выведена искомая информация;

-J, --status-column — пометить строки, соответствующие результатам поиска;

-n, --line-numbers — не выводить номера строк;

-N, --LINE-NUMBERS — вывести номера строк;

-s, --squeeze-blank-lines — заменить множество идущих подряд пустых строк одной пустой строкой;

-w, --hilite-unread — выделить первую строку нового фрагмента текста.

Во время просмотра текста утилитой можно управлять при помощи внутренних команд, набирая их на клавиатуре компьютера. Наиболее часто используемые из них:

h, H — справка;

Space, Ctrl+V, f, Ctrl+F — прокрутить текст на один экран вперёд;

Enter, Return, Ctrl+N, e, Ctrl+E, j, Ctrl+J — прокрутить текст на n строк вперед, по умолчанию n=1;

у, Ctrl+Y, Ctrl+P, k, Ctrl+K — прокрутить текст на n строк назад, по умолчанию n=1;

Ctrl+→ — прокрутить текст по горизонтали в конец строки;

Ctrl+← — прокрутить текст по горизонтали в начало строки;

:d — удалить текущий файл из списка файлов;

Ctrl+G, :f — вывести основную информацию о файле;

q, Q, :q, :Q, ZZ — выход.

Перечень всех опций и внутренних команд можно просмотреть в терминале, выполнив команду `man less`

```
nuviky@nuviky:~$ less test.txt
fsd
vxd
fgs
drydrtyu45

sdfr
gwerytg3
456v
345tertgvdrth
vdrf
ghvdfgh
dfghv
dfgvhdfghvd
fghv
dfg
hvd
fg
hvdr
th
ertuy
e456
745
6u
546nuby6u.j
rety
hvds
rth
vert
hvrt
uj
test.txt (END)_
```

Рисунок 5 – Пример выполнения команды less

Команда `grep` (расшифровывается как `global regular expression print`) - одна из самых востребованных команд в терминале Linux, которая входит в состав проекта GNU. Секрет популярности - её мощь, она даёт возможность пользователям сортировать и фильтровать текст на основе сложных правил.

Утилита `grep` решает множество задач, в основном она используется для поиска строк, соответствующих строке в тексте или содержимому файлов. Также она может находить по шаблону или регулярным выражениям. Команда в считанные секунды найдёт файл с нужной строчкой, текст в файле или отфильтрует из вывода только пару нужных строк.

Синтаксис команды `grep` выглядит следующим образом:

`$ grep [опции] шаблон [имя файла...]`

Или:

\$ команда | grep [опции] шаблон

Опции - это дополнительные параметры, с помощью которых указываются различные настройки поиска и вывода, например количество строк или режим инверсии.

Шаблон - это любая строка или регулярное выражение, по которому будет вестись поиск

Файл и команда - это то место, где будет вестись поиск. Как вы увидите дальше, grep позволяет искать в нескольких файлах и даже в каталоге, используя рекурсивный режим.

Возможность фильтровать стандартный вывод пригодится, например, когда нужно выбрать только ошибки из логов или найти PID процесса в многочисленном отчёте утилиты ps.

```
nuviky@nuviky:~$ grep -A4 "fg" test.txt
fgs
drydrtyu45

sdfr
gwerytg3
--
ghvdfgh
dfghv
dfgvhdfghvd
fghv
dfg
hvd
fg
hvdr
th
ertuy
e456
nuviky@nuviky:~$
```

Рисунок 6 – Пример выполнения команды grep

Find - это одна из наиболее важных и часто используемых утилит системы Linux. Это команда для поиска файлов и каталогов на основе специальных условий. Ее можно использовать в различных обстоятельствах, например, для поиска файлов по разрешениям, владельцам, группам, типу, размеру и другим подобным критериям.

Утилита `find` предустановлена по умолчанию во всех Linux дистрибутивах, поэтому вам не нужно будет устанавливать никаких дополнительных пакетов. Это очень важная находка для тех, кто хочет использовать командную строку наиболее эффективно.

Команда `find` имеет такой синтаксис:

`find [папка] [параметры] критерий шаблон [действие]`

Папка - каталог в котором будем искать

Параметры - дополнительные параметры, например, глубина поиска, и т.д.

Критерий - по какому критерию будем искать: имя, дата создания, права, владелец и т.д.

Шаблон – непосредственно значение по которому будем отбирать файлы.

Наиболее популярные параметры:

- P никогда не открывать символические ссылки
- L - получает информацию о файлах по символическим ссылкам. Важно для дальнейшей обработки, чтобы обрабатывалась не ссылка, а сам файл.
- maxdepth - максимальная глубина поиска по подкаталогам, для поиска только в текущем каталоге установите 1.
- depth - искать сначала в текущем каталоге, а потом в подкаталогах
- mount искать файлы только в этой файловой системе.
- version - показать версию утилиты `find`
- print - выводить полные имена файлов
- type f - искать только файлы
- type d - поиск папки в Linux

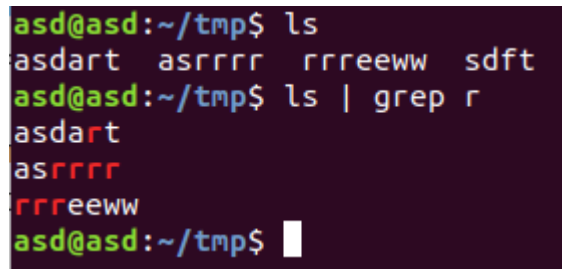
```
nuviky@nuviky:~$ find test.txt
test.txt
nuviky@nuviky:~$ _
```

Рисунок 7 – Пример выполнения команды `find`

2 Задание 2

Конвейер в Linux служит для передачи выходных данных из одной программы в другую как входные данные. Т.е. выполняется команда, мы получаем результат и передаем эти данные далее на обработку другой программе.

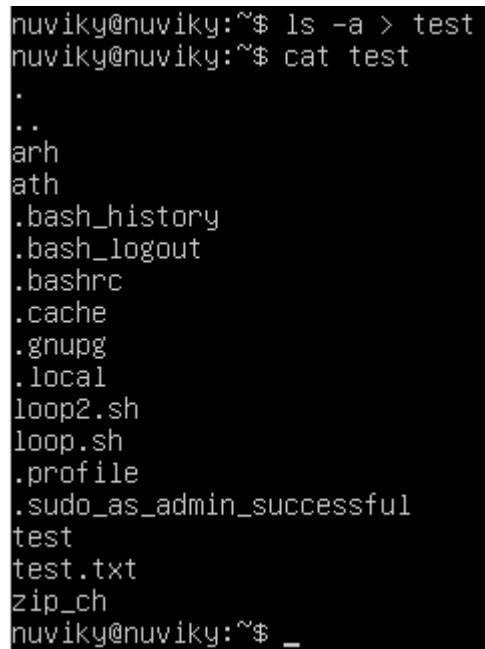
Например, если выполнить команду `ls | grep r`, то выходные данные команды `ls` передаем как входные данные для команды `grep r` (поиск по букве `r`) и в результате мы получаем файлы, в которых есть буква `r`. (Рисунок 8)



```
asd@asd:~/tmp$ ls
asdart  asrrrr  rrreeww  sdft
asd@asd:~/tmp$ ls | grep r
asdart
asrrrr
rrreeww
asd@asd:~/tmp$
```

Рисунок 8 – Пример использования конвейера

Перенаправление ввода-вывода осуществляет с помощью символов “<” и “>”. Например перенаправим вывод команды `ls -a` в файл и воспользуемся командой `ls -a > test`, результат показан на рисунке 9.



```
nuviky@nuviky:~$ ls -a > test
nuviky@nuviky:~$ cat test
.
..
arh
ath
.bash_history
.bash_logout
.bashrc
.cache
.gnupg
.local
loop2.sh
loop.sh
.profile
.sudo_as_admin_successful
test
test.txt
zip_ch
nuviky@nuviky:~$ _
```

Рисунок 9 – Пример перенаправления вывода в файл

Для примера перенаправления ввода воспользуемся командой `sort < test.txt`, в данном примере данные из файла `test.txt` сортируются, результат показан на рисунке 10.

```

nuviky@nuviky:~$ sort < test.txt
345tertgvdrth
456v
546nuby6uj
6u
745
dfg
dfghv
dfgvhdfghvd
drydrtyu45
e456
ertuy
fg
fghv
fgs
fsd
ghvdfgh
gwerytg3
hvd
hvdr
hvds
hvrt
rety
rth
sdfr
th
vdrf
vert
vxd
yj
nuviky@nuviky:~$ _

```

Рисунок 10 – Пример перенаправления ввода

3 Задание 3

Команда `chmod` используется для изменения прав доступа к файлам или каталогам. Создадим новый файл с правами доступа `root` и попробуем его открыть с помощью команды `cat`. Результат представлен на рисунке 11.

```

asd@asd:~/tmp$ chmod ugo-rwx testpr.txt
asd@asd:~/tmp$ cat testpr.txt
cat: testpr.txt: Отказано в доступе
asd@asd:~/tmp$

```

Рисунок 11 – Создание и открытие файла с правами доступа `root`

Из рисунка 11 видно, что для открытия файла у пользователя нет прав, исправим данную ситуацию командой `sudo chmod ugo+rwx testpr.txt`, которая выдаст права на запись. Результат приведен на рисунке 12.

```

asd@asd:~/tmp$ chmod ugo+rwx testpr.txt
asd@asd:~/tmp$

```

Рисунок 12 – Изменение прав файла

Команда `chown` позволяет изменять владельца или группу у папок и файлов. Изменим владельца файла `testpr.txt` с помощью команды `sudo chown nuviky testpr.txt`, результат показан на рисунке 13.

```
asd@asd:~/tmp$ sudo su
root@asd:/home/asd/tmp# chown nuviky testpr.txt
root@asd:/home/asd/tmp#
```

Рисунок 13 – Изменение владельца файла

4 Задание 4

Выведем все запущенные в данный момент процессы с помощью команды `top`, результат показан на рисунке 14.

```
top - 20:14:09 up 3:22, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 139 total, 1 running, 71 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4038488 total, 2663652 free, 141680 used, 1233156 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 3628856 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10831	root	20	0	0	0	0	I	0.7	0.0	0:01.28	kworker/2:3
1	root	20	0	78044	9256	6792	S	0.0	0.2	0:04.22	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.04	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:01.98	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.10	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.12	watchdog/1
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/1
16	root	20	0	0	0	0	S	0.0	0.0	0:00.55	ksoftirqd/1
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.10	watchdog/2
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/2
22	root	20	0	0	0	0	S	0.0	0.0	0:00.03	ksoftirqd/2
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/3
26	root	rt	0	0	0	0	S	0.0	0.0	0:00.13	watchdog/3
27	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/3
28	root	20	0	0	0	0	S	0.0	0.0	0:00.51	ksoftirqd/3
30	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/3:0H
31	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/4
32	root	rt	0	0	0	0	S	0.0	0.0	0:00.17	watchdog/4
33	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/4
34	root	20	0	0	0	0	S	0.0	0.0	0:00.07	ksoftirqd/4

Рисунок 14 – Все запущенные в данный момент процессы

Отсортируем процессы по полю `TIME+`, для этого нажмем комбинацию клавиш `Shift+f` и выберем из списка `TIME+`. Таблица с отсортированными процессами показана на рисунке 15.


```
top - 20:21:10 up 3:29, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 139 total, 1 running, 71 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4038488 total, 2663140 free, 142080 used, 1233268 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 3628408 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10544	root	20	0	0	0	0	I	0.0	0.0	0:07.02	kworker/5:5
1	root	20	0	78044	9256	6792	S	0.0	0.2	0:04.22	systemd
10916	root	20	0	0	0	0	I	0.0	0.0	0:03.86	kworker/4:5
40	root	20	0	0	0	0	S	0.0	0.0	0:03.07	ksoftirqd/5
10831	root	20	0	0	0	0	I	0.0	0.0	0:03.02	kworker/2:3
169	root	20	0	0	0	0	I	0.0	0.0	0:02.56	kworker/3:2
429	root	20	0	46700	5676	3248	S	0.0	0.1	0:02.28	systemd-udev
8	root	20	0	0	0	0	I	0.0	0.0	0:02.00	rcu_sched
48	root	20	0	0	0	0	I	0.0	0.0	0:01.72	kworker/1:1
10599	root	20	0	0	0	0	I	0.7	0.0	0:01.05	kworker/0:6
77	root	20	0	0	0	0	I	0.0	0.0	0:00.71	kworker/4:1
76	root	20	0	0	0	0	I	0.0	0.0	0:00.67	kworker/3:1
16	root	20	0	0	0	0	S	0.0	0.0	0:00.56	ksoftirqd/1
410	root	0	-20	0	0	0	I	0.0	0.0	0:00.55	kworker/1:1H
28	root	20	0	0	0	0	S	0.0	0.0	0:00.51	ksoftirqd/3
405	root	19	-1	94820	12596	11888	S	0.0	0.3	0:00.50	systemd-journal
906	message+	20	0	50288	4872	4028	S	0.0	0.1	0:00.42	dbus-daemon
1005	root	20	0	70724	6268	5492	S	0.0	0.2	0:00.39	systemd-logind
334	root	20	0	0	0	0	S	0.0	0.0	0:00.37	jbd2/dm-0-8
886	root	20	0	110416	1940	1724	S	0.0	0.0	0:00.30	irqbalance
10601	root	20	0	286244	7036	6156	S	0.0	0.2	0:00.28	accounts-daemon
39	root	rt	0	0	0	0	S	0.0	0.0	0:00.26	migration/5
27	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/3
33	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/4
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/1
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/2
22077	root	20	0	0	0	0	I	0.0	0.0	0:00.24	kworker/u12:1
214	root	0	-20	0	0	0	I	0.0	0.0	0:00.23	kworker/3:1H
21830	nuviky	20	0	21472	5244	3536	S	0.0	0.1	0:00.22	bash
22076	root	20	0	0	0	0	I	0.0	0.0	0:00.19	kworker/u12:2

Рисунок 15 – Отсортированные процессы по полю TIME+

Выведем все процессы пользователя nuviky, для этого воспользуемся командой `top -u nuviky`, результат показан на рисунке 16.

```
top - 20:19:27 up 3:27, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 139 total, 1 running, 71 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.2 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4038488 total, 2662668 free, 142584 used, 1233236 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 3627916 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
21804	nuviky	20	0	76692	7104	6188	S	0.0	0.2	0:00.01	systemd
21811	nuviky	20	0	112024	2692	72	S	0.0	0.1	0:00.00	(sd-pam)
21830	nuviky	20	0	21472	5244	3536	S	0.0	0.1	0:00.22	bash
21989	nuviky	20	0	13136	1096	1000	T	0.0	0.0	0:00.00	grep
21990	nuviky	20	0	13136	1012	916	T	0.0	0.0	0:00.00	grep
21991	nuviky	20	0	13136	1092	1000	T	0.0	0.0	0:00.00	grep
22097	nuviky	20	0	42804	4104	3464	R	0.0	0.1	0:00.00	top

Рисунок 16 – Все процессы пользователя nuviky

Для выделения процессов, которые выполняются в данный момент, необходимо во время выполнения команды `top` нажать на кнопку `z`, результат показан на рисунке 17.

```
top - 20:25:35 up 3:33, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 139 total, 1 running, 71 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4038488 total, 2663140 free, 142072 used, 1233276 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 3628416 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10544	root	20	0	0	0	0	I	0.0	0.0	0:07.03	kworker/5:5
1	root	20	0	78044	9256	6792	S	0.0	0.2	0:04.22	systemd
10916	root	20	0	0	0	0	I	0.0	0.0	0:03.93	kworker/4:5
40	root	20	0	0	0	0	S	0.0	0.0	0:03.07	ksoftirqd/5
10831	root	20	0	0	0	0	I	0.0	0.0	0:03.02	kworker/2:3
169	root	20	0	0	0	0	I	0.0	0.0	0:02.60	kworker/3:2
48	root	20	0	0	0	0	I	1.0	0.0	0:02.29	kworker/1:1
429	root	20	0	46700	5676	3248	S	0.0	0.1	0:02.28	systemd-udev
8	root	20	0	0	0	0	I	0.0	0.0	0:02.01	rcu_sched
10599	root	20	0	0	0	0	I	0.0	0.0	0:01.56	kworker/0:6
77	root	20	0	0	0	0	I	0.0	0.0	0:00.71	kworker/4:1
76	root	20	0	0	0	0	I	0.0	0.0	0:00.67	kworker/3:1
16	root	20	0	0	0	0	S	0.0	0.0	0:00.56	ksoftirqd/1
410	root	0	-20	0	0	0	I	0.0	0.0	0:00.55	kworker/1:1H
28	root	20	0	0	0	0	S	0.0	0.0	0:00.51	ksoftirqd/3
405	root	19	-1	94820	12596	11888	S	0.0	0.3	0:00.50	systemd-journal
906	message+	20	0	50288	4872	4028	S	0.0	0.1	0:00.42	dbus-daemon
1005	root	20	0	70724	6268	5492	S	0.0	0.2	0:00.39	systemd-logind
334	root	20	0	0	0	0	S	0.0	0.0	0:00.37	jbd2/dm-0-8
886	root	20	0	110416	1940	1724	S	0.0	0.0	0:00.32	irqbalance
22077	root	20	0	0	0	0	I	0.3	0.0	0:00.30	kworker/u12:1
10601	root	20	0	286244	7036	6156	S	0.0	0.2	0:00.29	accounts-daemon
39	root	rt	0	0	0	0	S	0.0	0.0	0:00.26	migration/5
22099	nuviku	20	0	42804	4032	3392	R	0.3	0.1	0:00.26	top
27	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/3
33	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/4
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/1
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/2
22076	root	20	0	0	0	0	I	0.0	0.0	0:00.24	kworker/u12:2
214	root	0	-20	0	0	0	I	0.0	0.0	0:00.23	kworker/3:1H

Рисунок 17 – Процессы, которые выполняются в данный момент

Для отображения абсолютных путей запущенных процессов, необходимо во время выполнения команды `top` нажать на кнопку `s`, результат показан на рисунке 18.

```
top - 20:27:23 up 3:35, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 139 total, 2 running, 71 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4038488 total, 2663264 free, 141948 used, 1233276 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 3628540 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10544	root	20	0	0	0	0	I	0.0	0.0	0:07.03	[kworker/5:5]
1	root	20	0	78044	9256	6792	S	0.0	0.2	0:04.22	/sbin/init maybe-ubiquity
10916	root	20	0	0	0	0	I	0.0	0.0	0:03.96	[kworker/4:5]
10831	root	20	0	0	0	0	R	1.1	0.0	0:03.53	[kworker/2:3]
40	root	20	0	0	0	0	S	0.0	0.0	0:03.07	[ksoftirqd/5]
169	root	20	0	0	0	0	I	0.0	0.0	0:02.62	[kworker/3:2]
48	root	20	0	0	0	0	I	0.0	0.0	0:02.29	[kworker/1:1]
429	root	20	0	46700	5676	3248	S	0.0	0.1	0:02.28	/lib/systemd/systemd-udev
8	root	20	0	0	0	0	I	0.0	0.0	0:02.02	[rcu_sched]
10599	root	20	0	0	0	0	I	0.0	0.0	0:01.56	[kworker/0:6]
77	root	20	0	0	0	0	I	0.0	0.0	0:00.71	[kworker/4:1]
76	root	20	0	0	0	0	I	0.0	0.0	0:00.67	[kworker/3:1]
16	root	20	0	0	0	0	S	0.0	0.0	0:00.56	[ksoftirqd/1]
410	root	0	-20	0	0	0	I	0.0	0.0	0:00.55	[kworker/1:1H]
28	root	20	0	0	0	0	S	0.0	0.0	0:00.51	[ksoftirqd/3]
405	root	19	-1	94820	12596	11888	S	0.0	0.3	0:00.50	/lib/systemd/systemd-journald
906	message+	20	0	50288	4872	4028	S	0.0	0.1	0:00.42	/usr/bin/dbus-daemon --system +
1005	root	20	0	70724	6268	5492	S	0.0	0.2	0:00.39	/lib/systemd/systemd-logind
334	root	20	0	0	0	0	S	0.0	0.0	0:00.37	[jbd2/dm-0-8]
22099	nuviky	20	0	42832	4032	3392	R	0.0	0.1	0:00.36	top
22077	root	20	0	0	0	0	I	0.0	0.0	0:00.33	[kworker/u12:1]
886	root	20	0	110416	1940	1724	S	0.0	0.0	0:00.32	/usr/sbin/irqbalance --foregro+
10601	root	20	0	286244	7036	6156	S	0.0	0.2	0:00.29	/usr/lib/accountsservice/accou+
39	root	rt	0	0	0	0	S	0.0	0.0	0:00.26	[migration/5]
27	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	[migration/3]
33	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	[migration/4]
22076	root	20	0	0	0	0	I	0.0	0.0	0:00.25	[kworker/u12:2]
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	[migration/1]
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	[migration/2]
214	root	0	-20	0	0	0	I	0.0	0.0	0:00.23	[kworker/3:1H]

Рисунок 18 – Абсолютные пути запущенных процессов

Для сортировки процессов по нагрузке на процессор, необходимо во время выполнения команды `top` нажать на сочетание кнопок `Shift+p`, результат показан на рисунке 19.

```
top - 21:07:10 up 4:15, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 139 total, 1 running, 71 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4038488 total, 2663380 free, 141824 used, 1233284 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 3628668 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10544	root	20	0	0	0	0	I	0.7	0.0	0:07.10	kworker/5:5
22107	nuviky	20	0	42804	4180	3540	R	0.3	0.1	0:00.71	top
1	root	20	0	78044	9256	6792	S	0.0	0.2	0:04.24	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.04	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.03	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:02.15	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.13	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.17	watchdog/1
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/1
16	root	20	0	0	0	0	S	0.0	0.0	0:00.56	ksoftirqd/1
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.13	watchdog/2
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/2
22	root	20	0	0	0	0	S	0.0	0.0	0:00.05	ksoftirqd/2
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/3
26	root	rt	0	0	0	0	S	0.0	0.0	0:00.17	watchdog/3
27	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/3
28	root	20	0	0	0	0	S	0.0	0.0	0:00.56	ksoftirqd/3
30	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/3:0H
31	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/4
32	root	rt	0	0	0	0	S	0.0	0.0	0:00.22	watchdog/4
33	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/4

Рисунок 19 – Сортировка процессов по нагрузке на процессор

Установим supervisor с помощью команды `sudo apt install supervisor`.

Процесс установки представлен на рисунке 20.

```
nuviky@nuviky:~$ sudo apt install supervisor
[sudo] password for nuviky:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-meld3 python-minimal
  python-pkg-resources python2.7 python2.7-minimal
Suggested packages:
  python-doc python-tk python-setuptools python2.7-doc binutils binfmt-support supervisor-doc
The following NEW packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-meld3 python-minimal
  python-pkg-resources python2.7 python2.7-minimal supervisor
0 upgraded, 10 newly installed, 0 to remove and 39 not upgraded.
Need to get 4,380 kB of archives.
After this operation, 19.0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Рисунок 20 – Установка supervisor

Произведем настройку supervisor, файл с конфигурацией находится в `/etc/supervisor/supervisord.conf`. Параметры находятся на рисунке 21.

```
[supervisorctl]
serverurl=unix:///var/run/supervisor.sock ; use a unix:// URL  for a unix socket

; The [include] section can just contain the "files" setting.  This
; setting can list multiple files (separated by whitespace or
; newlines).  It can also contain wildcards.  The filenames are
; interpreted as relative to this file.  Included files *cannot*
; include files themselves.

[include]
files = /etc/supervisor/conf.d/*.conf

[program:test]
command=sh /home/nuviky/loop2.sh
stdout_logfile=/home/nuviky/lg.log
autostart=true
autorestart=true
user=nuviky
stopsignal=KILL
```

Рисунок 21 – Настройка supervisor

Перезапустим supervisor для запуска процесса с помощью команды `/etc/init.d/supervisor restart` и проверим содержание файл логгирования. Шаги продемонстрированные на рисунке 22.

```
nuviky@nuviky:~$ /etc/init.d/supervisor restart
[....] Restarting supervisor (via systemctl): supervisor.service==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to restart 'supervisor.service'.
Authenticating as: mikhail (nuviky)
Password:
==== AUTHENTICATION COMPLETE ====
[ 366.374040] EXT4-fs error (device dm-0): ext4_find_entry:1466: inode #138484: comm supervisorctl: checksumming directory block 0
[ 366.376164] EXT4-fs error (device dm-0): ext4_find_entry:1466: inode #138484: comm supervisorctl: checksumming directory block 0
[ 366.378648] EXT4-fs error (device dm-0): ext4_find_entry:1466: inode #138484: comm supervisorctl: checksumming directory block 0
[ 366.466963] EXT4-fs error (device dm-0): ext4_find_entry:1466: inode #138484: comm supervisor: checksumming directory block 0
[ 366.469732] EXT4-fs error (device dm-0): ext4_find_entry:1466: inode #138484: comm supervisor: checksumming directory block 0
[ 366.473978] EXT4-fs error (device dm-0): ext4_find_entry:1466: inode #138484: comm supervisor: checksumming directory block 0
. ok
nuviky@nuviky:~$ cat lg.log
Hello
nuviky@nuviky:~$ cat lg.log
Hello
Hello
Hello
Hello
nuviky@nuviky:~$
```

Рисунок 22 – Запуск процесса с помощью supervisor

4 Задание 4

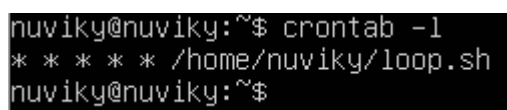
Для запуска программ по расписанию используется команда `cron`. Она предназначена для выполнения заданий в определенное время, или через определенные промежутки времени. Для редактирования заданий используется утилита `crontab`.

Чтобы создать расписание воспользуемся командой `crontab -e`, в открывшемся редакторе запишем расписание для запуска скрипта каждую минуту и проверим изменения с помощью команды `crontab -l`. Прделанные шаги показаны на рисунка 24 и 25.



```
GNU nano 2.9.3 /tmp/crontab.qx0brR/crontab
* * * * * /home/nuviky/loop.sh
```

Рисунок 23 – Изменение файла расписания с помощью nano



```
nuviky@nuviky:~$ crontab -l
* * * * * /home/nuviky/loop.sh
nuviky@nuviky:~$
```

Рисунок 24– Проверка текущего файла расписания

Вывод

В ходе выполнения лабораторной работы я ознакомился на практике с понятием процесса в операционной системе, приобрел опыт и навыки управления процессами в операционной системе Linux.