

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Отчет по лабораторной работе № 3

«Управление процессами ОС Ubuntu»

по курсу «ОС Linux»

Студент
Группа АИ-18

Грунау Г. Ю.

Руководитель

Кургасов В. В.

Липецк 2020 г.

Содержание

Выполнение работы	3
1. Повторить команды cat, head, tail, more, less, grep, find	3
2. Разобраться с понятиями конвейер, перенаправление ввода-вывода.	6
3. Ознакомиться с информацией из рекомендованных источников и других про конвейеризации	8
4. Повторить назначение прав доступа. Команды chmod, chown.....	9
5. Ознакомиться с информацией по теме процессы, посмотреть и опробовать примеры наиболее распространенных команд, изучить возможность запуска процессов в supervisor	11
6. Запуск процессов по расписанию.	19
Вывод	20

Выполнение работы

1. Повторить команды cat, head, tail, more, less, grep, find

```
lovediehate@myubuntuserver:~/lr2$ cat loop
while true; do true; done
lovediehate@myubuntuserver:~/lr2$ cat loop2
while true; do true; echo 'Hello'; done
lovediehate@myubuntuserver:~/lr2$ _
```

Рисунок 1 - Использование команды cat

cat — одна из наиболее часто используемых команд в Linux. Она считывает данные из файлов и выводит их содержимое. Это самый простой способ отображения содержимого файла в командной строке.

```
lovediehate@myubuntuserver:~/lr2$ cat text.txt
qqqqqqqqqqqqqqqq
wwwwwwwwwwwwwwww
eeeeeeeeeeeeeeee
rrrrrrrrrrrrrrrr
ttttttttttttttt
yyyyyyyyyyyyyyyy
uuuuuuuuuuuuuuu
iiiiiiiiiiiiiii
oooooooooooooooo
ppppppppppppppp
aaaaaaaaaaaaaaaaa
sssssssssssssss
ddddddddddddddd
ffffffffffffff
lovediehate@myubuntuserver:~/lr2$ head -n2 text.txt
qqqqqqqqqqqqqqqq
wwwwwwwwwwwwwwww
```

Рисунок 2 - Использование команды head

```
lovediehate@myubuntuserver:~/lr2$ tail -n3 text.txt
sssssssssssssss
ddddddddddddddd
ffffffffffffff
```

Рисунок 3 - Использование команды tail

Введена команда `ps -ef | more`

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	17:56	?	00:00:02	/sbin/init maybe-ubiquity
root	2	0	0	17:56	?	00:00:00	[kthreadd]
root	3	2	0	17:56	?	00:00:00	[rcu_gp]
root	4	2	0	17:56	?	00:00:00	[rcu_par_gp]
root	6	2	0	17:56	?	00:00:00	[kworker/0:0H-kblockd]
root	8	2	0	17:56	?	00:00:00	[kworker/u2:0-events_power_efficient]
root	9	2	0	17:56	?	00:00:00	[mm_percpu_wq]
root	10	2	0	17:56	?	00:00:00	[ksoftirqd/0]
root	11	2	0	17:56	?	00:00:00	[rcu_sched]
root	12	2	0	17:56	?	00:00:00	[migration/0]
root	13	2	0	17:56	?	00:00:00	[idle_inject/0]
root	14	2	0	17:56	?	00:00:00	[cpuhp/0]
root	15	2	0	17:56	?	00:00:00	[kdevtmpfs]
root	16	2	0	17:56	?	00:00:00	[netns]
root	17	2	0	17:56	?	00:00:00	[rcu_tasks_kthre]
root	18	2	0	17:56	?	00:00:00	[kauditd]
root	19	2	0	17:56	?	00:00:00	[khungtaskd]
root	20	2	0	17:56	?	00:00:00	[oom_reaper]
root	21	2	0	17:56	?	00:00:00	[writeback]
root	22	2	0	17:56	?	00:00:00	[kcompactd0]
root	23	2	0	17:56	?	00:00:00	[ksmd]
root	24	2	0	17:56	?	00:00:00	[khugepaged]
root	70	2	0	17:56	?	00:00:00	[kintegrityd]
root	71	2	0	17:56	?	00:00:00	[kblockd]
root	72	2	0	17:56	?	00:00:00	[blkcg_punt_bio]
root	73	2	0	17:56	?	00:00:00	[tpm_dev_wq]
root	74	2	0	17:56	?	00:00:00	[ata_sff]
root	75	2	0	17:56	?	00:00:00	[md]
root	76	2	0	17:56	?	00:00:00	[edac-poller]
root	77	2	0	17:56	?	00:00:00	[devfreq_wq]
root	78	2	0	17:56	?	00:00:00	[watchdogd]
root	83	2	0	17:56	?	00:00:00	[kswapd0]
root	84	2	0	17:56	?	00:00:00	[ecryptfs-kthrea]
root	86	2	0	17:56	?	00:00:00	[kthrotld]
root	87	2	0	17:56	?	00:00:00	[acpi_thermal_pm]
--More--							

Рисунок 4 - Использование `more`

Команда `more` позволяет выводить изображение в терминале на одной странице одновременно

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	17:56	?	00:00:02	/sbin/init maybe-ubiquity
root	2	0	0	17:56	?	00:00:00	[kthreadd]
root	3	2	0	17:56	?	00:00:00	[rcu_gp]
root	4	2	0	17:56	?	00:00:00	[rcu_par_gp]
root	6	2	0	17:56	?	00:00:00	[kworker/0:0H-kblockd]
root	8	2	0	17:56	?	00:00:00	[kworker/u2:0-events_power_efficient]
root	9	2	0	17:56	?	00:00:00	[mm_percpu_wq]
root	10	2	0	17:56	?	00:00:00	[ksoftirqd/0]
root	11	2	0	17:56	?	00:00:00	[rcu_sched]
root	12	2	0	17:56	?	00:00:00	[migration/0]
root	13	2	0	17:56	?	00:00:00	[idle_inject/0]
root	14	2	0	17:56	?	00:00:00	[cpuhp/0]
root	15	2	0	17:56	?	00:00:00	[kdevtmpfs]
root	16	2	0	17:56	?	00:00:00	[netns]
root	17	2	0	17:56	?	00:00:00	[rcu_tasks_kthre]
root	18	2	0	17:56	?	00:00:00	[kauditd]
root	19	2	0	17:56	?	00:00:00	[khungtaskd]
root	20	2	0	17:56	?	00:00:00	[oom_reaper]
root	21	2	0	17:56	?	00:00:00	[writeback]
root	22	2	0	17:56	?	00:00:00	[kcompactd0]
root	23	2	0	17:56	?	00:00:00	[ksmd]
root	24	2	0	17:56	?	00:00:00	[khugepaged]
root	70	2	0	17:56	?	00:00:00	[kintegrityd]
root	71	2	0	17:56	?	00:00:00	[kblockd]
root	72	2	0	17:56	?	00:00:00	[blkcg_punt_bio]
root	73	2	0	17:56	?	00:00:00	[tpm_dev_wq]
root	74	2	0	17:56	?	00:00:00	[ata_sff]
root	75	2	0	17:56	?	00:00:00	[md]
root	76	2	0	17:56	?	00:00:00	[edac-poller]
root	77	2	0	17:56	?	00:00:00	[devfreq_wq]
root	78	2	0	17:56	?	00:00:00	[watchdogd]
root	83	2	0	17:56	?	00:00:00	[kswapd0]
root	84	2	0	17:56	?	00:00:00	[ecryptfs-kthrea]
root	86	2	0	17:56	?	00:00:00	[kthrotld]
root	87	2	0	17:56	?	00:00:00	[acpi_thermal_pm]

Рисунок 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, а также искать как по направлению к концу, так и по направлению к началу текста

```
lovediehatemyubuntuserver:~/lr2$ ps -l
F S  UID      PID      PPID    C  PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000      937      649    0   80   0  -  1767 do_wai tty1        00:00:00 bash
0 T  1000     1091      937    0   80   0  -  1155 do_sig tty1        00:00:00 less
0 R  1000     1095      937    0   80   0  -  1888 -      tty1        00:00:00 ps
lovediehatemyubuntuserver:~/lr2$ ps -l | grep -v 'bash'
F S  UID      PID      PPID    C  PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 T  1000     1091      937    0   80   0  -  1155 do_sig tty1        00:00:00 less
0 R  1000     1096      937    0   80   0  -  1888 -      tty1        00:00:00 ps
0 S  1000     1097      937    0   80   0  -  1298 pipe_w tty1        00:00:00 grep
```

Рисунок 6 - Команда grep

Grep это утилита командной строки Linux, который даёт пользователям возможность вести поиск строки

```
lovediegate@myubuntuserver:~$ find / -name "vmlinuz*" 2>/dev/null
/boot/vmlinuz.old
/boot/vmlinuz
/boot/vmlinuz-5.4.0-48-generic
lovediegate@myubuntuserver:~$ _
```

Рисунок 7 - Команда find

Команда `find` представляет собой универсальный инструмент поиска: она позволяет искать файлы и каталоги, просматривать все каталоги в системе или только текущий каталог.

2. Разобраться с понятиями конвейер, перенаправление ввода-вывода.

```
lovediegate@myubuntuserver:~/lr2$ ls -l > text.txt
lovediegate@myubuntuserver:~/lr2$ cat < text.txt
total 16
drwxrwxr-x 2 lovediegate lovediegate 4096 Oct 29 19:15 dir
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 20:14 fif
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:22 filen.tar.gz
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:20 filename
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:42 foo.in
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:41 foo.out
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:43 foo.pipe
-rw-rw-r-- 1 lovediegate lovediegate 26 Oct 29 12:31 loop
-rw-rw-r-- 1 lovediegate lovediegate 40 Oct 29 12:36 loop2
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:55 my-file-pipe
-rw-rw-r-- 1 lovediegate lovediegate 20 Oct 29 19:53 out
-rw-rw-r-- 1 lovediegate lovediegate 0 Nov 12 18:45 text.txt
lovediegate@myubuntuserver:~/lr2$ cat >> text.txt
NEW STRING NEW STRING NEW STRING
^C
lovediegate@myubuntuserver:~/lr2$ cat < text.txt
total 16
drwxrwxr-x 2 lovediegate lovediegate 4096 Oct 29 19:15 dir
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 20:14 fif
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:22 filen.tar.gz
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:20 filename
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:42 foo.in
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:41 foo.out
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:43 foo.pipe
-rw-rw-r-- 1 lovediegate lovediegate 26 Oct 29 12:31 loop
-rw-rw-r-- 1 lovediegate lovediegate 40 Oct 29 12:36 loop2
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:55 my-file-pipe
-rw-rw-r-- 1 lovediegate lovediegate 20 Oct 29 19:53 out
-rw-rw-r-- 1 lovediegate lovediegate 0 Nov 12 18:45 text.txt
NEW STRING NEW STRING NEW STRING
```

Рисунок 8 - Перенаправление in out

Символ «|» на Рисунок 6 — это и есть конвейер. Его можно понимать как канал, в который один процесс может только писать, а другой — только читать из него. Выборка и помещение информации в такой канал происходит в порядке FIFO.

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.

3. Ознакомиться с информацией из рекомендованных источников и других про конвейеризации

Конвейеры — это возможность нескольких программ работать совместно, когда выход одной программы непосредственно идет на вход другой без использования промежуточных временных файлов. Синтаксис: команда1 | команда2, выполняет команду1 используя её поток вывода как поток ввода при выполнении команды2, что равносильно использованию двух перенаправлений и временного файла:

```
команда1 > ВременныйФайл  
команда2 < ВременныйФайл  
rm ВременныйФайл
```

Хороший пример командных конвейеров — это объединение `echo` с другой командой для получения интерактивности в неинтерактивных средах, к примеру:

```
echo -e "ИмяПользователя\nПароль" | ftp localhost
```

Конвейер (англ. pipeline) в терминологии операционных систем семейства Unix — некоторое множество процессов, для которых выполнено следующее перенаправление ввода-вывода: то, что выводит на поток стандартного вывода предыдущий процесс, попадает в поток стандартного ввода следующего процесса. Запуск конвейера реализован с помощью системного вызова `pipe()`.

4. Повторить назначение прав доступа. Команды chmod, chown

```
drwxrwxr-x 3 lovediegate lovediegate 4096 Nov 12 18:06 .
drwxr-xr-x 5 lovediegate lovediegate 4096 Nov 12 18:06 ..
drwxrwxr-x 2 lovediegate lovediegate 4096 Oct 29 19:15 dir
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 20:14 fif
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:22 filen.tar.gz
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:20 filename
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:42 foo.in
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:41 foo.out
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:43 foo.pipe
-rw-rw-r-- 1 lovediegate lovediegate 26 Oct 29 12:31 loop
-rw-rw-r-- 1 lovediegate lovediegate 40 Oct 29 12:36 loop2
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:55 my-file-pipe
-rw-rw-r-- 1 lovediegate lovediegate 20 Oct 29 19:53 out
-rw-rw-r-- 1 lovediegate lovediegate 793 Nov 12 18:45 text.txt
lovediegate@myubuntuserver:~/lr2$ chmod ugo+rwx text.txt
lovediegate@myubuntuserver:~/lr2$ ls -al
total 28
drwxrwxr-x 3 lovediegate lovediegate 4096 Nov 12 18:06 .
drwxr-xr-x 5 lovediegate lovediegate 4096 Nov 12 18:06 ..
drwxrwxr-x 2 lovediegate lovediegate 4096 Oct 29 19:15 dir
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 20:14 fif
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:22 filen.tar.gz
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:20 filename
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:42 foo.in
-rw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:41 foo.out
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:43 foo.pipe
-rw-rw-r-- 1 lovediegate lovediegate 26 Oct 29 12:31 loop
-rw-rw-r-- 1 lovediegate lovediegate 40 Oct 29 12:36 loop2
prw-rw-r-- 1 lovediegate lovediegate 0 Oct 29 19:55 my-file-pipe
-rw-rw-r-- 1 lovediegate lovediegate 20 Oct 29 19:53 out
-rwxrwxrwx 1 lovediegate lovediegate 793 Nov 12 18:45 text.txt
```

Рисунок 9 - Назначение прав

Команда chmod (Change MODE – сменить режим) – 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;

mode – права доступа, устанавливаемые на файл. Права доступа можно записать в 2 вариантах – символьном и абсолютном.

Команда `chown` (CHange OWNer – сменить владельца) – позволяет сменить владельца файла. Для использования этой команды необходимо либо иметь права владельца текущего файла или права `root`. Синтаксис команды прост:

`chown username:groupname filename`, где

`username` – имя пользователя – нового владельца файла;

`groupname` – имя группы – нового владельца файла;

`filename` – имя файла, у которого сменяется владелец.

```
drwxrwxr-x 2 lovedie hate lovedie hate 4096 Oct 29 19:15 dir
prw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 20:14 fif
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:22 filen.tar.gz
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:20 filename
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:42 foo.in
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:41 foo.out
prw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:43 foo.pipe
-rw-rw-r-- 1 lovedie hate lovedie hate   26 Oct 29 12:31 loop
-rw-rw-r-- 1 lovedie hate lovedie hate   40 Oct 29 12:36 loop2
prw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:55 my-file-pipe
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Nov 12 19:12 new
-rw-rw-r-- 1 user      user      7 Nov 12 19:12 new_File.txt
-rw-rw-r-- 1 lovedie hate lovedie hate   20 Oct 29 19:53 out
-rwxrwxrwx 1 lovedie hate lovedie hate   793 Nov 12 18:45 text.txt
lovedie hate@myubuntuserver:~/lr2$ sudo chown lovedie hate new_File.txt
lovedie hate@myubuntuserver:~/lr2$ ls -l
total 24
drwxrwxr-x 2 lovedie hate lovedie hate 4096 Oct 29 19:15 dir
prw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 20:14 fif
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:22 filen.tar.gz
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:20 filename
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:42 foo.in
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:41 foo.out
prw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:43 foo.pipe
-rw-rw-r-- 1 lovedie hate lovedie hate   26 Oct 29 12:31 loop
-rw-rw-r-- 1 lovedie hate lovedie hate   40 Oct 29 12:36 loop2
prw-rw-r-- 1 lovedie hate lovedie hate    0 Oct 29 19:55 my-file-pipe
-rw-rw-r-- 1 lovedie hate lovedie hate    0 Nov 12 19:12 new
-rw-rw-r-- 1 lovedie hate user      7 Nov 12 19:12 new_File.txt
-rw-rw-r-- 1 lovedie hate lovedie hate   20 Oct 29 19:53 out
-rwxrwxrwx 1 lovedie hate lovedie hate   793 Nov 12 18:45 text.txt
```

Рисунок 10 - Назначение владельца

5. Ознакомиться с информацией(5) по теме процессы, посмотреть и опробовать примеры наиболее распространенных(6) команд, изучить возможность запуска процессов в supervisor

Процесс – понятие совокупности программного кода и данных, загруженных в память ЭВМ.

Процесс – это не запущенная программа (приложение) или команда, так как приложение может создавать несколько процессов одновременно. Код процесса не обязательно должен выполняться в текущий момент времени, так как процесс может находиться в состоянии спящего. В этом случае выполнение кода такого процесса приостановлено. Существует всего 3 состояния, в которых может находиться процесс:

Работающий процесс – в данный момент код этого процесса выполняется.

Спящий процесс – в данный момент код процесса не выполняется в ожидании какого-либо события (нажатия клавиши на клавиатуре, поступление данных из сети и т.д.)

Процесс-зомби – сам процесс уже не существует, его код и данные выгружены из оперативной памяти, но запись в таблице процессов остается по тем или иным причинам.

top

```
top - 19:22:22 up 1:25, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 89 total, 1 running, 87 sleeping, 1 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.0 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 981.3 total, 359.0 free, 128.9 used, 493.3 buff/cache
MiB Swap: 1710.0 total, 1710.0 free, 0.0 used. 694.0 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	102080	11676	8568	S	0.0	1.2	0:02.48	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
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	0:00.07	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	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
77	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	devfreq_wq
78	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdogd

Рисунок 11 - Команда top

top -u lovediechate

```
top - 19:25:57 up 1:29, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 91 total, 1 running, 88 sleeping, 2 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 : 981.3 total, 358.5 free, 129.4 used, 493.3 buff/cache
MiB Swap: 1710.0 total, 1710.0 free, 0.0 used. 693.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
928	lovedie+	20	0	18532	9740	8092	S	0.0	1.0	0:00.06	systemd
929	lovedie+	20	0	103424	3464	4	S	0.0	0.3	0:00.00	(sd-pam)
937	lovedie+	20	0	7068	5084	3384	S	0.0	0.5	0:00.16	bash
1091	lovedie+	20	0	4620	2116	1852	T	0.0	0.2	0:00.09	less
1273	lovedie+	20	0	7916	3796	3220	T	0.0	0.4	0:00.01	top
1274	lovedie+	20	0	7916	3752	3176	R	0.0	0.4	0:00.00	top

Рисунок 12 – Команда top для пользователя

```
top - 19:27:38 up 1:31, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 91 total, 1 running, 88 sleeping, 2 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.7 id, 0.3 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 981.3 total, 358.5 free, 129.4 used, 493.3 buff/cache
MiB Swap: 1710.0 total, 1710.0 free, 0.0 used. 693.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
928	lovedie+	20	0	18532	9740	8092	S	0.0	1.0	0:00.06	/lib/systemd/systemd --user
929	lovedie+	20	0	103424	3464	4	S	0.0	0.3	0:00.00	(sd-pam)
937	lovedie+	20	0	7068	5084	3384	S	0.0	0.5	0:00.16	-bash
1091	lovedie+	20	0	4620	2116	1852	T	0.0	0.2	0:00.09	less
1273	lovedie+	20	0	7916	3796	3220	T	0.0	0.4	0:00.01	top -u user
1275	lovedie+	20	0	7916	3892	3312	R	0.0	0.4	0:00.01	top -u lovediehate

Рисунок 13 - Нажатие 'с'

Теперь виден абсолютный путь к программам.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
928	lovedie+	20	0	18532	9740	8092	S	0.0	1.0	0:00.06	/lib/systemd/systemd --user
929	lovedie+	20	0	103424	3464	4	S	0.0	0.3	0:00.00	(sd-pam)
937	lovedie+	20	0	7068	5084	3384	S	0.0	0.5	0:00.16	-bash
1091	lovedie+	20	0	4620	2116	1852	T	0.0	0.2	0:00.09	less
1273	lovedie+	20	0	7916	3796	3220	T	0.0	0.4	0:00.01	top -u user
1275	lovedie+	20	0	7916	3892	3312	R	0.0	0.4	0:00.11	top -u lovediehate

Рисунок 14 – Нажатие 'd'

Изменен интервал обновления снимка.

```
PID to signal/kill [default pid = 672] 1091
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
928	lovedie+	20	0	18532	9740	8092	S	0.0	1.0	0:00.06	/lib/systemd/systemd --user
929	lovedie+	20	0	103424	3464	4	S	0.0	0.3	0:00.00	(sd-pam)
937	lovedie+	20	0	7068	5084	3384	S	0.0	0.5	0:00.16	-bash
1091	lovedie+	20	0	4620	2116	1852	T	0.0	0.2	0:00.09	less
1273	lovedie+	20	0	7916	3796	3220	T	0.0	0.4	0:00.01	top -u user
1275	lovedie+	20	0	7916	3892	3312	R	0.0	0.4	0:00.20	top -u lovediehate

Рисунок 15 - Убийство процесса кнопкой 'k'

```
PID to renice [default pid = 1275] 1091
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1275	lovedie+	20	0	7916	3892	3312	R	0.4	0.4	0:00.34	top -u lovediehate
928	lovedie+	20	0	18532	9740	8092	S	0.0	1.0	0:00.06	/lib/systemd/systemd --user
929	lovedie+	20	0	103424	3464	4	S	0.0	0.3	0:00.00	(sd-pam)
937	lovedie+	20	0	7068	5084	3384	S	0.0	0.5	0:00.16	-bash
1091	lovedie+	21	1	4620	2116	1852	T	0.0	0.2	0:00.09	less
1273	lovedie+	20	0	7916	3796	3220	T	0.0	0.4	0:00.01	top -u user

Рисунок 16 – Изменение приоритета процесса с 'r'

```
lovediehate@myubuntuserver:~/lr2$ top -n 1 -b > top-output.txt
lovediehate@myubuntuserver:~/lr2$ head -n5 top-output.txt
top - 19:38:20 up 1:41, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 91 total, 1 running, 87 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 6.2 sy, 0.0 ni, 93.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 981.3 total, 356.3 free, 131.6 used, 493.4 buff/cache
MiB Swap: 1710.0 total, 1710.0 free, 0.0 used. 691.3 avail Mem
```

Рисунок 17 – Перенаправление вывода в файл

```

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 18 – Справка на 'h'

Supervisor – это менеджер процессов, который существенно упрощает управление долго работающими программами, предоставляя простой и понятный интерфейс.

Для установки нужно ввести команду `sudo apt-get install supervisor`

```
root@myubuntu-server:/# sudo apt-get install supervisor
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  supervisor-doc
The following NEW packages will be installed:
  supervisor
0 upgraded, 1 newly installed, 0 to remove and 80 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 0s (994 kB/s)
Selecting previously unselected package supervisor.
(Reading database ... 104478 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) ...
Created symlink /etc/systemd/system/multi-user.target.wants/supervisor.service → /lib/systemd/system/supervisor.service.
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.2) ...
root@myubuntu-server:/# cd /etc/supervisor
```

Рисунок 19 – Установка supervisor

Допустим, у нас есть некий скрипт Somebody.

```
"Somebody" 5L, 55C written
root@myubuntuserver:/home/lovediehatе# sh Somebody
That I Used To Know
That I Used To Know
That I Used To Know
That I Used To Know
That I Used To Know
That I Used To Know
That I Used To Know
^C
root@myubuntuserver:/home/lovediehatе# ls
Somebody arch fifo lr2 my_arch
root@myubuntuserver:/home/lovediehatе# cat < Somebody
while true
do
echo 'That I Used To Know'
sleep 1
done
root@myubuntuserver:/home/lovediehatе# cd /etc/supervisor
root@myubuntuserver:/etc/supervisor# ls
conf.d supervisord.conf
root@myubuntuserver:/etc/supervisor# cd conf.d
root@myubuntuserver:/etc/supervisor/conf.d# ls
root@myubuntuserver:/etc/supervisor/conf.d# vi test.conf_
```

Рисунок 20 – Скрипт

Новые программы добавляются в Supervisor посредством конфигурационных файлов, которые передают переменные среды и сообщают, какой из исполнительных файлов нужно запустить в определенный момент, как обрабатывать вывод.

Создаем тестовый файл конфигураций в /etc/supervisor/conf.d

```
[program:test]
command=/home/lovediehatе/Somebody
autostart=true
autorestart=true
stderr_logfile=/home/lovediehatе/somebody.error.log
stdout_logfile=/home/lovediehatе/somebody.out.log
```

Рисунок 21 – Содержимое конфига

Ниже приведено описание каждой строки и некоторые тонкие настройки, которые могут пригодиться в дальнейшем.

```
[program:test]
```

```
command=/home/lovediehate/Somebody
```

Конфигурация начинается с определения программы и полного пути к ней.

```
autostart=true
```

```
autorestart=true
```

Эти строки определяют базовое автоматическое поведение скрипта в определенных условиях. Опция `autostart` запускает программу при загрузке системы; значение `false` заставит включать программу вручную. Опция `autorestart` определяет, как Supervisor будет управлять программой в случае ее отключения, и имеет три опции:

`false` – Supervisor никогда не будет перезапускать программу после завершения ее работы;

`true` – Supervisor будет всегда перезапускать программу после завершения работы;

`unexpected` – Supervisor будет перезапускать программу только в случае, если она завершила работу из-за возникновения неожиданного кода ошибки (любой стандартный код, кроме 0 и 2).

```
stderr_logfile=/home/lovediehate/somebody.err.log
```

```
stdout_logfile=/home/lovediehate/somebody.out.log
```

Последние две строки определяют местонахождение двух основных лог-файлов программы. В соответствии с именами опций, `stdout` и `stderr` задают расположение файлов `stdout_logfile` и `stderr_logfile`.

Запустим команды `supervisorctl reread && supervisorctl update`, чтобы Supervisor считал новые настройки и они вступили в силу. Затем проверим

содержимое лога вывода. Как видно на скрине, Supervisor запустил процесс вывода строки.

```
root@myubuntuuserver:/etc/supervisor/conf.d# supervisorctl reread && supervisorctl update && sleep 4
&& cd /home/lovediehatе/ && cat smbd.out.log
test: changed
test: stopped
test: updated process group
That I Used To Know
root@myubuntuuserver:/home/lovediehatе# supervisorctl reread && supervisorctl update && sleep 4 && cd
/home/lovediehatе/ && cat smbd.out.log
No config updates to processes
That I Used To Know
That I Used To Know
That I Used To Know
That I Used To Know
That I Used To Know
root@myubuntuuserver:/home/lovediehatе#
```

Рисунок 22 – Результат

Для проверки статуса процессов используем команду «supervisorctl».

Основные команды: start, stop, restart.

```
root@myubuntuuserver:/home/lovediehatе# supervisorctl
test                                RUNNING    pid 35387, uptime 0:06:21
supervisor> test stop
*** Unknown syntax: test stop
supervisor> stop test
test: stopped
supervisor> start test
test: started
```

Рисунок 23 – Остановка процесса

6. Запуск процессов по расписанию.

Crontab — это команда, используемая для установки, удаления или вывода файла конфигурации cron, используемого для управления демоном cron. Cron используется для планирования задач, которые будут выполняться периодически.

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`.

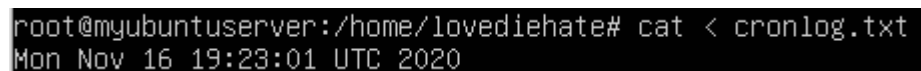
Строка `*/1 * * * * echo `date` >> /home/lovediehat/cronlog.txt` означает, что мы каждую минуту будем выводить текущую дату в файл `cronlog.txt`.



```
*/1 * * * * echo `date` >> /home/lovediehat/cronlog.txt
```

Рисунок 24 – Ввод crontab -e

Как мы видим, дата успешно вывелась в текстовом файле.



```
root@myubuntuuserver:/home/lovediehat# cat < cronlog.txt
Mon Nov 16 19:23:01 UTC 2020
```

Рисунок 25 – Результат в файле

Вывод

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