

Администрирование отечественных операционных систем
Практическая работа №5. Работа с текстовой информацией в ОС Astra
Linux

ДИСЦИПЛИНА	Администрирование отечественных операционных систем
ИНСТИТУТ	Институт перспективных технологий и индустриального программирования
КАФЕДРА	Цифровая кафедра
ВИД УЧЕБНОГО МАТЕРИАЛА	Практическая работа
ПРЕПОДАВАТЕЛЬ	Макиевский Станислав Евгеньевич
СЕМЕСТР	1 семестр, 2023-2024

Выполнил студент группы ИКБО-24-20 Ефимцев Станислав Максимович.



Цель работы: научиться работать с текстовой информацией, а также освоить перенаправление потоков в командной строке ОС Astra Linux.

Задание:

- научиться работать с тремя стандартными потоками с помощью известных утилит;
- научиться перенаправлять потоки;
- научиться работать с каналами;
- научиться работать с текстом в редакторе Vim.

1. Работа с потоками и перенаправление потоков

1.1. Познакомьтесь с теоретической информацией про потоки в ОС Linux.

Стандартный ввод при работе пользователя в терминале передается через клавиатуру.

Стандартный вывод и стандартная ошибка отображаются на дисплее терминала пользователя в виде текста.

Ввод и вывод распределяется между тремя стандартными потоками:

`stdin` – стандартный ввод (клавиатура),

`stdout` – стандартный вывод (экран),

`stderr` – стандартная ошибка (вывод ошибок на экран).

Потоки также пронумерованы:

`stdin` – 0,

`stdout` – 1,

`stderr` – 2.

Из стандартного ввода команда может только считывать данные, а два других потока могут использоваться только для записи. Данные выводятся на экран и считываются с клавиатуры, так как стандартные потоки по умолчанию ассоциированы с терминалом пользователя. Потоки можно подключать к чему угодно: к файлам, программам и даже устройствам. В командном интерпретаторе `bash` такая операция называется перенаправлением:

`< file` – использовать файл как источник данных для стандартного потока ввода.

`> file` – направить стандартный поток вывода в файл. Если файл не существует, он будет создан, если существует — перезаписан сверху.

`2> file` – направить стандартный поток ошибок в файл. Если файл не существует, он будет создан, если существует — перезаписан сверху.

`>>file` – направить стандартный поток вывода в файл. Если файл не существует, он будет создан, если существует — данные будут дописаны к нему в конец.

`2>>file` – направить стандартный поток ошибок в файл. Если файл не существует, он будет создан, если существует — данные будут дописаны к нему в конец.



Администрирование отечественных операционных систем

`&>file` или `>&file` – направить стандартный поток вывода и стандартный поток ошибок в файл. Другая форма записи: `>file 2>&1`.

Обозначения `stdin`, `stdout`, `stderr` представляют собой макросы, которые используются при написании программ на языке Си (и не только). См. `man stdin`. Также в каталоге `/dev` создаются символические ссылки с именами `stdin`, `stdout`, `stderr`, которые указывают на соответствующие файловые дескрипторы стандартных потоков текущего процесса.

Просмотрим эти ссылки с помощью команды: `sudo ls -l /dev/std*`

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):

```
efimtsev@stas:~$ sudo ls -l /dev/std*
[sudo] пароль для efimtsev:
lrwxrwxrwx 1 root root 15 ноя 18 23:43 /dev/stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root 15 ноя 18 23:43 /dev/stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root 15 ноя 18 23:43 /dev/stdout -> /proc/self/fd/1
```

1.2. Ознакомьтесь с теоретической информацией про стандартный поток ввода и выполните задание.

Стандартный входной поток обычно переносит данные от пользователя к программе. Программы, которые предполагают стандартный ввод, обычно получают входные данные от устройства типа клавиатура. Стандартный ввод прекращается по достижении EOF (конец файла), который указывает на то, что данных для чтения больше нет.

EOF вводится нажатием сочетания клавиш `Ctrl+D`.

Рассмотрим работу со стандартным выводом на примере команды `cat` (от `CONCATENATE`, в переводе «связать» или «объединить что-то»).

Эта команда обычно используется для объединения содержимого двух файлов и отправляет полученные входные данные на дисплей терминала в



Администрирование отечественных операционных систем

качестве стандартного вывода и останавливается после того, как получает EOF.

Задание: запустите команду `cat` без параметров. В открывшейся строке введите, например, «1» и нажмите клавишу Enter. Далее введите «a» и нажмите клавишу Enter.

Для завершения ввода данных следует нажать сочетание клавиш `Ctrl + D`.

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):

```
efimtsev@stas:~$ cat
1
1
a
a
efimtsev@stas:~$ ~
```

1.3. Ознакомьтесь с теоретической информацией про стандартный поток вывода и выполните задание.

Стандартный вывод записывает данные, сгенерированные программой. Когда стандартный выходной поток не перенаправляется в какой-либо файл, он выводит текст на дисплей терминала.

При использовании без каких-либо дополнительных опций, команда `echo` выводит на экран любой аргумент, который передается ему в командной строке.

Задание: введите команду `echo` и запишите для нее любой аргумент.

При выполнении `echo` без каких-либо аргументов, возвращается пустая строка.

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.



Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):

```
efimtsev@stas:~$ echo "My name is Slim Shady"  
My name is Slim Shady  
efimtsev@stas:~$
```

Задание: создайте (или используйте готовые) три файла и объедините их: file1, file2 и file3 в один файл bigfile:

```
cat file1 file2 file3 > bigfile
```

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):

```
efimtsev@stas:~$ echo "My name" > file1  
efimtsev@stas:~$ echo " is " > file2  
efimtsev@stas:~$ echo "Slim Shady" > file3  
efimtsev@stas:~$ cat file1 file2 file3 > bigfile  
efimtsev@stas:~$ cat bigfile  
My name  
 is  
Slim Shady  
efimtsev@stas:~$
```

Команда `cat` по очереди выводит содержимое файлов, перечисленных в качестве параметров на стандартный поток вывода. Стандартный поток вывода перенаправлен в файл `bigfile`.



Администрирование отечественных операционных систем

1.4. Ознакомьтесь с теоретической информацией про стандартную ошибку и выполните задание.

Стандартная ошибка записывает ошибки, возникающие в ходе исполнения программы. Как и в случае стандартного вывода, по умолчанию этот поток выводится на терминал дисплея.

Рассмотрим пример стандартной ошибки с помощью команды `ls`, которая выводит список содержимого каталогов. При запуске без аргументов `ls` выводит содержимое в пределах текущего каталога.

Задание: введем команду `ls` с каталогом `%` в качестве аргумента:

В результате должно выводиться содержимое соответствующей папки. Но так как каталога `%` не существует, на дисплей терминала будет выведен текст стандартной ошибки.

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):

```
efimtsev@stas:~$ ls %  
ls: невозможно получить доступ к '%': Нет такого файла или каталога  
efimtsev@stas:~$
```

Задание: перенаправьте поток ошибок в устройство `/dev/null`.
`cat /etc/* 2> /dev/null`

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):



```
#
# The following definitions allow abstract device names.  They are used if
# the
# device name does not contain the the characters ',', ':', '/' and '@'
#
# Unless you have a good reason, use speed == -1 and let wodim use its internal
# drive specific defaults.
#
# drive name device      speed  fifosize driveropts
#
#default=    USCSI:1,0,0    -1 -1    burnfree
#sanyo=       1,4,0        -1  -1 burnfree
#cdrom=       0,6,0         2   1m ""
#remote=      REMOTE:rscsi@somehost:1,0,0    16 32m  burnfree
#
cdrom=        -1          -1    -1    burnfree
efimtsev@stas:~$
```

1.5. Ознакомьтесь с теоретической информацией про перенаправление потоков и выполните задание.

Linux включает в себя команды перенаправления для каждого потока.

Команды со знаками > или < означают перезапись существующего содержимого файла:

- > – стандартный вывод,
- < – стандартный ввод,
- 2> – стандартная ошибка.

Команды со знаками >> или << не перезаписывают существующее содержимое файла, а присоединяют данные к нему:

- >> – стандартный вывод,
- << – стандартный ввод,
- 2>> – стандартная ошибка.

Задание: с помощью команды cat перезапишите содержимое файла file1, который создается в результате цикла (сделайте ввод нескольких строк). Далее завершите ввод и дополните содержимое файла file1. Просмотрите итоговое содержимое файла.

```
cat > file1
cat >> file1
```

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):



```
efimtsev@stas:~$ cat > file1
Файл
Машина
Bug
efimtsev@stas:~$ cat >> file1
ВВог
Устройство
Справка
efimtsev@stas:~$ cat file1
Файл
Машина
Bug
ВВог
Устройство
Справка
efimtsev@stas:~$
```

1.6. Ознакомьтесь с теоретической информацией про каналы и выполните задание.

Каналы используются для перенаправления потока из одной программы в другую. Стандартный вывод данных после выполнения одной команды перенаправляется в другую через канал. Данные первой программы, которые получает вторая программа, не будут отображаться. На дисплей терминала будут выведены только отфильтрованные данные, возвращаемые второй командой.

Синтаксис: команда1 | команда2 |... | командаN

Задание: введите команду для просмотра содержимого директории, используя постраничный вывод.

`ls | less`

В результате каждый файл текущего каталога будет размещен на новой строке.

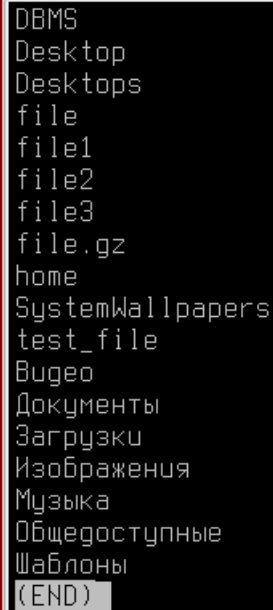
Перенаправлять данные с помощью каналов можно как из одной команды в другую, так и из одного файла к другому, а перенаправление с помощью `>` и `>>` возможно только для перенаправления данных в файлах.

При использовании нескольких фильтров в одной команде рекомендуется разделять их с помощью знака канала «`|`».

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):





```
DBMS
Desktop
Desktops
file
file1
file2
file3
file.gz
home
SystemWallpapers
test_file
Bugeo
Документы
Загрузки
Изображения
Музыка
Общедоступные
Шаблоны
(END)
```

2. Работа с текстом в редакторе vim

`vi` был одним из первых визуальных текстовых редакторов, однако начиная с определенной версии программный продукт стал доступен только под коммерческой лицензией. Энтузиасты начали разрабатывать свои аналоги этого текстового редактора, одним из которых является `vim` (сокращение от `vi Improved` – `vi` усовершенствованный). `vim` получил широкое распространение в Linux-дистрибутивах и при запуске команды `vi` в консоли вероятнее всего откроется именно он.

2.1. Запустите текстовый редактор `vim` и введите новое имя файла `vim_file`. Введите несколько строчек любого текста.

```
vim vim_file
```

Примечание: `vim +25 имя_файла` – открыть файл и переместить курсор на 25 строку (удобно использовать, например, при отладке служб, когда сообщается, какая строка в конфигурационном файле привела к ошибке).

Переход в командный режим – ESC.

Переход в режим редактирования:

`i` – начать вставку текста перед текущим символом.

`I` – перейти в начало строки и начать вставку текста.

`a` – начать вставку текста после текущего символа.

`A` – перейти в конец строки и начать вставку текста.

Навигация:

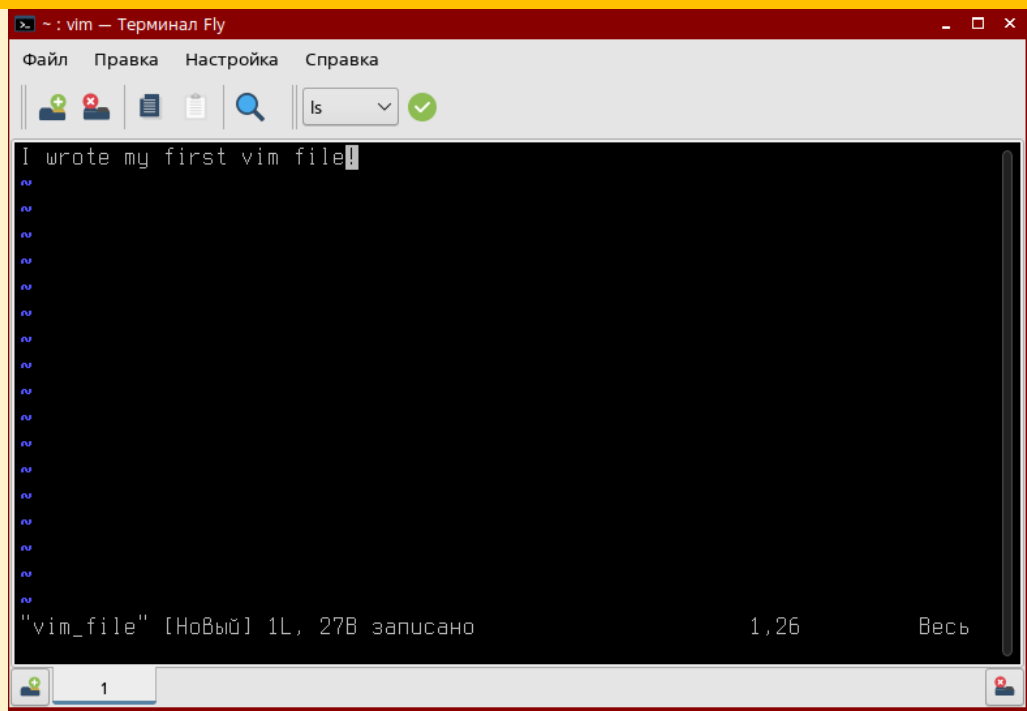
`h` – смещение курсора на один символ влево.



Администрирование отечественных операционных систем

- j – сдвиг курсора на один символ вниз.
- k – сдвиг курсора на один символ вверх.
- l – сдвиг курсора на один символ вправо.
- w – сдвиг курсора в начало следующего слова в данной строке.
- b – сдвиг курсора в начало предыдущего слова в данной строке.
- G – переместить курсор в конец файла.
- gg – вернуться в начало файла.

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.
Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):



2.2. Освойте редактирование текста в vim.

Задание: удалите и замените часть текста с помощью командного режима, используя различные клавиши для быстрого удаления и замены.

Часто используемые операции с текстом:

x – удаление текущего символа (клавиша Delete).

X – удаление предыдущего символа (клавиша BackSpace).

R – режим замены, все вводимые символы будут последовательно заменять находящиеся под курсором.

Если нужно заменить всего один символ, находящийся под курсором, достаточно нажать r и нужный символ – замена будет произведена без



Администрирование отечественных операционных систем

перехода в режим редактирования. Внизу экрана появляется соответствующий индикатор режима – INSERT или REPLACE. Вернуться в общий режим можно нажатием кнопки ESC (индикатор должен пропасть).

`d` – удалить символы/строки.

`dd` – удаление строки (перед командами можно указывать число, которое определяет, сколько раз будет повторена команда). Например, `3dd` – удаление 3 строк, начиная с текущей.

`D` – удалить символы от текущего положения курсора до конца строки.

Примечание: при нажатии `d` или `y` ничего не происходит. Дело в том, что редактор ожидает дальнейших указаний – сколько строк символов нужно удалить/скопировать и в каком направлении от текущего положения курсора вести отсчет. Чтобы удалить пять символов вправо от курсора, нужно последовательно нажать `d5l`.

Задание: скопируйте несколько начальных символов или строк и вставьте их в конец файла.

Дополнительные операции с текстом:

`v` – включить режим выделения текста.

`y` – скопировать символы/строки.

`yy` – скопировать строку целиком.

`p` – вставить символы/строки.

`u` – отменить последнее действие.

`student@prac-work-question:~#` Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только `git`):

```
I wrote my fvcxm file!ttt
I wrote my fvcxm file!ttt
I wrote my fvcxm file!ttt
~
~
~
~
~
~
```



2.3. Сохраните содержимое файла и осуществите выход из редактора.

Некоторые командные режимы:

:q – выйти из редактора

:q! – выйти из редактора без сохранения изменений в файле.

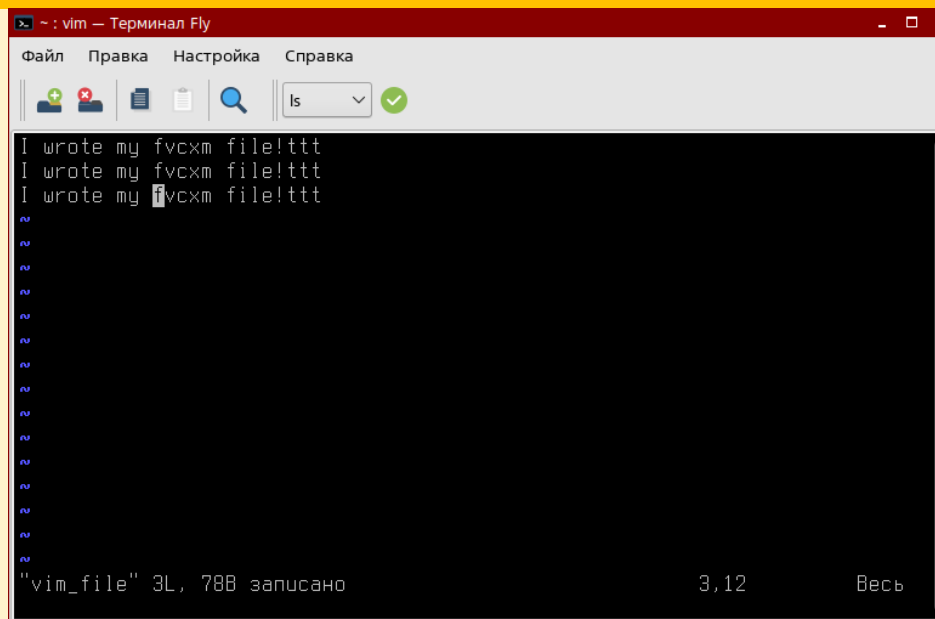
:w – сохранить изменения в файл.

:wq – сохранить изменения и закрыть редактор.

:r имя_файла – добавить содержимое указанного файла в редактируемый.

:r! команда_консоли – выполнить команду и добавить ее вывод в файл.

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.
Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):



```
efimtsev@stas:~$ vim vim_file
efimtsev@stas:~$
```

2.4. Снова откройте файл в редакторе и осуществите поиск и замену любого символа. Далее выйдите из редактора без сохранения изменений в файле.



Администрирование отечественных операционных систем

/регулярное_выражение – поиск вперед.

?регулярное_выражение – поиск назад.

n – продолжить поиск вперед.

N – продолжить поиск назад.

:%s/регулярное_выражение/замена/g – найти и заменить (% – диапазон поиска – весь файл, g – продолжить поиск регулярного выражения в строке, если в этой строке нашлась последовательность символов, соответствующая регулярному выражению).

Схема записи регулярных выражений:

^	начало строки	*	предыдущий символ повторяется 0 и более раз
\$	конец строки	+	предыдущий символ повторяется 1 и более раз
.	любой символ	?	предыдущий символ повторяется 0 или 1 раз
[символы]	один символ из диапазона	{n,m}	предыдущий символ повторяется от n до m раз
[^символы]	один символ из не диапазона	{n,}	предыдущий символ повторяется n и более раз
символы символы	или	{n}	предыдущий символ повторяется n раз
(символы)	группировка символов	\	отмена специального значения последующего метасимвола
\<слово>	слово		

Примеры регулярных выражений:

^\$ – пустая строка.

^# – строка, которая начинается с символа # (комментарии).

\<bugs?\> – строки, содержащие слова bug или bugs.

bash\$ – строка, заканчивающаяся на bash.

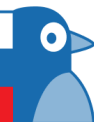
^[A-Z] – строка, начинающаяся с заглавного символа.

#|// – строка, содержащая или символ # или два подряд идущих символа /.

\<twent(y|ies)\> – строки, содержащие слова twenty или twenties.

Примечание: более подробно регулярные выражения будут разбираться в практике по созданию сценариев bash.

Чтобы закрепить навыки работы с редактором, освежить в памяти его возможности или узнать что-то новое, можно в терминале набрать команду vimtutor.



[illegible][illegible]

Администрирование отечественных операционных систем

- `-r` – сортировка по убыванию;
- `-k` список_полей – сортировка по указанным полям (номера полей, можно использовать запятую для указания нескольких полей);
- `-t` символ – определение разделителя между полями.

Пример: показ учетных записей пользователей, отсортированных по уменьшению UID:

```
cat /etc/passwd | sort -t: -k 3 -nr
```

Основные параметры команды `cut`:

- `-c` диапазон_символов – вырезание указанного диапазона символов. Для указания диапазона используется символ минус;
- `-f` список_полей – вырезание указанных полей;
- `-d` символ – определение разделителя между полями.

Пример: модификация предыдущего примера. Выводятся только первое (имя учетной записи пользователя) и третье (UID) поля.

```
cat /etc/passwd | sort -t: -k 3 -nr | cut -d: -f 1,3
```

Чтобы использовать команду `uniq`, строки в файле (или в стандартном потоке) должны быть предварительно отсортированы.

Основные параметры команды `uniq`:

- если команда используется без параметров, то убираются все повторы строк;
- `-u` – остаются только уникальные строки (строки, которые имели повторы, удаляются вместе с повторами);
- `-d` – выводятся только строки, которые имели повторы.

Пример: выводится список пользователей, которые владеют файлами в каталоге `/tmp` и во всех подкаталогах `/tmp`, с указанием количества файлов (параметр `-c` команды `uniq` подсчитывает кол-во повторов).

```
sudo ls -lR /tmp | grep '^-' | cut -d" " -f3 | sort |  
uniq -c
```

Команда `tr` позволяет символы из одного набора заменять на символы из другого набора (замена осуществляется символ на символ).

Пример: строчные символы заменяются на прописные.

```
who | tr 'a-z' 'A-Z'
```

Параметр `-d` команды `tr` используется для удаления указанных символов.

Пример: в текстовом файле, полученном из ОС Windows, удаляется символ возврата каретки (`\r`). Текстовые строки в Linux завершаются символом `\n` (новая строка, new line), а в Windows – двумя `\n\r`.

```
cat dosfile.txt | tr -d '\r' > linuxfile.txt
```



Администрирование отечественных операционных систем

Основные параметры команды `xargs`:

- `-I` символы или `-i` символы – вместо указанных символов подставляются данные, полученные из стандартного входного потока (по умолчанию данные из входного потока помещаются в конец командной строки);

- `-p` – каждое сформированное командой `xargs` действие будет требовать подтверждения (удобно использовать для отладки).

Пример: для всех файлов в текущем каталоге, имена которых начинаются с `file`, создаются их копии с именами, к которым добавляется суффикс `.bak`.

```
ls file* | xargs -i[] cp [] [].bak
```

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):

```
efimtsev@stas:~$ vim vim_file
efimtsev@stas:~$ ps -e | wc -l
138
```

```
efimtsev@stas:~$ cat /etc/passwd | sort -t: -k 3 -nr
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
efimtsev:x:1000:1000:efimtsev,,,:/home/efimtsev:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper::/usr/sbin/nologin
avahi:x:114:126:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
logcheck:x:113:125:logcheck system account,,,:/var/lib/logcheck:/usr/sbin/nologin
hplip:x:112:7:HPLIP system user,,,:/run/hplip:/bin/false
nm-openvpn:x:111:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
Debian-exim:x:110:122::/var/spool/exim4:/usr/sbin/nologin
ntp:x:109:121::/nonexistent:/usr/sbin/nologin
sshd:x:108:65534::/run/sshd:/usr/sbin/nologin
fly-dm:x:107:118::/var/lib/fly-dm:/usr/sbin/nologin
pulse:x:106:115:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
messagebus:x:105:111::/nonexistent:/usr/sbin/nologin
astra-orientation:x:104:110::/var/cache/astra-orientation:/usr/sbin/nologin
```



```
efimtsev@stas:~$ cat /etc/passwd | sort -t: -k 3 -nr | cut -d: -f 1,3
nobody:65534
efimtsev:1000
systemd-coredump:999
avahi:114
logcheck:113
hplip:112
nm-openvpn:111
Debian-exim:110
ntp:109
sshd:108
fly-dm:107
pulse:106
messagebus:105
astra-orientation:104
_apt:103
systemd-resolve:102
systemd-network:101
systemd-timesync:100
```

```
efimtsev@stas:~$ sudo ls -lR /tmp | grep '^-' | cut -d" " -f3 | sort | uni
q -c
[sudo] пароль для efimtsev:
    1 astra-orientation
    1 root
efimtsev@stas:~$
```

```
efimtsev@stas:~$ who | tr 'a-z' 'A-Z'
EFIMTSEV :0          2023-11-18 23:44
EFIMTSEV PTS/0      2023-11-18 23:44 (:0)
efimtsev@stas:~$
```

```
efimtsev@stas:~$ cat dosfile.txt | tr -d '\r' > linuxfile.txt
efimtsev@stas:~$ cat linuxfile.txt
Привет, как дела?\n\r
- Привет, все отлично. Ты как?\n\r
У меня тоже все хорошо, спасибо)\n\r
efimtsev@stas:~$
```

```
efimtsev@stas:~$ ls file* | xargs -i[] cp [] []bak
efimtsev@stas:~$ ls
AstraLinux  Desktops  file3      SystemWallpapers  Музыка
astra.tar   dosfile.txt file3.bak   test_file         Общедоступные
astra.tar.gz file       file.bak   vim_file          Шаблоны
bigfile     file1     file.gz    Bugeo
Cisco       file1.bak file.gz.bak Документы
DBMS        file2     home       Загрузки
Desktop     file2.bak linuxfile.txt Изображения
efimtsev@stas:~$
```



Администрирование отечественных операционных систем

3.2. Изучите текстовые редакторы `grep`, `sed`, `awk` при конвейерной обработке текстовых потоков и сделайте примеры ниже.

Утилита `grep` осуществляет поиск строк, соответствующих регулярному выражению, в текстовых потоках.

Формат использования:

`grep [параметры] 'регулярное_выражение' [файлы]`

Команды из семейства `grep`:

- `grep` – поиск с использованием базового набора метасимволов регулярных выражений;

- `egrep` (`grep -E`) – поиск с использованием расширенного набора метасимволов регулярных выражений;

- `fgrep` (`grep -F`) – поиск по «фиксированной» строке (специальные значения метасимволов игнорируются).

Утилита `grep` часто используется в конвейерной обработке текстовых потоков с использованием неименованных каналов (`pipe`).

```
ps -ef | grep fly-fm
```

Часто используемые параметры команд `grep`, `egrep`, `fgrep`:

- `-i` – игнорирование регистра символов;

- `-v` – поиск на несоответствие регулярному выражению;

- `-r` – рекурсивный поиск;

- `-l` – выводятся только имена файлов, в которых есть строки, соответствующие регулярному выражению;

- `-H` – перед каждой найденной строкой выводится префикс – имя файла.

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только `git`):

```
efimtsev@stas:~$ ps -ef | grep fly-fm
efimtsev 1723 1211 0 00:31 pts/0    00:00:00 grep fly-fm
efimtsev@stas:~$
```



Администрирование отечественных операционных систем

Утилита `sed` используется для редактирования текстовых потоков.

Формат утилиты `sed`:

```
sed [параметры] 'правила_отбора инструкции' [файлы]
```

Правила отбора:

- по номеру строк, например:

- `1,10` – с первой по десятую строку;

- `10,$` – с десятой по последнюю строку;

- по соответствию регулярному выражению (`/regexp/`).

Инструкции редактирования (`p, d, i, a, s, c`).

Для использования расширенного набора метасимволов регулярных выражений следует установить параметр `-E`.

Часто используемые параметры `sed`:

- `-i` – изменения производятся в указанном в качестве аргумента файле (`--in-place`);

- `-e` – используется, если в одной команде нужно указать несколько инструкций редактирования. Параметр `-e` размещается перед каждой инструкцией;

- `-E` – используется расширенный вариант регулярных выражений.

Примеры использования инструкций редактирования:

- `p` – вывод (печать, `print`) строки, часто используется с параметром `-n`, который «подавляет» происходящий по умолчанию вывод всех строк.

Примеры:

- 1) выводятся первые десять строк, полученных от `ps`:

```
ps -el | sed -n '1,10 p'
```

- 2) выводятся процессы ядра:

```
ps -ef | sed -E -n '/[0-9] \[.+\]$/' p'
```

- `d` – удаление. Пример: выводятся первые десять строк, полученных от `ps`

```
ps -el | sed '11,$ d'
```

- `i` текст – вставка текста (перед найденной строкой);

- `a` текст – добавление текста (после найденной строки);

Пример: вставка «разделителей» перед и после первой строки

```
ps -el | head | sed -e '1 i-----' -e '1 a-----'
```

- `s` текст – замена найденной строки на текст;

- `s/поиск/замена/` – осуществление поиска и замены

Пример: в файле `passwd` символы `/sh`, находящиеся в конце строки, заменяются на `/bash`. Поиск и замена помещаются внутри `#` (решеток), поскольку в строках поиска и замены используется символ `/` (косая черта)

```
sed -i 's#/sh$#/bash#' passwd
```



Администрирование отечественных операционных систем

сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):



Администрирование отечественных операционных систем

```
efimtsev@stas:~$ ps -el | sed -n '1,10 p'
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	0	1	0	0	80	0	-	25762	-	?	00:00:03	systemd
1	S	0	2	0	0	80	0	-	0	-	?	00:00:00	kthreadd
1	I	0	3	2	0	60	-20	-	0	-	?	00:00:00	rcu_gp
1	I	0	4	2	0	60	-20	-	0	-	?	00:00:00	rcu_par_gp
1	I	0	6	2	0	60	-20	-	0	-	?	00:00:00	kworker/0:0H-kblockd
1	I	0	9	2	0	60	-20	-	0	-	?	00:00:00	mm_percpu_wq
1	S	0	10	2	0	80	0	-	0	-	?	00:00:00	rcu_task
1	S	0	11	2	0	80	0	-	0	-	?	00:00:00	rcu_task
1	S	0	12	2	0	80	0	-	0	-	?	00:00:00	ksoftirqd/0

```
efimtsev@stas:~$
```

```
efimtsev@stas:~$ ps -ef | sed -E -n '/[0-9] \[.+\]/ p'
```

root	2	0	0	ноя18	?	00:00:00	[kthreadd]
root	3	2	0	ноя18	?	00:00:00	[rcu_gp]
root	4	2	0	ноя18	?	00:00:00	[rcu_par_gp]
root	6	2	0	ноя18	?	00:00:00	[kworker/0:0H-kblockd]
root	9	2	0	ноя18	?	00:00:00	[mm_percpu_wq]
root	10	2	0	ноя18	?	00:00:00	[rcu_tasks_rude_]
root	11	2	0	ноя18	?	00:00:00	[rcu_tasks_trace]
root	12	2	0	ноя18	?	00:00:00	[ksoftirqd/0]
root	13	2	0	ноя18	?	00:00:01	[rcu_sched]
root	14	2	0	ноя18	?	00:00:00	[migration/0]
root	15	2	0	ноя18	?	00:00:00	[idle_inject/0]
root	16	2	0	ноя18	?	00:00:00	[cpuhp/0]
root	17	2	0	ноя18	?	00:00:00	[cpuhp/1]
root	18	2	0	ноя18	?	00:00:00	[idle_inject/1]
root	19	2	0	ноя18	?	00:00:00	[migration/1]
root	20	2	0	ноя18	?	00:00:00	[ksoftirqd/1]
root	22	2	0	ноя18	?	00:00:00	[kworker/1:0H-events_highpri]
root	23	2	0	ноя18	?	00:00:00	[kdevtmpfs]

```
efimtsev@stas:~$ ps -el | sed '11,$ d'
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	0	1	0	0	80	0	-	25762	-	?	00:00:03	systemd
1	S	0	2	0	0	80	0	-	0	-	?	00:00:00	kthreadd
1	I	0	3	2	0	60	-20	-	0	-	?	00:00:00	rcu_gp
1	I	0	4	2	0	60	-20	-	0	-	?	00:00:00	rcu_par_gp
1	I	0	6	2	0	60	-20	-	0	-	?	00:00:00	kworker/0:0H-kblockd
1	I	0	9	2	0	60	-20	-	0	-	?	00:00:00	mm_percpu_wq
1	S	0	10	2	0	80	0	-	0	-	?	00:00:00	rcu_task
1	S	0	11	2	0	80	0	-	0	-	?	00:00:00	rcu_task
1	S	0	12	2	0	80	0	-	0	-	?	00:00:00	ksoftirqd/0

```
efimtsev@stas:~$
```



```
efimtsev@stas:~$ ps -el | head | sed -e '1 i-----' -e '1 a-----'
-----
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
-----
4 S   0    1    0  0  80   0 - 25762 -      ?         00:00:03 systemd
1 S   0    2    0  0  80   0 -      0 -      ?         00:00:00 kthreadd
1 I   0    3    2  0  60  -20 -      0 -      ?         00:00:00 rcu_gp
1 I   0    4    2  0  60  -20 -      0 -      ?         00:00:00 rcu_par_gp
1 I   0    6    2  0  60  -20 -      0 -      ?         00:00:00 kworker/0:0H-kblockd
1 I   0    9    2  0  60  -20 -      0 -      ?         00:00:00 mm_percpu_wq
1 S   0   10    2  0  80   0 -      0 -      ?         00:00:00 rcu_task
1 S   0   11    2  0  80   0 -      0 -      ?         00:00:00 rcu_task
s_trace
```

```
efimtsev@stas:~$ sed -i 's#/sh$#/bash#' passwd
efimtsev@stas:~$
```

Утилита `awk` предназначена для построчной обработки текстовых потоков.

Формат использования `awk`:

`awk [параметры] 'правила_отбора{действия}' [файлы]`

Утилита `awk` умеет работать с полями строк с помощью встроенных переменных (по умолчанию поля разделяются любым количеством пробелов или табуляций, можно задать разделитель с помощью `-F`):

- `$0` – вся строка
- `$1`, `$2`, ... - первое, второе, ... поле
- `NF`, `NR` – количество полей в строке, номер строки

В качестве действий можно использовать команды `print` или `printf`.

Утилита `awk` может использоваться для сложной обработки текстов, составления отчетов, проведения анализа текстовой информации, а также для составления набора команд с последующей передачей их на выполнение командным интерпретатором `bash`.

В качестве правил отбора можно использовать:

– `/регулярное_выражение/` – соответствие строки регулярному выражению. Пример: вывод списка учетных записей пользователей, начинающихся с `u`

```
getent passwd | awk -F ':' '/^u/{print $1}'
```

– `$номер_поля операция_сравнения значение` – выполнение условия для конкретных полей. Операции сравнения могут арифметические (`>`, `>=`, `<`, `<=`, `==`, `!=`), строковые (`==`, `!=`), соответствие регулярному выражению (`~`, `!~`).

Примеры:



Администрирование отечественных операционных систем

1) Вывод учетных записей «обычных» пользователей (с UID ≥ 1000 и исключая пользователя nobody

```
getent passwd | \
awk -F ':' '$3 >= 1000 && $1 != "nobody" {print $1, $3}'
```

2) Вывод терминальных пользователей (у которых имя командного интерпретатора заканчивается на sh). NF в данном случае используется для указания последнего поля в строке.

```
getent passwd | awk -F ':' '$NF ~ /sh$/ {print $1, $NF}'
```

student@prac-work-question:~# Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.

Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git):

```
efimtsev@stas:~$ getent passwd | awk -F ':' '$3 >= 1000 && $1 != "nobody" {print $1, $3}'
efimtsev 1000
```

```
efimtsev@stas:~$ getent passwd | awk -F ':' '$NF ~ /sh$/ {print $1, $NF}'
root /bin/bash
efimtsev /bin/bash
```

