

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТНАЯ РАБОТА №3

по дисциплине «OS Linux»

на тему «Процессы в операционной системе Linux»

Студент

Сухоруких А.О.

Группа АС–18

Руководитель

Кургасов В.В.

К.Т.Н.

Липецк 2020 г.

Оглавление

Ход работы.....	4
2. Разобраться с понятиями конвейер, перенаправление ввода–вывода.....	10
3. Повторить назначение прав доступа. Команды chmod, chown	12
4. Ознакомиться с информацией по теме процессы, посмотреть и опробовать примеры наиболее распространенных команд, изучить возможность запуска процессов в supervisor.....	16
5. Изучить возможность автоматического запуска программ по расписанию.	21
Вывод.....	24

Цель работы

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

Ход работы

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

Создадим файл, в который запишем цифры от одного до двадцати пяти.

Назовем его test. Процесс создания файла test показан на рисунке 1.

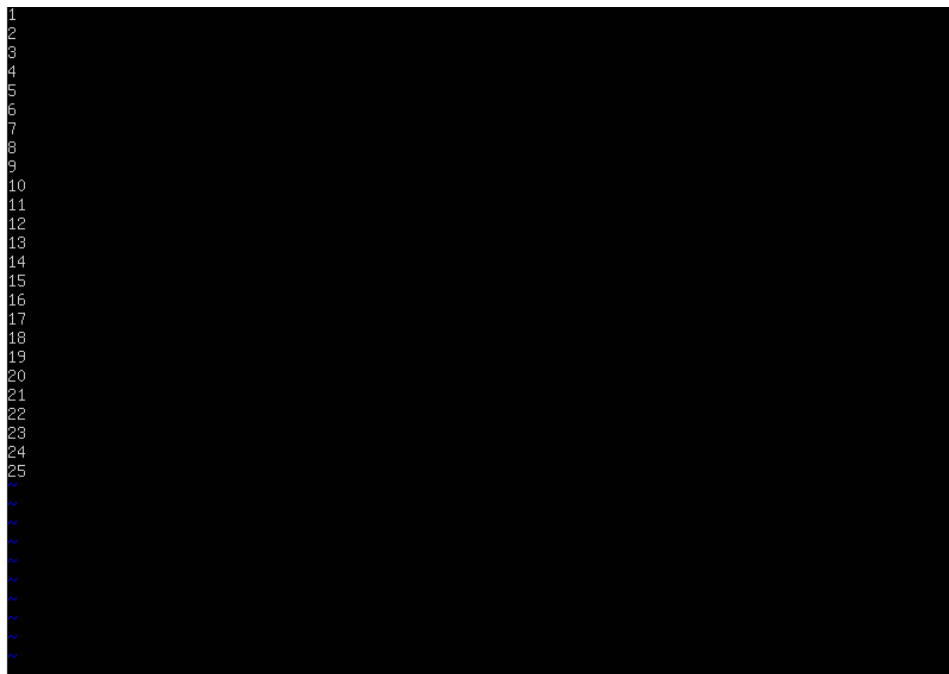


Рисунок 1 – Создание файла test

Задача команды cat – считать данные из файла или стандартного ввода и вывести их на экран.

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.

Пример работы команды cat показан на рисунке 2.

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
"test" (New) 25L, 66C written  
artem@artemserver:~$ cat test  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
artem@artemserver:~$
```

Рисунок 2 – Команда cat

Команда «head» в Unix или Linux системах используется для печати «N» линии из файла в терминал.

Основные 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.

Пример работы команды показан на рисунке 3.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$ head test
1
2
3
4
5
6
7
8
9
10
artem@artemserver:~$
```

Рисунок 3 –Пример работы команды head

Команда tail в операционных системах UNIX или Linux используется для вывода последних N строк из файла на терминал.

Основные опции:

- с – 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;

Результат выполнения команды tail показан на рисунке 4

```
12
13
14
15
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$ head test
1
2
3
4
5
6
7
8
9
10
artem@artemserver:~$ tail test
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$
```

Рисунок 4 – Результат выполнения команды tail

Команда «more» терминальная команда, которая используется при открытии файла для интерактивного чтения. Если содержимое файла слишком велико, чтобы помещаться на одном экране, оно отображает содержимое страницы за страницей. Пример выполнения команды показан на рисунке 5.

```
artem@artemserver:~$ more test
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$
```

Рисунок 5 – Результат выполнения команды more

Т.к. содержимое файла целиком поместилось в экран, то был выведен весь файл целиком.

Команда «less» также используется для открытия заданного файла для интерактивного чтения, позволяющего прокручивать и искать. Если содержимое файла слишком велико, оно выводит на экран и поэтому вы можете прокручивать страницу за страницей. В отличие от команды «more», она позволяет прокручивать в обоих направлениях. Пример работы команды less показан на рисунке 6



```
artem@artemserver:~$ less test
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
test (END)
```

Рисунок 6 – Результат выполнения команды less

Команда `grep` 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.

Основные опции:

- b – показывать номер блока перед строкой;
- c – подсчитать количество вхождений шаблона;
- h – не выводить имя файла в результатах поиска внутри файлов Linux;
- i – не учитывать регистр;
- l – отобразить только имена файлов, в которых найден шаблон;
- n – показывать номер строки в файле;
- s – не показывать сообщения об ошибках;

–v – инвертировать поиск, выдавать все строки кроме тех, что содержат шаблон;

–w – искать шаблон как слово, окружённое пробелами;

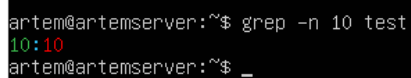
–e – использовать регулярные выражения при поиске;

–An – показать вхождение и n строк до него;

–Bn – показать вхождение и n строк после него;

–Cn – показать n строк до и после вхождения;

Поиск в файле test строки содержащую число 10 и вывести номер этой строки показан на рисунке 7



```
artem@artemserver:~$ grep -n 10 test
10:10
artem@artemserver:~$ _
```

Рисунок 7 – Поиск в файле test строки содержащую число 10 и вывод номер этой строки

Команда find имеет такой синтаксис:

find [каталог] [параметры] критерий шаблон

Основные 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 каталога в Linux

Основные критерии:

–name – поиск файлов по имени

–perm – поиск файлов в Linux по режиму доступа

- user – поиск файлов по владельцу
- group – поиск по группе
- mtime – поиск по времени модификации файла
- atime – поиск файлов по дате последнего чтения
- nogroup – поиск файлов, не принадлежащих ни одной группе
- nouser – поиск файлов без владельцев
- newer – найти файлы новее чем указанный
- size – поиск файлов в Linux по их размеру

Поиск файла по имени в текущем каталоге показан на рисунке 8.

```
artem@artemserver:~$ find . -name test
./test
artem@artemserver:~$ _
```

Рисунок 8 – Поиск файла по имени в текущем каталоге

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

Конвейер 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.

Пример конвейера показан на рисунках 9–10

```
artem@artemserver:~$ ls /bin | more
```

Рисунок 9 – Пример конвейера

```
bc
bootctl
bootctl
bsd-from
bsd-write
btrfs
btrfs-convert
btrfs-find-root
btrfs-image
btrfs-map-logical
btrfs-select-super
btrfsck
btrfstune
bunzip2
busctl
busybox
byobu
byobu-config
byobu-ctrl-a
byobu-disable
byobu-disable-prompt
byobu-enable
byobu-enable-prompt
byobu-export
byobu-janitor
byobu-keybindings
byobu-launch
byobu-launcher
byobu-launcher-install
byobu-launcher-uninstall
byobu-layout
byobu-prompt
byobu-quiet
byobu-reconnect-sockets
byobu-screen
byobu-select-backend
--More--
```

Рисунок 10 – Пример выполнения конвейера

Перенаправление ввода–вывода

В Linux все субстанции считаются файлами, в том числе и потоки ввода вывода linux – файлы. В каждом дистрибутиве есть три основных файла потоков, которые могут использовать программы, они определяются оболочкой и идентифицируются по номеру дескриптора файла:

STDIN или 0 – этот файл связан с клавиатурой и большинство команд получают данные для работы отсюда;

STDOUT или 1 – это стандартный вывод, сюда программа отправляет все результаты своей работы. Он связан с экраном, или если быть точным, то с терминалом, в котором выполняется программа;

STDERR или 2 – все сообщения об ошибках выводятся в этот файл.

Перенаправление ввода / вывода позволяет заменить один из этих файлов на свой. Например, вы можете заставить программу читать данные из файла в файловой системе, а не клавиатуры, также можете выводить ошибки в файл, а не на экран и т.д. Все это делается с помощью символов "<" и ">".

```

artem@artemserver:~$ sort > sort_result
123
345
6
221
457
7
89
artem@artemserver:~$ ls
loop loop2 loop3 ls newchannel newdir result result_new sort_result test
artem@artemserver:~$ cat sort_result
123
221
345
457
6
7
89
artem@artemserver:~$ _

```

Рисунок 11 – Перенаправление вывода

```

artem@artemserver:~$ sort < test > sort_result
artem@artemserver:~$ cat sort_result
1
10
11
12
13
14
15
16
17
18
19
2
20
21
22
23
24
25
3
4
5
6
7
8
9
artem@artemserver:~$

```

Рисунок 12 – Перенаправление ввода и вывода

3. Повторить назначение прав доступа. Команды `chmod`, `chown`

Команда `chmod` используется для изменения прав доступа к файлам или каталогам.

В Linux и других Unix-подобных операционных системах для каждого файла существует набор правил, которые определяют, кто и как может получить доступ к этому файлу. Эти правила называются правами доступа к файлам или режимами файлов. Имя команды `chmod` означает «режим изменения» и используется для определения способа доступа к файлу.

В общем виде команды `chmod` выглядят так:

\$ `chmod` опции права /путь/к/файлу

Есть три основных вида прав:

r – чтение;

w – запись;

x – выполнение;

Также есть три категории пользователей, для которых вы можете установить эти права на файл linux:

u – владелец файла;

g – группа файла;

o – все остальные пользователи;

В качестве действий могут использоваться знаки "+" – включить или "-" – отключить:

u+x – разрешить выполнение для владельца;

ugo+x – разрешить выполнение для всех;

ug+w – разрешить запись для владельца и группы;

o-x – запретить выполнение для остальных пользователей;

ugo+rwX – разрешить все для всех;

Но права можно записывать не только таким способом. Есть еще восьмеричный формат записи:

0 – никаких прав;

1 – только выполнение;

2 – только запись;

3 – выполнение и запись;

4 – только чтение;

5 – чтение и выполнение;

6 – чтение и запись;

7 – чтение запись и выполнение.

Для изменения прав доступа воспользуемся командой `chmod 764 test`. В результате владелец файла получит все права доступа, группа получит право только на чтение и запись, а все остальные пользователи право только на чтение. Результат выполнения команды показан на рисунке 13

```

root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-r--r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
-rw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchanel
drwxrwxr-x 3 artem artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rw-rw-r-- 1 artem artem 66 Nov 12 17:10 test
root@artemserver:/home/artem# chmod 764 test
root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-r--r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
-rw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchanel
drwxrwxr-x 3 artem artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 artem artem 66 Nov 12 17:10 test
root@artemserver:/home/artem#

```

Рисунок 13 – Изменение прав доступа

Chown — UNIX-утилита, изменяющая владельца и/или группу для указанных файлов. В качестве имени владельца/группы берётся первый аргумент, не являющийся опцией. Если задано только имя пользователя (или числовой идентификатор пользователя), то данный пользователь становится владельцем каждого из указанных файлов, а группа этих файлов не изменяется. Если за именем пользователя через двоеточие следует имя группы (или числовой идентификатор группы), без пробелов между ними, то изменяется также и группа файла.

Основные опции:

- c, –changes – подробный вывод всех выполняемых изменений;
- f, –silent, –quiet – минимум информации, скрыть сообщения об ошибках;
- dereference – изменять права для файла к которому ведет символическая ссылка вместо самой ссылки (поведение по умолчанию);
- h, –no-dereference – изменять права символических ссылок и не трогать файлы, к которым они ведут;
- from – изменять пользователя только для тех файлов, владельцем которых является указанный пользователь и группа;

- R, –recursive – рекурсивная обработка всех подкаталогов;
- H – если передана символическая ссылка на директорию – перейти по ней;
- L – переходить по всем символическим ссылкам на директории;
- P – не переходить по символическим ссылкам на директории (по умолчанию)

Изменим владельца каталога newdir на root. Результат показан на рисунке 14

```
root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-r--r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
prw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchannel
drwxrwxr-x 3 artem artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 artem artem 66 Nov 12 17:10 test
root@artemserver:/home/artem# chown root ./newdir
root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-r--r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
prw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchannel
drwxrwxr-x 3 root artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 artem artem 66 Nov 12 17:10 test
root@artemserver:/home/artem# _
```

Рисунок 14 – Изменение владельца папки

Изменим владельца и группу файла test на root. Результат показан на рисунке 15

```
root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-r--r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
prw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchannel
drwxrwxr-x 3 root artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 artem artem 66 Nov 12 17:10 test
root@artemserver:/home/artem# chown root:root test
root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-r--r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
prw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchannel
drwxrwxr-x 3 root artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 root root 66 Nov 12 17:10 test
root@artemserver:/home/artem# _
```

Рисунок 15 – Изменение владельца и группы

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

Процессом называется выполняемая в данный момент программа или ее потомки. Каждый процесс запускается от имени какого-то пользователя. Каждый пользователь может управлять поведением процессов, им запущенных. При этом пользователь root может управлять всеми процессами — как запущенными от его имени.

Каждый процесс в системе имеет уникальный номер — идентификационный номер процесса (Process Identification, PID). Этот номер используется ядром операционной системы, а также некоторыми утилитами для управления процессами.

Команда `ps -f` выводит следующую информацию о процессах:

UID – пользователь, от имени которого запущен процесс;

PID – идентификатор процесса;

PPID – идентификатор родительского процесса;

C – процент времени CPU, используемого процессом;

STIME – время запуска процесса;

TTY – терминал, из которого запущен процесс;

TIME – общее время процессора, затраченное на выполнение процессора;

CMD – команда запуска процессора;

Пример выполнения команды `ps -f` показан на рисунке 16


```

root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-rw-r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
prw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchannel
drwxrwxr-x 3 root artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 artem artem 66 Nov 12 17:10 test
root@artemserver:/home/artem# chown root:root test
root@artemserver:/home/artem# ls -l
total 32
-rw-rw-r-- 1 artem artem 27 Oct 29 17:19 loop
-rw-rw-r-- 1 artem artem 27 Oct 30 17:34 loop2
-rw-rw-r-- 1 root root 27 Oct 30 17:35 loop3
-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
prw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchannel
drwxrwxr-x 3 root artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 root root 66 Nov 12 17:10 test
root@artemserver:/home/artem# ps -f

```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	685	1	0	13:58	tty1	00:00:00	/bin/login -p --
root	1021	1011	0	14:04	tty1	00:00:00	sudo su
root	1023	1021	0	14:04	tty1	00:00:00	su
root	1024	1023	0	14:04	tty1	00:00:00	bash
root	1074	1024	0	14:20	tty1	00:00:00	ps -f

```

root@artemserver:/home/artem#

```

Рисунок 16 – Пример выполнения команды ps -f

Запустим два процесса. Первый процесс loop в фоновом режиме, второй loop2 на переднем плане и остановим его послав сигнал STOP. Результат выполнения показан на рисунке 17

```

-rw-rw-r-- 1 artem artem 0 Oct 30 16:17 ls
prw-rw-r-- 1 artem artem 0 Oct 30 16:33 newchannel
drwxrwxr-x 3 root artem 4096 Oct 30 16:33 newdir
-rw-rw-r-- 1 artem artem 80 Oct 30 16:10 result
-rw-rw-r-- 1 artem artem 118 Oct 30 16:33 result_new
-rw-rw-r-- 1 artem artem 66 Nov 12 17:58 sort_result
-rwxrwxr-- 1 root root 66 Nov 12 17:10 test
root@artemserver:/home/artem# ps -f

```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	685	1	0	13:58	tty1	00:00:00	/bin/login -p --
root	1021	1011	0	14:04	tty1	00:00:00	sudo su
root	1023	1021	0	14:04	tty1	00:00:00	su
root	1024	1023	0	14:04	tty1	00:00:00	bash
root	1074	1024	0	14:20	tty1	00:00:00	ps -f

```

root@artemserver:/home/artem# sh loop&
[1] 1075
root@artemserver:/home/artem# ps -f

```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	685	1	0	13:58	tty1	00:00:00	/bin/login -p --
root	1021	1011	0	14:04	tty1	00:00:00	sudo su
root	1023	1021	0	14:04	tty1	00:00:00	su
root	1024	1023	0	14:04	tty1	00:00:00	bash
root	1075	1024	85	14:22	tty1	00:00:05	sh loop
root	1076	1024	0	14:22	tty1	00:00:00	ps -f

```

root@artemserver:/home/artem# sh loop2
^Z
[2]+  Stopped                  sh loop2
root@artemserver:/home/artem# ps -f

```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	685	1	0	13:58	tty1	00:00:00	/bin/login -p --
root	1021	1011	0	14:04	tty1	00:00:00	sudo su
root	1023	1021	0	14:04	tty1	00:00:00	su
root	1024	1023	0	14:04	tty1	00:00:00	bash
root	1075	1024	80	14:22	tty1	00:00:53	sh loop
root	1077	1024	37	14:23	tty1	00:00:11	sh loop2
root	1079	1024	0	14:23	tty1	00:00:00	ps -f

```

root@artemserver:/home/artem# _

```

Рисунок 17 – Запуск двух процессов

Создадим сценарий, который будет каждые 5 секунд выводить текущую дату. Процесс создания скрипта показан, на рисунке 18.

A screenshot of a terminal window with a black background and white text. The text shows the creation of a script named 'nowdate'. The script's content is:

```
while true
do
echo `date`
sleep 5
done
```

 Below the script content, there are several lines of tilde characters (~) representing the script's body. At the bottom of the terminal, it says:

```
"nowdate" [New] 5L, 39C written
```

Рисунок 18 – Текст скрипта

После установки, supervisor нужно сконфигурировать и добавить программы/процессы, которыми он будет управлять. Файл конфигурации по умолчанию находится в /etc/supervisor/supervisord.conf. Добавим написанный скрипт. Добавление скрипта показано на рисунке 19.

```

; supervisor config file

[unix_http_server]
file=/var/run/supervisor.sock ; (the path to the socket file)
chmod=0700 ; socket file mode (default 0700)

[supervisord]
logfile=/var/log/supervisor/supervisord.log ; (main log file;default $CWD/supervisord.log)
pidfile=/var/run/supervisord.pid ; (supervisord pidfile;default supervisord.pid)
childlogdir=/var/log/supervisor ; ('AUTO' child log dir, default $TEMP)

; the below section must remain in the config file for RPC
; (supervisorctl/web interface) to work, additional interfaces may be
; added by defining them in separate rpcinterface: sections
[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[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:date]
command=sh /home/artem/nowdate.sh
autostart=true
autorestart=true
stderr_logfile=/home/artem/errnowdate.log
stdout_logfile=/home/artem/nowdate/logs_log
user=root
-- INSERT --

```

35,39

All

Рисунок 19 – Изменение файла конфигурации

[program:date] — название процесса/воркера, к которому будут относиться все последующие параметры секции;

command— команда на запуск файла, то есть путь к нужному файлу;

autostart — запуск воркера вместе с запуском supervisor;

autorestart— перезапуск воркера, если тот по какой-то причине упал;

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 программу supervisorctl имеет интерактивный режим, который позволяет управлять программами supervisor.

Введем команду help для вывода информации о командах. На рисунке 20 мы видим основные команды supervisor

```

root@artemserver:/etc/supervisor# supervisorctl
date                                RUNNING    pid 3019, uptime 0:00:12
supervisor> help

default commands (type help <topic>):
=====
add      exit      open  reload  restart  start  tail
avail    fg         pid   remove  shutdown status  update
clear    maintail  quit  reread  signal   stop   version

supervisor> _

```

Рисунок 20 – Результат команды help в supervisor

Остановим, запустим и через некоторое время перезапустим наш скрипт.

Результат выполнения данных операций показан на рисунках 21 – 22

```

root@artemserver:/etc/supervisor# supervisorctl
date                                RUNNING    pid 3019, uptime 0:00:12
supervisor> help

default commands (type help <topic>):
=====
add      exit      open  reload  restart  start  tail
avail    fg         pid   remove  shutdown status  update
clear    maintail  quit  reread  signal   stop   version

supervisor> stop date
date: stopped
supervisor> start date
date: started
supervisor> restart date
date: stopped
date: started
supervisor> _

```

Рисунок 21 – Остановка, запуск и перезапуск процесса

```

default commands (type help <topic>):
=====
add      exit      open  reload  restart  start  tail
avail    fg         pid   remove  shutdown status  update
clear    maintail  quit  reread  signal   stop   version

supervisor> stop date
date: stopped
supervisor> start date
date: started
supervisor> restart date
date: stopped
date: started
supervisor> quit

root@artemserver:/etc/supervisor# ls
conf.d  supervisor.conf
root@artemserver:/etc/supervisor# cd /
root@artemserver:/# cd home/artem
root@artemserver:/home/artem# ls
bash: /ls: No such file or directory
root@artemserver:/home/artem# ls
errloop.log  loop  loop3  newchanel  nowdate.sh  nowloop.log  result_new  test
errnowdate.log  loop2  ls  newdir  nowdatelogs.log  result  sort_result
root@artemserver:/home/artem# tail nowdatelogs.log
Fri Nov 13 17:05:25 UTC 2020
Fri Nov 13 17:05:30 UTC 2020
Fri Nov 13 17:05:35 UTC 2020
Fri Nov 13 17:05:40 UTC 2020
Fri Nov 13 17:05:45 UTC 2020
Fri Nov 13 17:05:50 UTC 2020
Fri Nov 13 17:05:55 UTC 2020
Fri Nov 13 17:06:00 UTC 2020
Fri Nov 13 17:06:05 UTC 2020
Fri Nov 13 17:06:10 UTC 2020
root@artemserver:/home/artem#

```

Рисунок 22 – Файл с логами процесса

5. Изучить возможность автоматического запуска программ по расписанию.

cron – это планировщик, который позволяет выполнять нужные скрипты раз в час, раз в день, неделю или месяц, а также в любое заданное вами время или через любой интервал.

Существует несколько конфигурационных файлов, из которых он берет информацию о том что и когда нужно выполнять. Сервис открывает файл /etc/crontab, в котором указаны все нужные данные. Часто, в современных дистрибутивах там прописан запуск утилиты run-parts, которая запускает нужные скрипты из следующих папок:

/etc/cron.minutely – каждую минуту;

/etc/cron.hourly – каждый час;

/etc/cron.daily – каждый день;

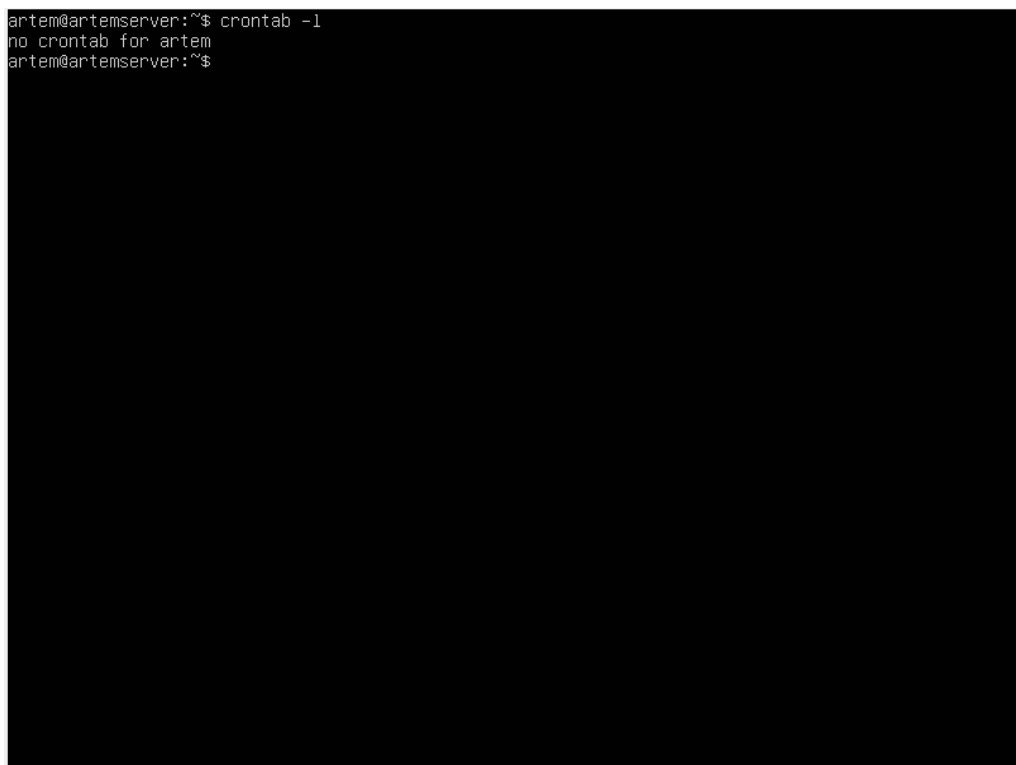
/etc/cron.weekly – каждую неделю;

/etc/cron.monthly – каждый месяц.

Синтаксис настройки одной задачи cron:

минута час день месяц день_недели /путь/к/исполняемому/файлу

Сначала посмотрим, что программ для запуска по расписанию у пользователя на данный момент нет, для этого воспользуемся командой `crontab -l`. Результат выполнения команды показан на рисунке 23.



```
artem@artemserver:~$ crontab -l
no crontab for artem
artem@artemserver:~$
```

Рисунок 23 – Выполнение команды `crontab -l`

Затем введем команду `crontab -e` чтобы создать расписание. Пусть каждые 5 мин будет выполняться команда `ls`. Для этого необходимо написать `*/*5*** ls`. Пример создания программ для запуска по расписанию показан на рисунках 24.

[illegible]

Рисунок 24 – Пример создания программ для запуска по расписанию

Вывод

В ходе выполнения данной лабораторной работы были получены знания о создании процессов в ОС Linux, работе процессов в разных режимах, создании расписаний