

Red Hat System Administration I

UNIT 4

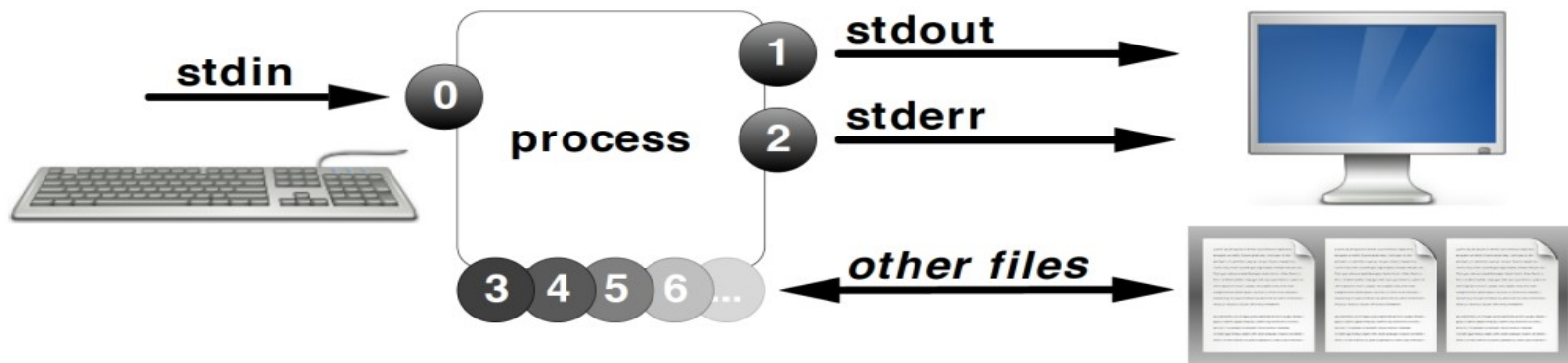
Creating, Viewing, and Editing Text Files

Objectives

- Redirect the text output of a program to a file or to another program.
- Edit existing text files and create new files from the shell prompt with a text editor.
- Copy text from a graphical window to a text file using a text editor running in the graphical environment.

Standard input, standard output, and standard error




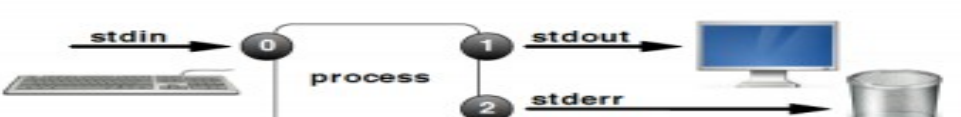


- Channels (File Descriptors)



Number	Channel name	Description	Default connection	Usage
0	stdin	Standard input	Keyboard	read only
1	stdout	Standard output	Terminal	write only
2	stderr	Standard error	Terminal	write only
3+	<i>filename</i>	Other files	<i>none</i>	read and/or write

Redirecting output to a file

- Output Redirection Operators

Usage	Explanation (note)	Visual aid
<code>>file</code>	redirect stdout to a file ⁽¹⁾	
<code>>>file</code>	redirect stdout to a file, append to current file content ⁽²⁾	
<code>2>file</code>	redirect stderr to a file ⁽¹⁾	
<code>2>/dev/null</code>	discard stderr error messages by redirecting to /dev/null	
<code>&>file</code>	combine stdout and stderr to one file ⁽¹⁾	
<code>>>file 2>&1</code>	combine stdout and stderr , append to current file content ^{(2) (3)}	

Examples for output redirection

```
[student@desktopX ~]$ date > /tmp/saved-timestamp
```

```
[student@desktopX ~]$ tail -n 100 /var/log/dmesg > /tmp/last-100-boot-messages
```

```
[student@desktopX ~]$ cat file1 file2 file3 file4 > /tmp/all-four-in-one
```

```
[student@desktopX ~]$ ls ~/.* > /tmp/my-configuration-file-names
```

```
[student@desktopX ~]$ echo "new line of information" >> /tmp/many-lines-of-information
```

```
[student@desktopX ~]$ diff previous-file current-file >> /tmp/tracking-changes-made
```

```
[student@desktopX ~]$ find /etc -name passwd 2> /tmp/errors
```

```
[student@desktopX ~]$ find /etc -name passwd > /tmp/output 2> /tmp/errors
```

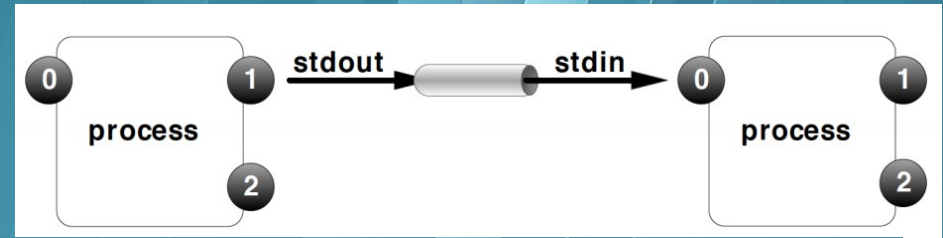
```
[student@desktopX ~]$ find /etc -name passwd > /tmp/output 2> /dev/null
```

```
[student@desktopX ~]$ find /etc -name passwd &> /tmp/save-both
```

```
[student@desktopX ~]$ find /etc -name passwd >> /tmp/save-both 2>&1
```


Constructing pipelines

- Redirection controls channel output to or from files while piping sends channel output to another process

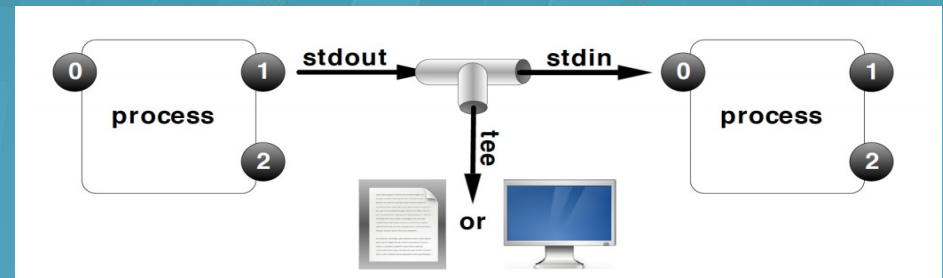


- Example

```
[student@desktopX ~]$ ls -l /usr/bin | less
```

```
[student@desktopX ~]$ ls | wc -l > /tmp/how-many-files
```

```
[student@desktopX ~]$ ls -t | head -n 10 > /tmp/ten-last-changed-files
```



```
[student@desktopX ~]$ ls -l | tee /tmp/saved-output
```

```
[student@desktopX ~]$ tty  
/dev/pts/0
```

```
[student@desktopX ~]$ ls -l | tee /dev/pts/0 | mail -s subject  
student@desktop1.example.com
```

Practice: I/O Redirection and Pipelines Quiz

Questions

1. Display command output to terminal, ignore all errors.
2. Send command output to file; errors to different file.
3. Send output and errors to the same new, empty file.
4. Send output and errors to the same file, but preserve existing file content.
5. Run a command, but throw away all possible terminal displays.
6. Send command output to both the screen and a file at the same time.
7. Run command, save output in a file, discard error messages.

Answer

Result needed	Redirection syntax used
Display command output to terminal, ignore all errors.	<code>2>/dev/null</code>
Send command output to file; errors to different file.	<code>>file 2>file2</code>
Send output and errors to the same new, empty file.	<code>&>file</code>
Send output and errors to the same file, but preserve existing file content.	<code>>>file 2>&1</code>
Run a command, but throw away all possible terminal displays.	<code>&>/dev/null</code>
Send command output to both the screen and a file at the same time.	<code> tee file</code>
Run command, save output in a file, discard error messages.	<code>> file 2> /dev/null</code>

Editing Text Files from the Shell Prompt

- **Vim**
 - Vim is an improved version of the vi editor distributed with Linux and UNIX systems. Vim is highly configurable and efficient for practiced users, including such features as split screen editing, color formatting, and highlighting for editing text.

Moving between Vim modes

- **command mode**
- **insert mode**
- **exit mode**

Command mode

- `:help`
- `:set`
- `ctrl + v`
- `ctrl+w -s`
- `ctrl+w -v`
- `ctrl+w -c`
- `:%s/words/replace/g`
- `y`
 - `yy,y[x]y`
- `d`
 - `dd,d[x]d`
- `c`
 - `cc,c[x]c`
- `p`

Insert mode

- A
- a
- I
- i

- O
- o
- S
- s

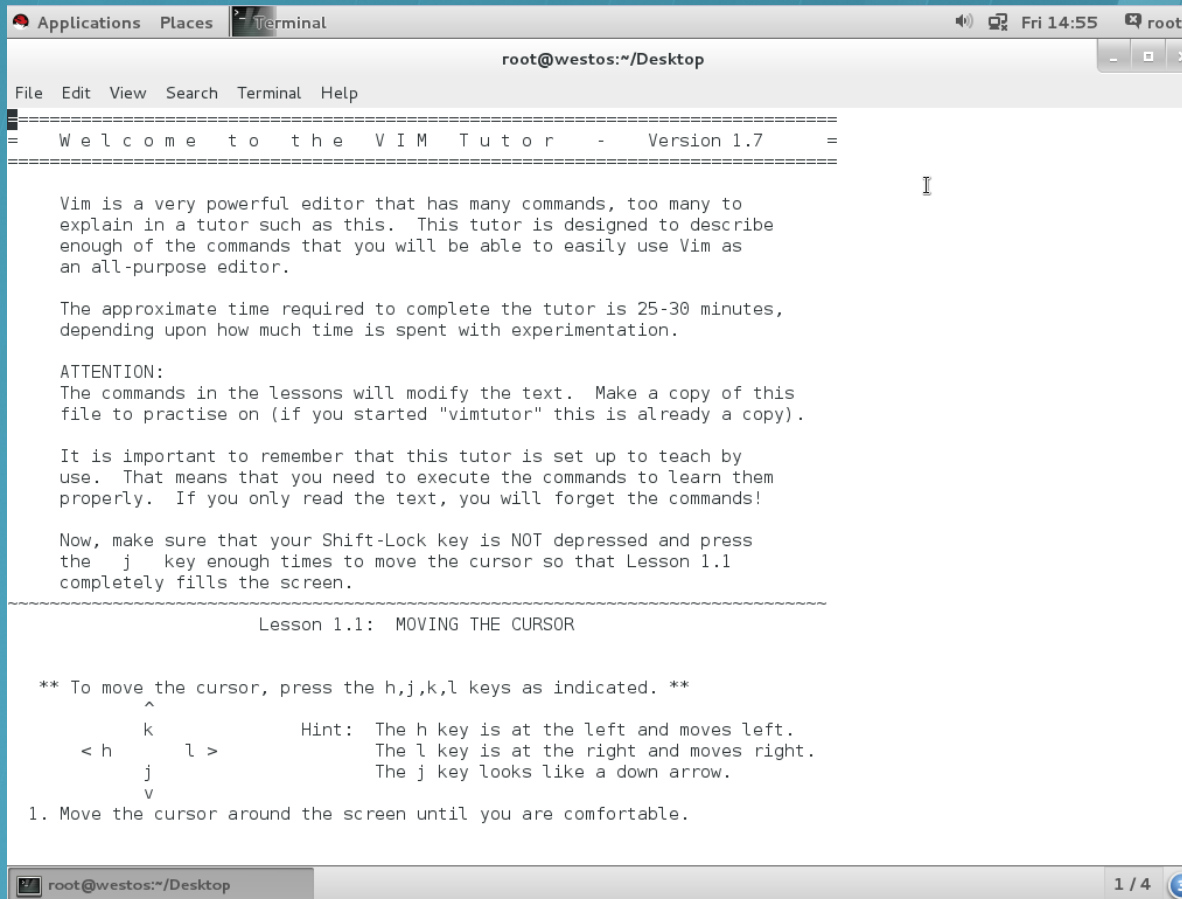
Exit mode

- `:q`
- `:q!`
- `:wq`
- `:wq!`

The help for vim

- **vimtutor**

[student@serverX ~]\$vimtutor



```
Applications Places Terminal
root@westos:~/Desktop
File Edit View Search Terminal Help
=====
Welcome to the VIM Tutor - Version 1.7
=====

Vim is a very powerful editor that has many commands, too many to
explain in a tutor such as this. This tutor is designed to describe
enough of the commands that you will be able to easily use Vim as
an all-purpose editor.

The approximate time required to complete the tutor is 25-30 minutes,
depending upon how much time is spent with experimentation.

ATTENTION:
The commands in the lessons will modify the text. Make a copy of this
file to practise on (if you started "vimtutor" this is already a copy).

It is important to remember that this tutor is set up to teach by
use. That means that you need to execute the commands to learn them
properly. If you only read the text, you will forget the commands!

Now, make sure that your Shift-Lock key is NOT depressed and press
the j key enough times to move the cursor so that Lesson 1.1
completely fills the screen.

-----
Lesson 1.1: MOVING THE CURSOR

** To move the cursor, press the h,j,k,l keys as indicated. **
      ^
      k
    < h   l >      Hint: The h key is at the left and moves left.
      j          The l key is at the right and moves right.
      v          The j key looks like a down arrow.

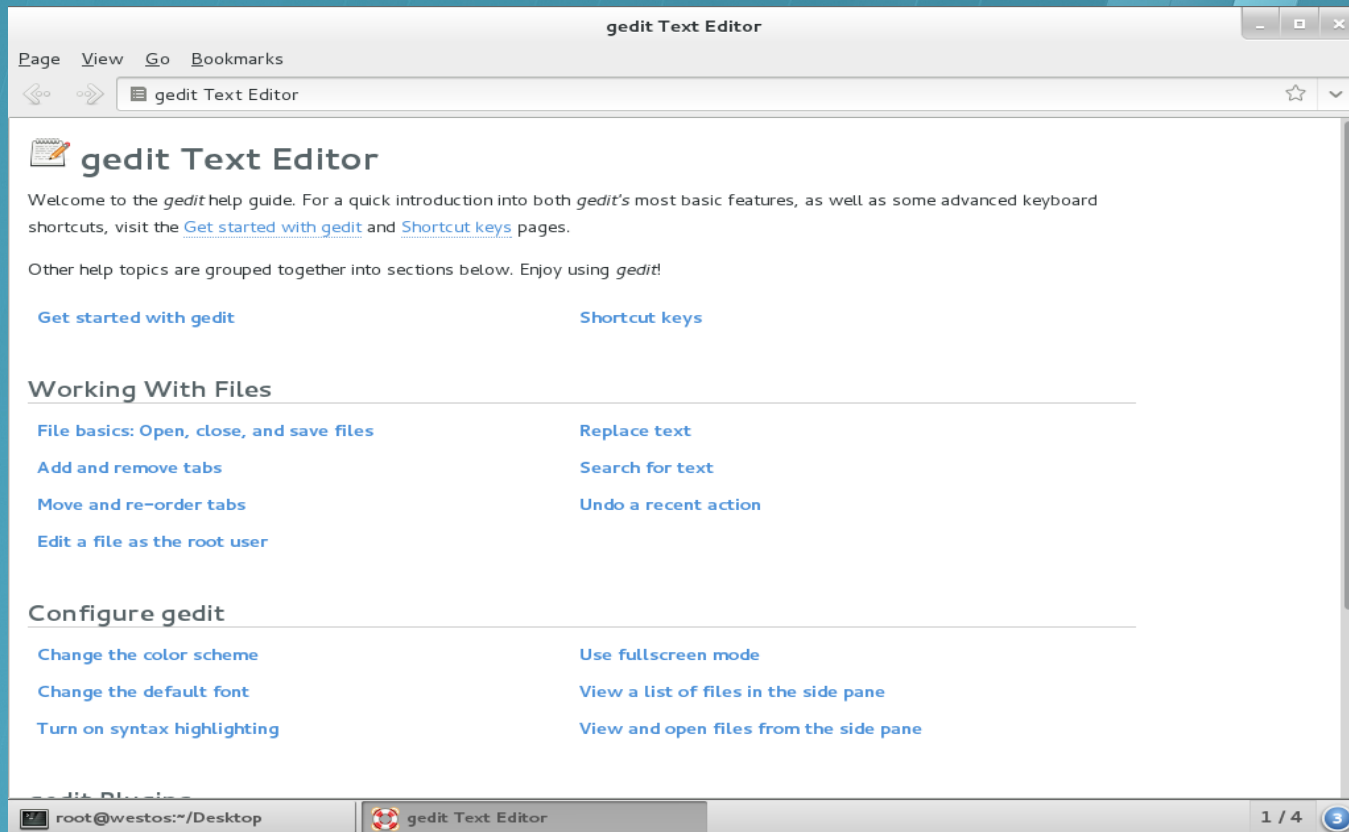
1. Move the cursor around the screen until you are comfortable.
```

Editing files with gedit

- Applications → Accessories → gedit
- File → New (Ctrl-n)
- File → Save (Ctrl-s)
- File → Open (Ctrl-o)
- Ctrl-x
- Ctrl-c
- Ctrl-v

The help for gedit

- gedit man page
[student@serverX ~]\$ yelp help:gedit



Lab

<lab1>

Redirect a long listing of all content in student's home directory, including hidden directories and files, into a file named **editing_final_lab.txt**. Your home directory files may not exactly match those shown in the example graphics. This lab edits arbitrary lines and columns. The important outcome is to practice the visual selection process.

<lab2>

Edit the file using Vim, to take advantage of visual mode.

<lab3>

Remove the first three lines, since those lines are not normal file names. Enter line-based visual mode with upper case **V**

<lab4>

Remove the permission columns for group and world on the first line. In this step, enter visual mode with lower case **v**, which allows selecting characters on a single line only.

Lab

<lab5>

Remove the permission columns for group and world on the remaining lines. This step will use a more efficient block selection visual mode to avoid having to repeat the single line edit multiple times. This time, enter visual mode with the control sequence **Ctrl-v**, which allows selecting a block of characters on multiple lines

<lab6>

Remove the group owner column, leaving only one "student" column on all lines. Use the same block selection technique as the last step.

<lab7>

Remove the time column, but leave the month and day on all lines. Again, use the block selection visual mode.

<lab8>

Remove the **Desktop** and **Public** rows. This time, enter visual mode with upper case **V**, which automatically selects full lines.

<lab9>

Save and exit. Make a backup, using the date (in seconds) to create a unique file name.

Lab

<lab10>

Mail the file contents as the message, not an attachment, to the user student

<lab11>

Append a dashed line to the file to recognize the beginning of newer content.

<lab12>

Append a full process listing, but only for processes owned by the current user **student** and running on the currently used terminal. View the process listing and send the listing to the file with one command line.

That's all