

# Липецкий государственный технический университет

Кафедра прикладной математики

## Отчет по лабораторной работе №3 «Планирование процессов»

Студент

\_\_\_\_\_  
подпись, дата

Стукановский А.О.  
фамилия, инициалы

Группа

ПМ-18

Руководитель  
доц., к.п.н. кафедры АСУ  
ученая степень, ученое звание

\_\_\_\_\_  
подпись, дата

Кургасов В. В.  
фамилия, инициалы

Липецк 2020 г.

# Содержание

<b>Цель работы</b>	<b>3</b>
<b>Задание кафедры</b>	<b>3</b>
<b>Выполнение работы</b>	<b>4</b>
Повторить команды cat, head, tail, more, less, grep, find . . . . .	4
cat . . . . .	4
head . . . . .	4
tail . . . . .	5
more . . . . .	6
less . . . . .	8
grep . . . . .	9
find . . . . .	10
Перенаправление ввода вывода, конвейер, конвейеризация . . . . .	12
Перенаправление ввода вывода . . . . .	12
Конвейеры . . . . .	12
Права доступа . . . . .	13
chmod . . . . .	13
chown . . . . .	14
Процессы. Запуск процессов в supervisor . . . . .	15
top . . . . .	15
Запуск процессов в supervisor . . . . .	16
Автоматический запуск по расписанию . . . . .	17
<b>Вывод</b>	<b>19</b>
<b>Список литературы</b>	<b>20</b>

## Цель работы

Ознакомиться на практике с планированием процессов в ОС Ubuntu.

## Задание кафедры

- 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.

# Выполнение работы

## Повторить команды cat, head, tail, more, less, grep, find

### cat

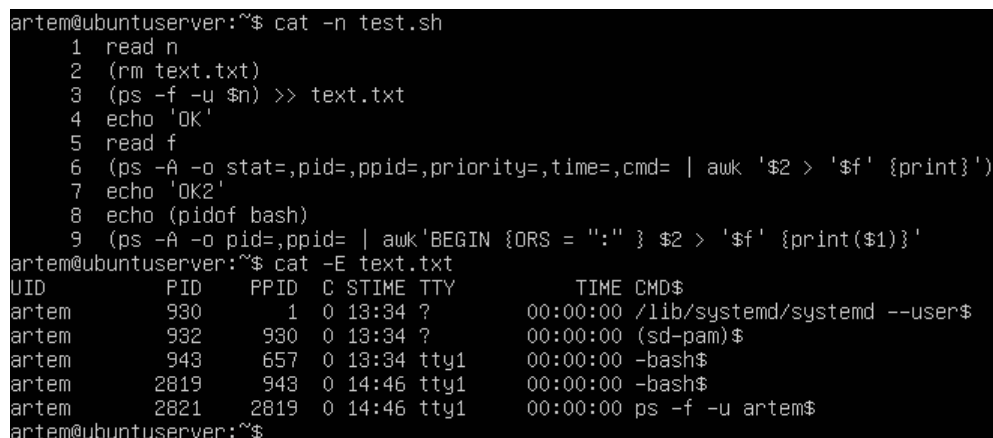
Название команды - это сокращения от слова catenate. По сути, задача команды cat очень проста - она читает данные из файла или стандартного ввода и выводит их на экран. Это все, чем занимается утилита. Но с помощью ее опций и операторов перенаправления вывода можно сделать очень многое. Сначала рассмотрим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

`$ cat опции файл1 файл2 ...`

Вы можете 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.

Пример работы команды приведён на рисунке 1.



```
artem@ubuntuuserver:~$ cat -n test.sh
 1 read n
 2 (rm text.txt)
 3 (ps -f -u $n) >> text.txt
 4 echo 'OK'
 5 read f
 6 (ps -A -o stat=,pid=,ppid=,priority=,time=,cmd= | awk '$2 > '$f' {print}}')
 7 echo 'OK2'
 8 echo (pidof bash)
 9 (ps -A -o pid=,ppid= | awk 'BEGIN {ORS = ":" } $2 > '$f' {print($1)}')
artem@ubuntuuserver:~$ cat -E text.txt
UID      PID      PPID  C  STIME TTY          TIME CMD$
artem    930        1  0  13:34 ?        00:00:00 /lib/systemd/systemd --user$
artem    932       930  0  13:34 ?        00:00:00 (sd-pam)$
artem    943       657  0  13:34 tty1    00:00:00 -bash$
artem   2819       943  0  14:46 tty1    00:00:00 -bash$
artem   2821     2819  0  14:46 tty1    00:00:00 ps -f -u artem$
artem@ubuntuuserver:~$
```

Рисунок 1.

### head

Команда head выводит начальные строки (по умолчанию — 10) из одного или нескольких документов. Также она может показывать данные, которые передает на вывод другая утилита.

Синтаксис у команды head следующий:

`$ head опции файл`

Здесь:

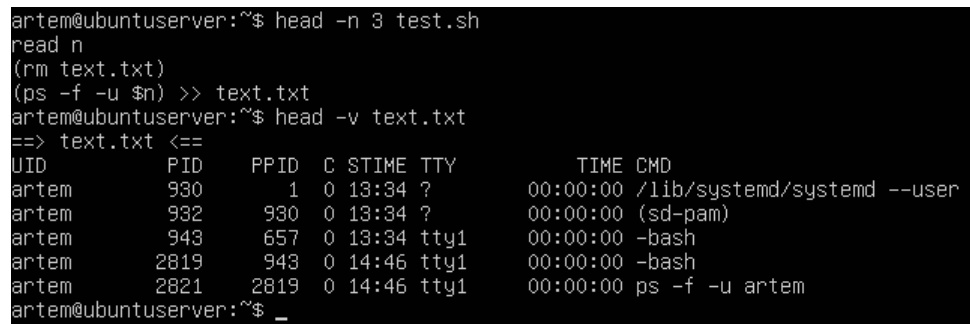
Опции — это параметр, который позволяет настраивать работу команды таким образом, чтобы результат соответствовал конкретным потребностям пользователя.

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.

Пример работы команды приведён на рисунке 2.



```
artem@ubuntu-server:~$ head -n 3 test.sh
read n
(rm text.txt)
(ps -f -u $n) >> text.txt
artem@ubuntu-server:~$ head -v text.txt
==> text.txt <==
UID      PID     PPID    C  STIME TTY          TIME CMD
artem    930      1      0  13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932     930      0  13:34 ?        00:00:00 (sd-pam)
artem    943     657      0  13:34 tty1    00:00:00 -bash
artem   2819     943      0  14:46 tty1    00:00:00 -bash
artem   2821    2819      0  14:46 tty1    00:00:00 ps -f -u artem
artem@ubuntu-server:~$ _
```

Рисунок 2.

## tail

Tail позволяет выводить заданное количество строк с конца файла, а также выводить новые строки в интерактивном режиме.

Синтаксис:

`$ 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;

Пример работы команды приведён на рисунке 3.

```
artem@ubuntuserver:~$ tail -q text.txt
UID      PID    PPID    C  STIME TTY          TIME CMD
artem    930      1    0 13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932     930    0 13:34 ?        00:00:00 (sd-pam)
artem    943     657    0 13:34 tty1    00:00:00 -bash
artem   2819     943    0 14:46 tty1    00:00:00 -bash
artem   2821    2819    0 14:46 tty1    00:00:00 ps -f -u artem
artem@ubuntuserver:~$ tail -n 4 test.sh
(ps -A -o stat=,pid=,ppid=,priority=,time=,cmd= | awk '$2 > '$f' {print}')
echo 'OK2'
echo (pidof bash)
(ps -A -o pid=,ppid= | awk 'BEGIN {ORS = ":" } $2 > '$f' {print($1)}')
artem@ubuntuserver:~$
```

Рисунок 3.

## more

Утилита more предназначена для постраничного просмотра файлов в терминале Linux.

В эмуляторе терминала Linux команда записывается так:

\$ *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;
- -p — 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.

Пример работы команды приведён на рисунках с 4 по 6.

```
artem@ubuntuserver:~$ more -d txt2.txt_
```

Рисунок 4.

```

PID TTY      TIME CMD
  1 ?        00:00:01 systemd
  2 ?        00:00:00 kthreadd
  3 ?        00:00:00 rcu_gp
  4 ?        00:00:00 rcu_par_gp
  6 ?        00:00:00 kworker/0:0H-kblockd
  8 ?        00:00:00 kworker/u2:0-events_power_efficient
  9 ?        00:00:00 mm_percpu_wq
 10 ?        00:00:00 ksoftirqd/0
 11 ?        00:00:00 rcu_sched
 12 ?        00:00:00 migration/0
 13 ?        00:00:00 idle_inject/0
 14 ?        00:00:00 cpuhp/0
 15 ?        00:00:00 kdevtmpfs
 16 ?        00:00:00 netns
 17 ?        00:00:00 rcu_tasks_kthre
 18 ?        00:00:00 kauditd
 19 ?        00:00:00 khungtaskd
 20 ?        00:00:00 oom_reaper
 21 ?        00:00:00 writeback
 22 ?        00:00:00 kcompactd0
 23 ?        00:00:00 ksmd
 24 ?        00:00:00 khugepaged
 70 ?        00:00:00 kintegrityd
 71 ?        00:00:00 kblockd
 72 ?        00:00:00 blkcg_punt_bio
 73 ?        00:00:00 tpm_dev_wq
 74 ?        00:00:00 ata_sff
 75 ?        00:00:00 md
 76 ?        00:00:00 edac-poller
 77 ?        00:00:00 devfreq_wq
 78 ?        00:00:00 watchdogd
 81 ?        00:00:00 kswapd0
 82 ?        00:00:00 ecryptfs-kthrea
 84 ?        00:00:00 kthrotld
 85 ?        00:00:00 acpi_thermal_pm
--More--(36%) [Press space to continue, 'q' to quit.]_

```

Рисунок 5.

```
86 ?      00:00:00 scsi_ah_0
87 ?      00:00:00 scsi_tm_0
88 ?      00:00:00 scsi_ah_1
89 ?      00:00:00 scsi_tm_1
91 ?      00:00:00 vfio-irqfd-clea
93 ?      00:00:00 ipv6_addrconf
102 ?     00:00:00 kstrp
105 ?     00:00:00 kworker/u3:0
118 ?     00:00:00 charger_manager
119 ?     00:00:00 kworker/0:1H-kblockd
162 ?     00:00:00 scsi_ah_2
163 ?     00:00:00 scsi_tm_2
164 ?     00:00:00 kworker/0:3-events
175 ?     00:00:00 cryptd
183 ?     00:00:00 irq/18-vmwgfx
185 ?     00:00:00 ttm_swap
202 ?     00:00:00 kdmflush
225 ?     00:00:00 raid5wq
265 ?     00:00:00 jbd2/dm-0-8
266 ?     00:00:00 ext4-rsv-conver
334 ?     00:00:00 systemd-journal
365 ?     00:00:00 systemd-udevd
386 ?     00:00:00 iprt-VBoxQueue
504 ?     00:00:00 kaluad
505 ?     00:00:00 kmpath_rdacd
506 ?     00:00:00 kmpathd
507 ?     00:00:00 kmpath_handlerd
508 ?     00:00:00 multipathd
518 ?     00:00:00 loop0
519 ?     00:00:00 loop1
521 ?     00:00:00 loop2
523 ?     00:00:00 loop3
525 ?     00:00:00 loop4
527 ?     00:00:00 loop5
530 ?     00:00:00 loop6
531 ?     00:00:00 loop7
--More--(73%)[Press space to continue, 'q' to quit.]
```

Рисунок 6.

## less

Less — 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.

Запись команды less в терминале выглядит так:

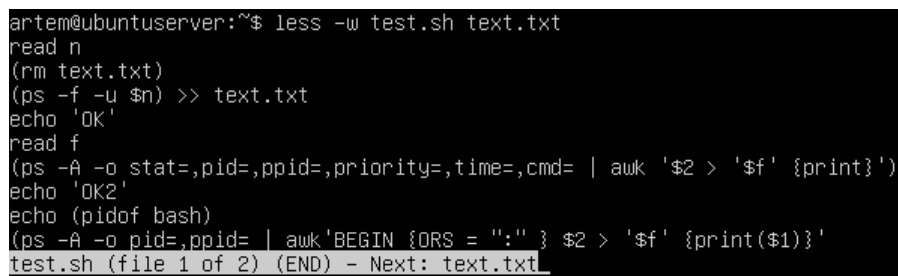
\$ less опции файл

- -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`, `-hilitte-unread` — выделить первую строку нового фрагмента текста.

Пример работы команды приведён на рисунке 7.



```
artem@ubuntu-server:~$ less -w test.sh text.txt
read n
(rm test.txt)
(ps -f -u $n) >> text.txt
echo 'OK'
read f
(ps -A -o stat=,pid=,ppid=,priority=,time=,cmd= | awk '$2 > '$f' {print}'))
echo 'OK2'
echo (pidof bash)
(ps -A -o pid=,ppid= | awk 'BEGIN {ORS = ":"} $2 > '$f' {print($1)}')
test.sh (file 1 of 2) (END) - Next: text.txt
```

Рисунок 7.

## grep

С помощью `grep` можно искать строки в файлах, фильтровать вывод команд. Синтаксис команды выглядит следующим образом:

`$ grep [опции] шаблон [файл]`

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

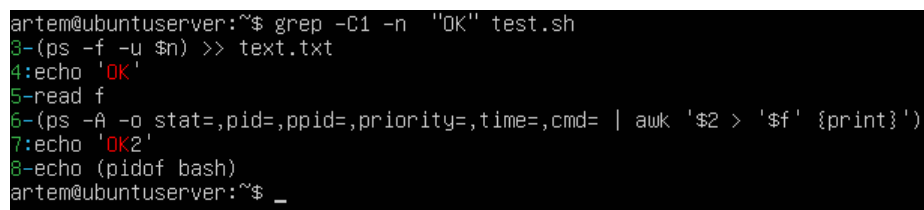
Файл и команда - это то место, где будет вестись поиск. Как вы увидите дальше, `grep` позволяет искать в нескольких файлах и даже в каталоге, используя рекурсивный режим.

Основные опции утилиты:

- `-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;

Пример работы команды приведён на рисунке 8.



```

artem@ubuntuserver:~$ grep -C1 -n "OK" test.sh
3-(ps -f -u $n) >> text.txt
4:echo 'OK'
5-read f
6-(ps -A -o stat=,pid=,ppid=,priority=,time=,cmd= | awk '$2 > '$f' {print}')
7:echo 'OK2'
8-echo (pidof bash)
artem@ubuntuserver:~$ _

```

Рисунок 8.

## find

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.

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.

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 папки 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

Пример работы команды приведён на рисунках 9 и 10.

```
artem@ubuntuserver:~$ find -print
.
./loop2
./profile
./bash_history
./bashrc
./config
./config/procps
./txt2.txt
./ps.swp
./pipe
./text.txt
./snap
./snap/tree
./snap/tree/common
./snap/tree/common/.local
./snap/tree/common/.local/lib
./snap/tree/common/.local/lib/locale
./snap/tree/18
./snap/tree/current
./loop2.swp
./test.sh.swp
./test.sh
./viminfo
./bash_logout
./sudo_as_admin_successful
./loop
./result.txt
./text.txt.swp
./cache
./cache/motd.legal-displayed
artem@ubuntuserver:~$
```

Рисунок 9.

```

artem@ubuntuserver:~$ find -maxdepth 1 -print
.
./loop2
./.profile
./.bash_history
./.bashrc
./.config
./txt2.txt
./ps.swp
./pipe
./text.txt
./snap
./loop2.swp
./test.sh.swp
./test.sh
./viminfo
./bash_logout
./sudo_as_admin_successful
./loop
./result.txt
./text.txt.swp
./cache
artem@ubuntuserver:~$ _

```

Рисунок 10.

## Перенаправление ввода вывода, конвейер, конвейеризация

### Перенаправление ввода вывода

Перенаправление обычно осуществляется вставкой специального символа `>` между командами. Обычно синтаксис выглядит так: команда1 `>` файл1 — выполняет команду1, помещая стандартный вывод в файл1; команда1 `<` файл1 выполняет команду1, используя в качестве источника ввода файл1 (вместо клавиатуры). На каждый запрос ввода программы считывается одна строка текста из файла. Конструкция команда1 `<` файл1 `>` файл2 совмещает два предыдущих варианта: выполняет команду1 ввода из файла1 и вывода в файл2.

Если использовать `»` вместо `>` то перенаправление вывода будет дополнять файл, вместо его пересоздания.

Пример перенаправления ввода вывода изображён на рисунке 11

```

artem@ubuntuserver:~$ ps -l > pvv.txt
artem@ubuntuserver:~$ less pvv.txt
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000      915      648  0  80   0 -  1768 do_wai tty1      00:00:00 bash
0 T  1000     1060      915  0  80   0 -  1979 do_sig tty1      00:00:00 top
0 R  1000     1285      915  0  80   0 -  1888 -      tty1      00:00:00 ps
pvv.txt (END)

```

Рисунок 11.

### Конвейеры

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

Пример работы конвейера изображён на рисунке 12. Здесь результат `ps -f` перенаправляется в `grep` чтобы наложить ограничения на выходные данные.

```
artem@ubuntu:~$ ps -f | grep bash
artem      915      648  0 16:56 tty1    00:00:00 -bash
artem     1331      915  0 19:14 tty1    00:00:00 grep --color=auto bash
artem@ubuntu:~$ _
```

Рисунок 12.

## Права доступа

### chmod

Команда, предназначенная для установки полномочий.

Синтаксис:

```
$ chmod [опции] права /путь_к_файлу
```

Опции:

- -c - выводить информацию обо всех изменениях;
- -f - не выводить сообщения об ошибках;
- -v - выводить максимум информации;
- --preserve-root - не выполнять рекурсивные операции для корня "/";
- --reference - взять маску прав из указанного файла;
- -R - включить поддержку рекурсии;
- --version - вывести версию утилиты;

Синтаксис настройки прав такой:

```
группа_пользователей действие вид_прав
```

В качестве действий могут использоваться знаки "+" включить или "-" отключить. Рассмотрим несколько примеров:

$u + x$  - 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, он более сложен для понимания, но пишется короче и проще. Я не буду рассказывать как считать эти цифры, просто запомните какая цифра за что отвечает, так проще:

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.

Права на папку linux такие же, как и для файла. Во время установки прав сначала укажите цифру прав для владельца, затем для группы, а потом для остальных. Например:

744 - разрешить все для владельца, а остальным только чтение;

755 - все для владельца, остальным только чтение и выполнение;

764 - все для владельца, чтение и запись для группы, и только чтение для остальных;

777 - всем разрешено все.

Пример изменения прав доступа изображён на рисунке 13.

```
artem@ubuntuuser:~$ chmod u+rwx test.sh
artem@ubuntuuser:~$ ls test.sh
test.sh
artem@ubuntuuser:~$ ls -l test.sh
-rwxrwxr-x 1 artem artem 237 Nov  4 14:46 test.sh
artem@ubuntuuser:~$
```

Рисунок 13.

## chown

Владельца файла и группу можно менять, для этого используются команды chown.//Синтаксис chown:

`$ chownпользователь [опции] /путь_к_файлу`

В поле пользователь надо указать пользователя, которому мы хотим передать файл. Также можно указать через двоеточие группу, например, пользователь:группа. Тогда изменится не только пользователь, но и группа. 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).

Пример изменения владельца файла приведён на рисунке 14.

```
artem@ubuntuuser:~$ ls
1.txt 2.txt loop loop2 pipe pvv.txt result.txt snap test.sh text.txt txt2.txt
artem@ubuntuuser:~$ chown user ./1.txt
chown: changing ownership of './1.txt': Operation not permitted
artem@ubuntuuser:~$ sudo chown user ./1.txt
[sudo] password for artem:
artem@ubuntuuser:~$ ls
1.txt 2.txt loop loop2 pipe pvv.txt result.txt snap test.sh text.txt txt2.txt
artem@ubuntuuser:~$ ls /home/user
1.txt 3.txt simvol2.txt snap strict2.txt
```

Рисунок 14.

## Процессы. Запуск процессов в supervisor

### top

Для работы с процессами удобно использовать top.

Результат ввода команды приведён на рисунке 15.

```
top - 20:27:20 up 3:31, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 96 total, 1 running, 94 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 : 3936.4 total, 3461.1 free, 136.8 used, 338.5 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 3577.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	102020	11676	8584	S	0.0	0.3	0:01.06	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	0:00.05	ksoftirqd/0
11	root	20	0	0	0	0	I	0.0	0.0	0:01.64	rcu_sched
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	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 15.

Интерактивные команды, которые вы можете выполнять во время работы программы:

- h - вывод справки по утилите;
- q или Esc - выход из top;
- A - выбор цветовой схемы;
- d или s - изменить интервал обновления информации;
- H - выводить потоки процессов;
- k - послать сигнал завершения процессу;
- W - записать текущие настройки программы в конфигурационный файл;
- Y - посмотреть дополнительные сведения о процессе, открытые файлы, порты, логи и т.д.;
- Z - изменить цветовую схему;
- l - скрыть или вывести информацию о средней нагрузке на систему;
- m - выключить или переключить режим отображения информации о памяти;

- x - выделять жирным колонку, по которой выполняется сортировка;
- y - выделять жирным процессы, которые выполняются в данный момент;
- z - переключение между цветным и одноцветным режимами;
- c - переключение режима вывода команды, доступен полный путь и только команда;
- F - настройка полей с информацией о процессах;
- o - фильтрация процессов по произвольному условию;
- u - фильтрация процессов по имени пользователя;
- V - отображение процессов в виде дерева;
- i - переключение режима отображения процессов, которые сейчас не используют ресурсы процессора;
- n - максимальное количество процессов, для отображения в программе;
- L - поиск по слову;
- <> - перемещение поля сортировки вправо и влево;

### Запуск процессов в supervisor

Создадим конфигурационный файл для запуска сценария test.sh. Конфигурационный файл приведён на рисунке 16

```
artem@ubuntu-server:~$ less /etc/supervisor/conf.d/test.conf
[program:test]
command=/home/artem/test.sh
autostart=true
autorestart=true
stderr_logfile=/home/artem/test_err.log
stdout_logfile=/home/artem/test_out.log
stopsignal=KILL
/etc/supervisor/conf.d/test.conf (END)
```

Рисунок 16.

Создав конфигурационный файл, нужно известить Supervisor о появлении новой программы; для этого используется команда supervisorctl. Сперва нужно просмотреть наличие новых файлов (рисунок 17), а затем уже активировать их (рисунок 18).

```
artem@ubuntu-server:~$ supervisorctl reread
error: <class 'PermissionError'>, [Errno 13] Permission denied: file: /usr/lib/python3/dist-packages/supervisor/xmlrpc.py line: 560
artem@ubuntu-server:~$ sudo supervisorctl reread
test: available
artem@ubuntu-server:~$
```

Рисунок 17.

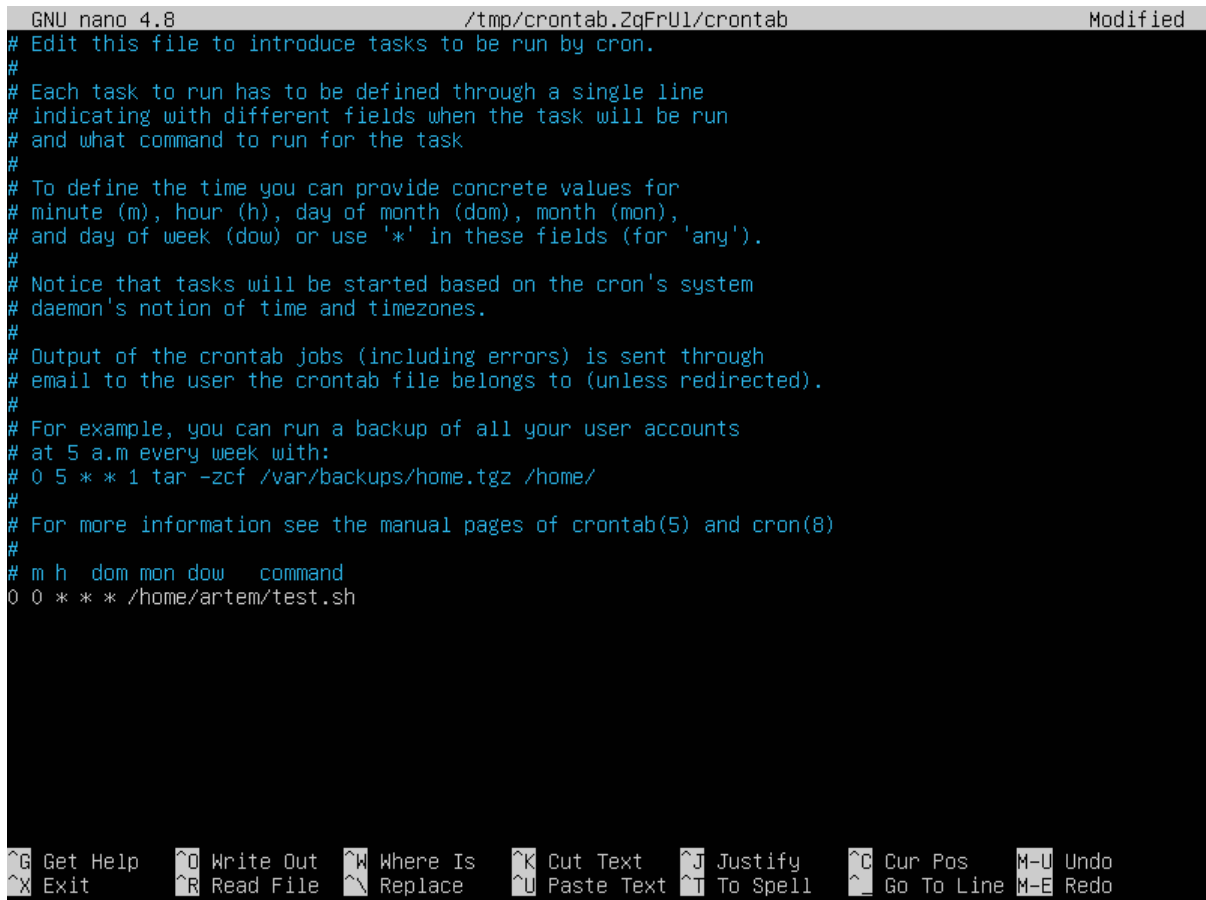
```
artem@ubuntu-server:~$ sudo supervisorctl update
test: added process group
artem@ubuntu-server:~$
```

Рисунок 18.



## Автоматический запуск по расписанию

Cron - это сервис, как и большинство других сервисов Linux, он запускается при старте системы и работает в фоновом режиме. Его основная задача выполнять нужные процессы в нужное время. Для работы с cron необходимо выполнить команду `crontab -e` (рисунок 19).



```
GNU nano 4.8 /tmp/crontab.ZqFrUl/crontab Modified
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
0 0 * * * /home/artem/test.sh
```

Рисунок 19.

Время задается особым синтаксисом. Синтаксис настройки одной задачи cron в crontab:

минута час день месяц день\_недели /путь\_к\_исполняемому\_файлу

Нужно сказать, что обязательно нужно писать полный путь к команде, потому что для команд, запускаемых от имени cron переменная среды `PATH` будет отличаться, и сервис просто не сможет найти вашу команду. Это вторая самая распространенная причина проблем с Cron. Дата и время указываются с помощью цифр или символа '\*'. Этот символ означает, что нужно выполнять каждый раз, если в первом поле - то каждую минуту и так далее. Ну а теперь перейдем к примерам.

Используя команду `crontab -l` проверим наличие поставленного на расписание процесса(рисунок 20).

```
artem@ubuntuserver:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
0 0 * * * /home/artem/test.sh
artem@ubuntuserver:~$ _
```

Рисунок 20.

## Вывод

В ходе лабораторной работы я на практике ознакомился с управлением процессами в ОС Ubuntu.

## Список литературы

- [1] Львовский, С.М. Набор и верстка в системе  $\text{\LaTeX}$  [Текст] / С.М. Львовский. М.: МЦНМО, 2006. — 448 с.
- [2] SEDICOMM. 30 полезных команд «ps» для мониторинга процессов Linux: <https://blog.sedicomm.com/2018/05/28/30-poleznyh-komand-ps-dlya-monitoringa-protsessov-linux/> (дата обращения: 29.10.2020). - Текст: электронный.