LINUX REFRESHER COURSE

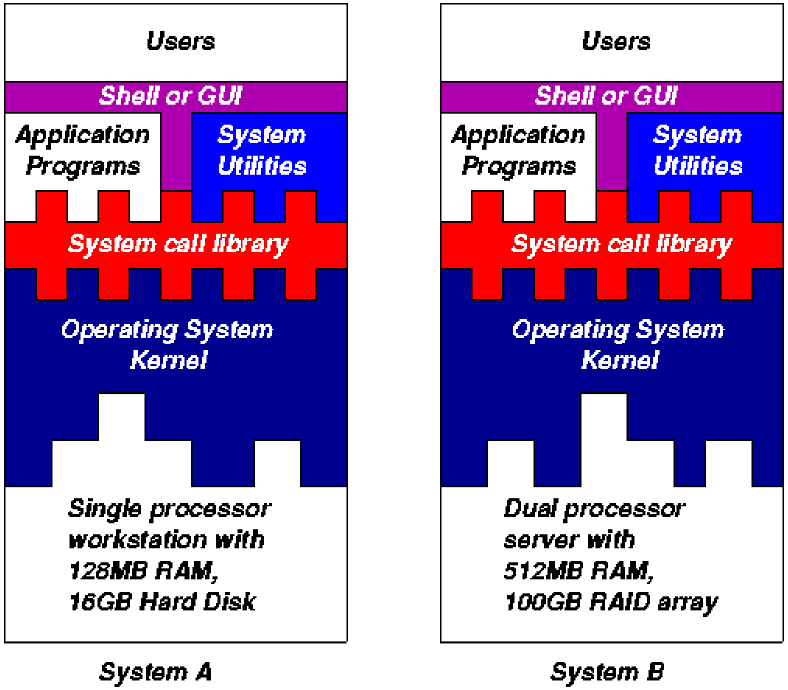
Muhammad Sadiq Sarfaraz
Institute of Scientific Computing
Technical University Braunschweig

Contents

- Operating System
- Linux Development
- Fundamental Components of Linux
- Linux File System
- File permission in Linux
- File manipulation commands
- Input Output Redirection
- Process Management Commmands
- Shell scripts
- Some Useful applications/open source software resources

Operating System

- An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers network cards etc.) in a **safe, efficient** and **abstract** way.



Operating System

- Kernel: The operating system kernel is in direct control of the underlying hardware

- System calls Library: Basic hardware-independent kernel services are exposed to higher-level programs through a library of system calls

- Application programs (e.g. word processors, spreadsheets) and system utility programs (simple but useful application programs that come with the operating system). They are executed using a shell or GUI

Linux Development

- Linux is a free open source UNIX OS for PCs that was originally developed in 1991 by Linus Torvalds

- The open source nature of Linux means

that the source code for the Linux kernel is freely available so that anyone can add features and correct deficiencies.

- several different development streams or distributions have emerged, e.g. Redhat, Mandrake, Debian, Ubuntu etc
- A distribution comprises a prepackaged kernel, system utilities, GUI interfaces and application programs.

SOME LINUX DISTRIBUTIONS

archlinux
debian
fedora
foresight linux
gentoo linux
Mandriva
linux Mint
kubuntu
openSUSE
PCLinuxOS
redhat.
sabayon
slackware
slax
ubuntu
xubuntu

Fundamental Components of Linux

- Linux has all of the components of a typical OS

1. Kernels

   The Linux kernel includes device driver support for a large number of PC hardware devices (graphics cards, network cards, hard disks etc.), advanced processor and memory management features

2. Shells and GUI

   Linux supports two forms of command input: through textual command line shells similar to those found on most Linux systems (e.g. sh - the Bourne shell, bash - the Bourne again shell and csh - the C shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers

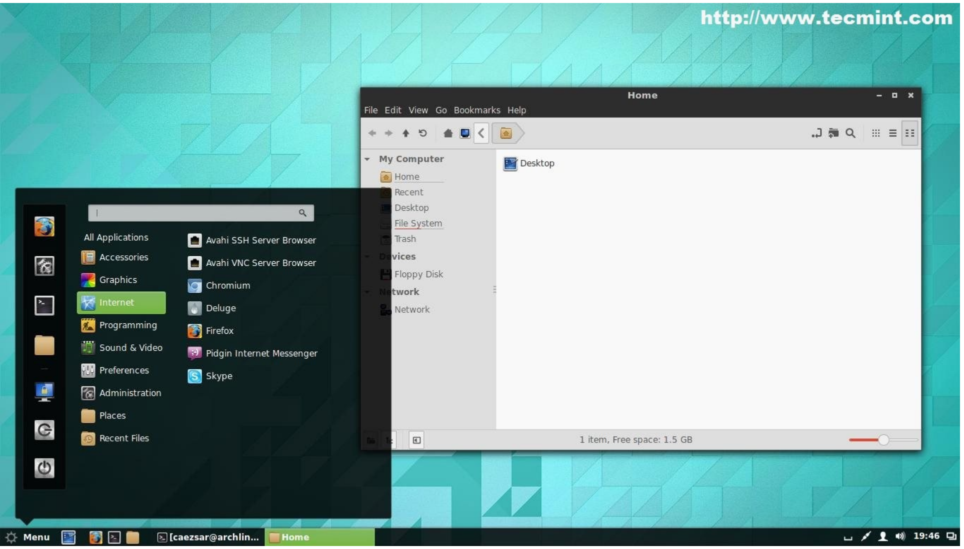
3. System Utilities

This includes commands such as ls, cp, grep, awk, sed, bc, wc, more, and so on. These system utilities are designed to be powerful tools that do a single task extremely well (e.g. grep finds text inside files while wc counts the number of words, lines and bytes inside a file

4. Application and Programs

several useful application programs as standard. Examples include the emacs editor, xv (an image viewer), gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language) and soffice

Interaction with Linux System

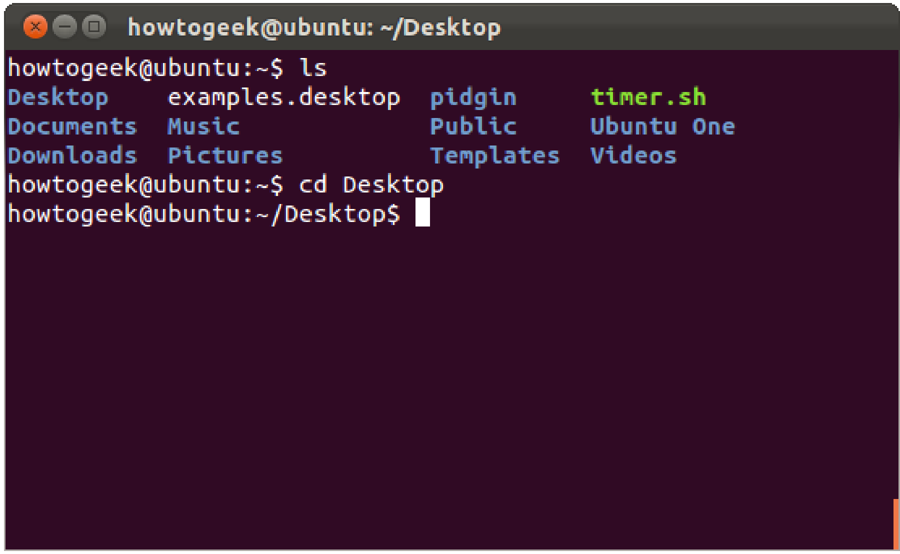


1 item, Free space: 1.5 GB

http://www.tecmint.com

Desktop Enviornment

Linux Terminal



```
howtogeek@ubuntu:~$ ls
Desktop    examples.desktop  pidgin     timer.sh
Documents  Music             Public     Ubuntu One
Downloads  Pictures          Templates  Videos
howtogeek@ubuntu:~$ cd Desktop
howtogeek@ubuntu:~/Desktop$ 
```

Linux File System

- The Linux operating system is built around the concept of a file system
- Every item stored in a file system belongs to one of four types:

1. **Ordinary files**

   Ordinary files can contain text, data, or program information.

2. **Directories**

   Directories are containers or folders that hold files, and other directories.

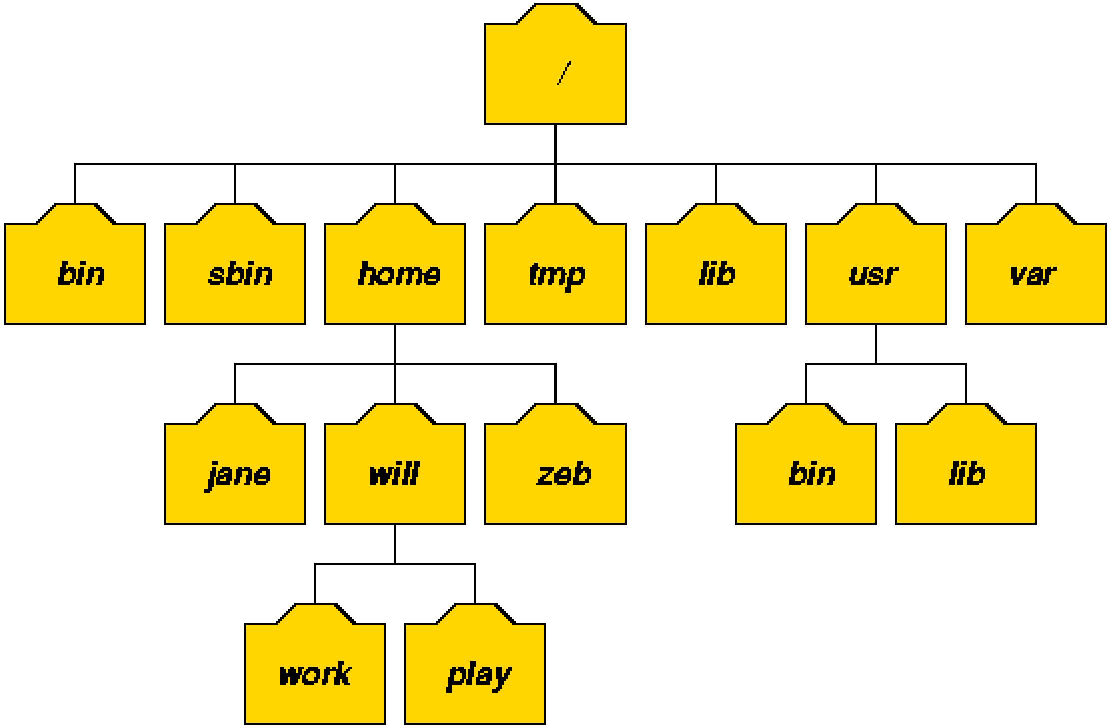
3. **Devices**

   To provide applications with easy access to hardware devices

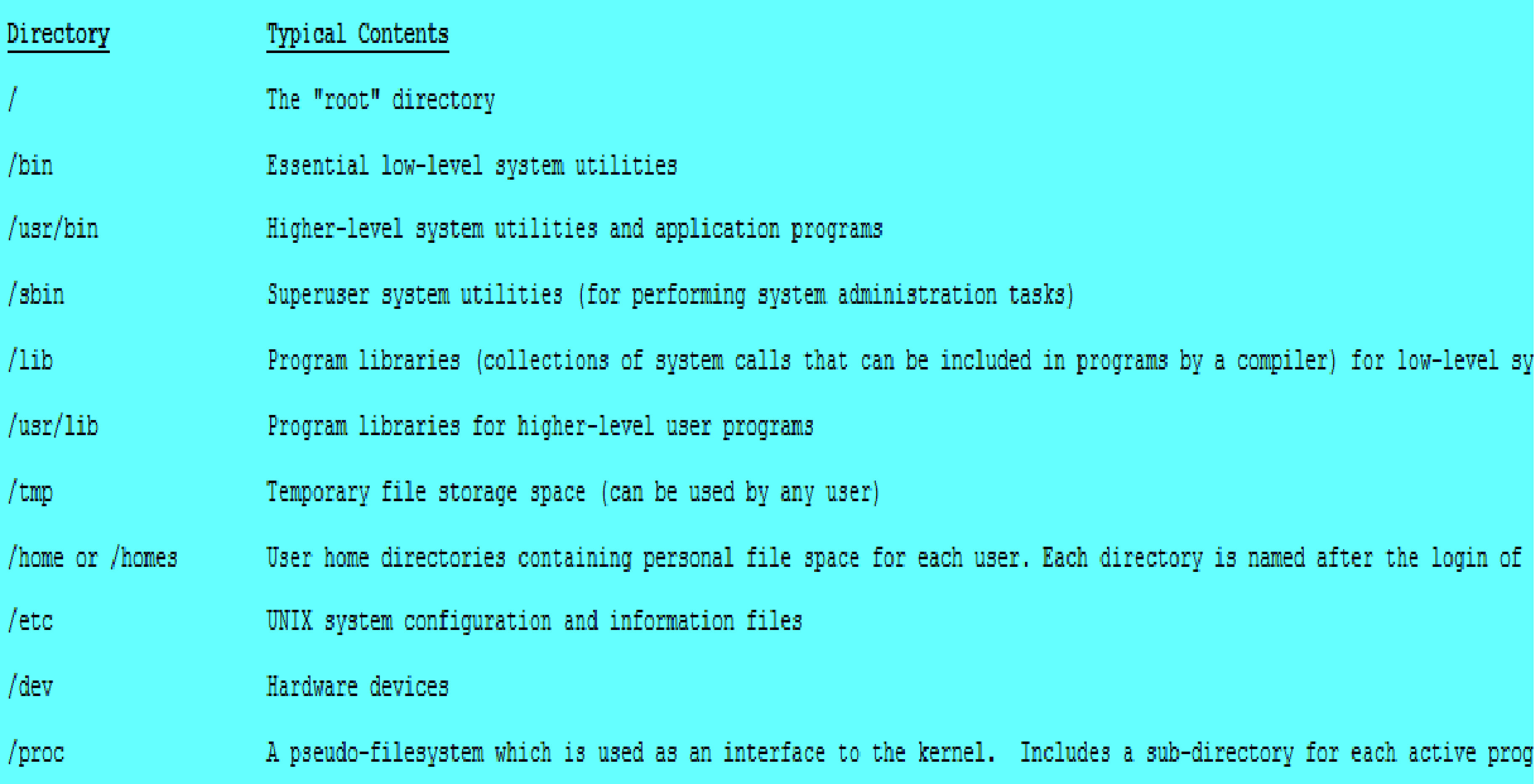
4. **Links**

   A link is a pointer to another file

Linux File System



Linux File System



Directory	Typical Contents
/	The "root" directory
/bin	Essential low-level system utilities
/usr/bin	Higher-level system utilities and application programs
/sbin	Superuser system utilities (for performing system administration tasks)
/lib	Program libraries (collections of system calls that can be included in programs by a compiler) for low-level sy
/usr/lib	Program libraries for higher-level user programs
/tmp	Temporary file storage space (can be used by any user)
/home or /homes	User home directories containing personal file space for each user. Each directory is named after the login of
/etc	UNIX system configuration and information files
/dev	Hardware devices
/proc	A pseudo-filesystem which is used as an interface to the kernel. Includes a sub-directory for each active prog

- **Some widely used directory and file handling commands**

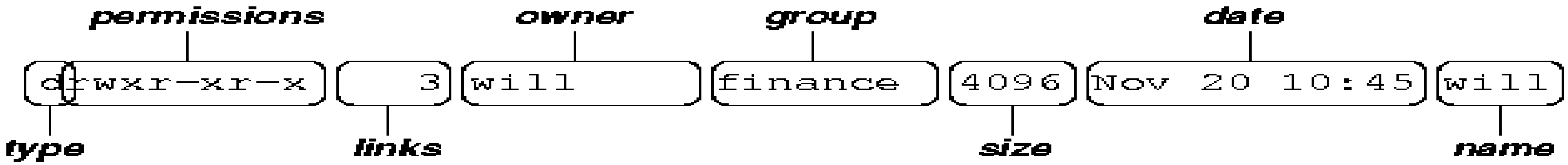
1. pwd:displays the full absolute path to the your current location in the filesystem.

2. ls: lists the contents of a directory.

3. others: cd, cp,mv,rm,cat ….

Linux File System

- File Details



```
drwxr-xr-x   3 will   finance 4096 Nov 20 10:45 will
```

- *type* is a single character specifying type of file
- *permissions* is a set of characters describing access rights.
- *links* refers to the number of filesystem links pointing to the file/directory
- *owner* is usually the user who created the file or directory.
- *group* denotes a collection of users who are allowed to access the file
- *size* is the length of a file.
- *date* is the date when the file or directory was last modified (wr
- *name* is the name of the file or directory.

File permission in Linux

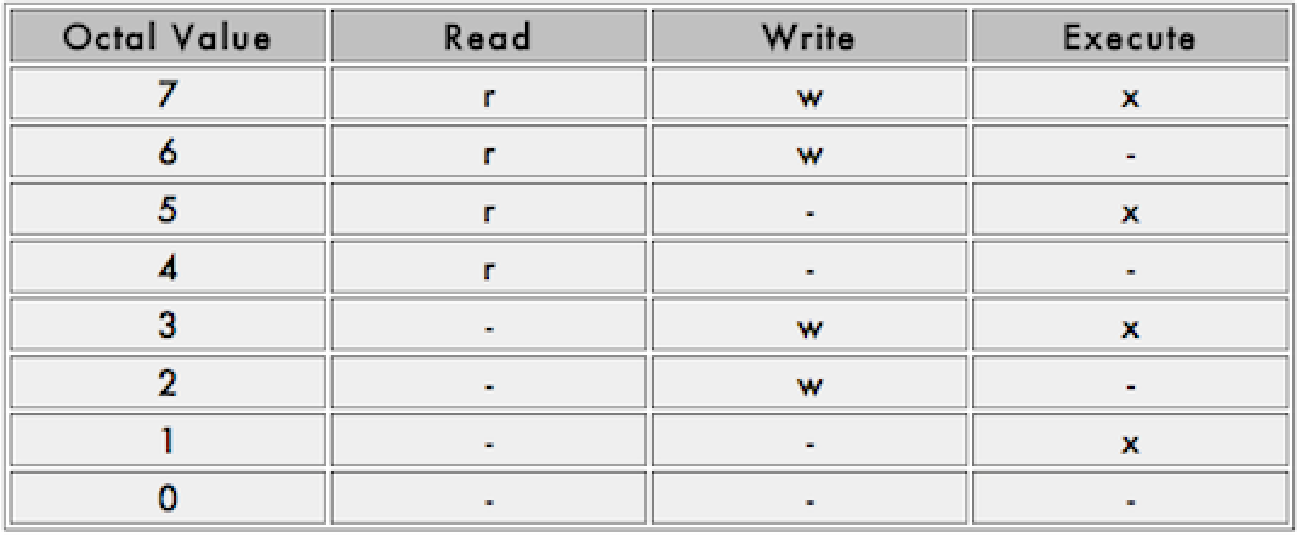
- three types of permissions, describing what operations can be performed on it.
  1. read (r),
  2. write (w)
  3. execute (x)
- three categories of users to perform these operations
  1. user/owner (u)
  2. group (g)
  3. others (o)

File permission in Linux

- File and directory permissions can only be modified by their owners using chmod (change mode) command

  chmod *options files*

- chmod accepts options in two forms. Firstly, permissions may be specified as a sequence of 3 octal digits (octal is like decimal except that the digit range is 0 to 7 instead of 0 to 9).



Octal Value	Read	Write	Execute
7	r	w	x
6	r	w	-
5	r	-	x
4	r	-	-
3	-	w	x
2	-	w	-
1	-	-	x
0	-	-	-

File permission in Linux

- For example the command:

```
$ chmod 600 private.txt
```

sets the permissions on private.txt to rw------- (i.e. only the owner can read and write to the file).

- Permissions may be specified symbolically, using the symbols u (user), g (group), o (other), a (all), r (read), w (write), x (execute), + (add permission), - (take away permission) and = (assign permission)

File Permission in Linux

- Symbolically setting file permission

```
chmod ug=rw,o+rw,a+x *.txt
```

sets the permissions on all files ending in *.txt to rw-rw---- (i.e. the owner and users in the file's group can read and write to the file, while the general public do not have any sort of access)

- chmod also supports a -R option which can be used to recursively modify file permissions, e.g.

```
$ chmod -R go+r play
```

will grant group and other read rights to the directory play and all of the files and directories within play.

Some file manipulation commands

- **file** *filename(s)* file analyzes a file's contents for you and reports a high-level description of what type of file it appears to be:

```
$ file myprog.c letter.txt webpage.html
myprog.c:      C program text
letter.txt:    English text
webpage.html:  HTML document text
```

- **head, tail** *filename* head and tail display the first and last few lines in a file respectively. You can specify the number of lines as an option, e.g.

```
 $ tail -20 messages.txt
$ head -5 messages.txt
```

Some file manipulation commands

- **find**, to search the path for a particular file

$ find *directory* -name *targetfile*

You can also execute commands on the files you find, e.g.

```
$ find . -name "*.txt" -exec wc -l '{}' ';'
```

counts the number of lines in every text file in and below the current directory.

- **grep** (General Regular Expression Print)

$ grep *options pattern files*

grep searches the named files (or standard input if no files are named) for lines that match a given pattern. The default behaviour of grep is to print out the matching lines.

Some file manipulation commands

- grep example

```
$ grep hello *.txt
```

searches all text files in the current directory for lines containing "hello“

- **tar** (tape archiver) tar backs up entire directories and files onto a tape device or (more commonly) into a single disk file known as an archive

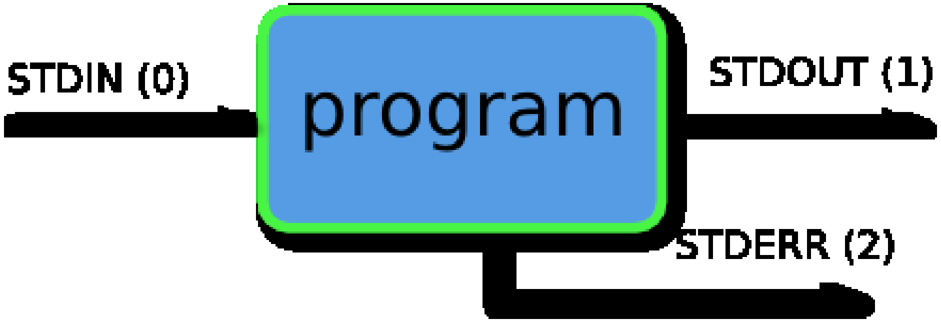
To create a disk file tar archive, use

```
  $ tar -cvf archivenamefilenames
   $ tar -tvf archivename  (To list the contents of a tar archive)
   $ tar -xvf archivename (To restore files from a tar archive)
```

Input Output Redirection

- Every program we run on the command line automatically has three data streams connected to it.

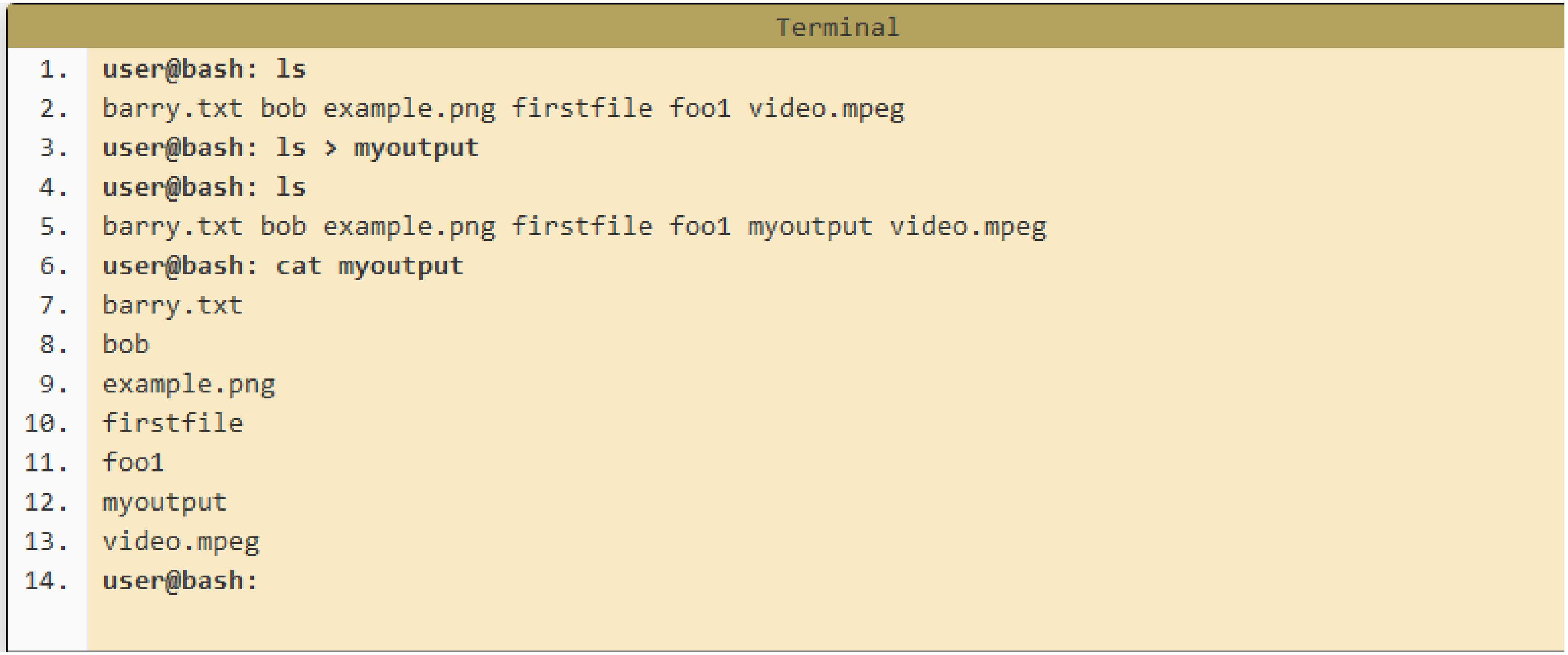
1. STDIN (0) - Standard input (data fed into the program)
2. STDOUT (1) - Standard output (data printed by the program, defaults to the terminal)
3. STDERR (2) - Standard error (for error messages, also defaults to the te



- redirection is the means by which one may connect these streams between programs and files to direct data in interesting and useful ways.

Input Output Redirection

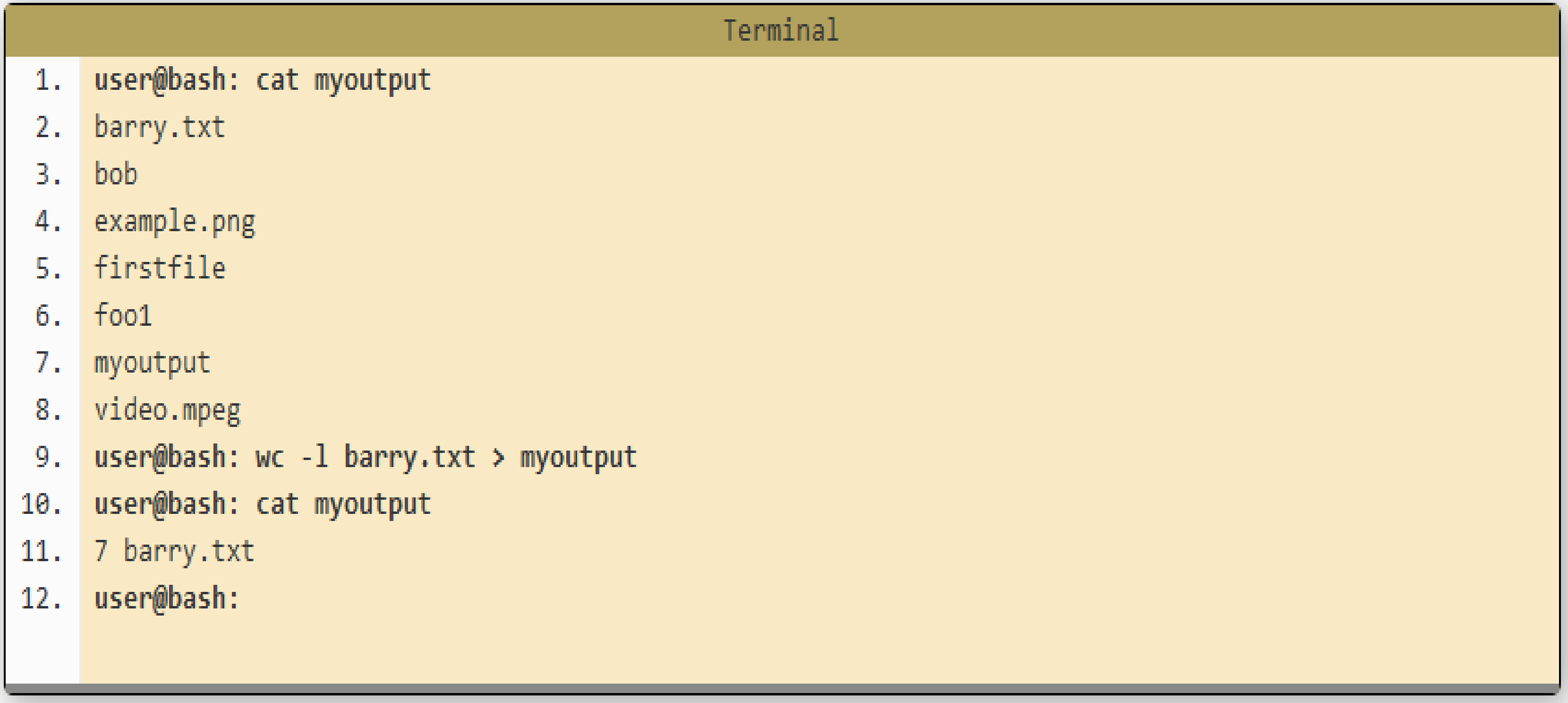
Output Redirection Example 1: Redirect output to file myoutput



```
user@bash: ls
barry.txt bob example.png firstfile foo1 video.mpeg
user@bash: ls > myoutput
user@bash: ls
barry.txt bob example.png firstfile foo1 myoutput video.mpeg
user@bash: cat myoutput
barry.txt
bob
example.png
firstfile
foo1
myoutput
video.mpeg
user@bash:
```

Input Output Redirection

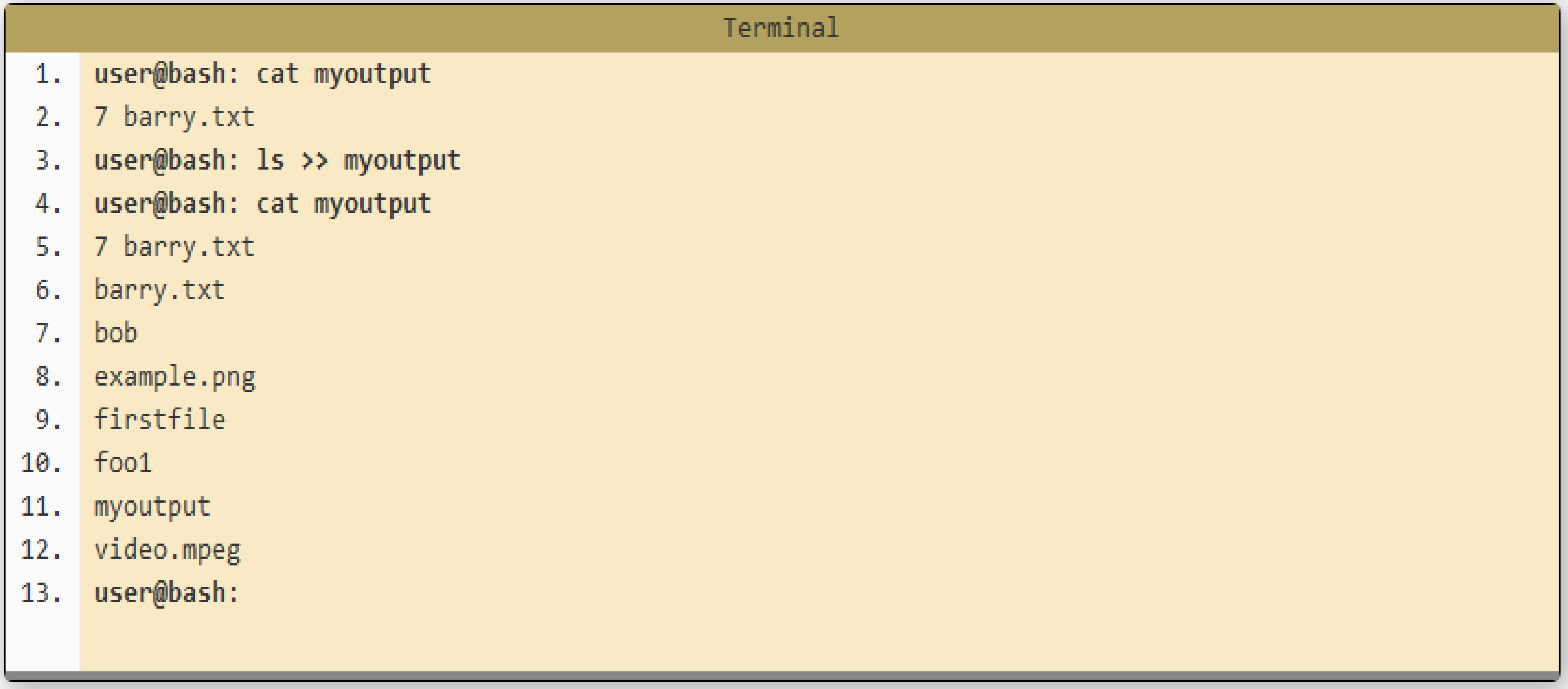
Output Redirection Example 2:overwrite myoutput file with the output of command wc -l barry.txt



```
user@bash: cat myoutput
barry.txt
bob
example.png
firstfile
foo1
myoutput
video.mpeg
user@bash: wc -l barry.txt > myoutput
user@bash: cat myoutput
7 barry.txt
user@bash:
```

Input Output Redirection

Output Redirection Example 3 : append myoutput file with output of command ls

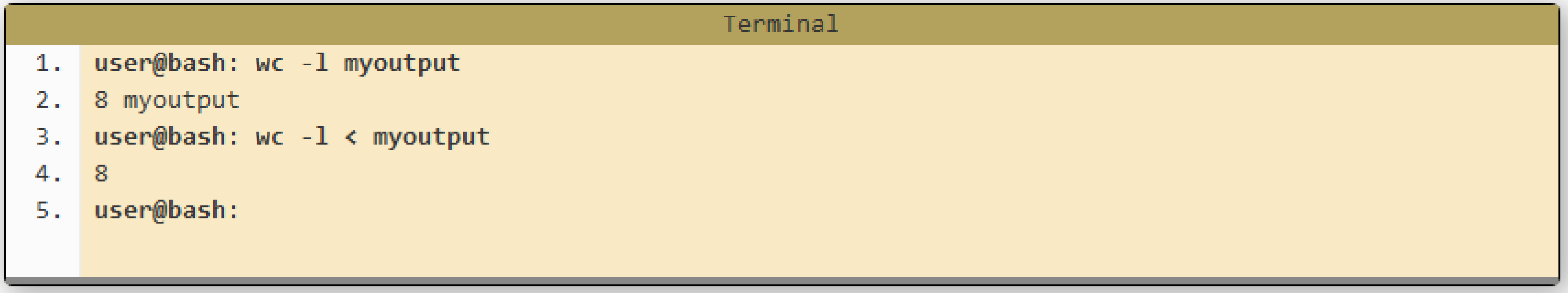


```
user@bash: cat myoutput
7 barry.txt
user@bash: ls >> myoutput
user@bash: cat myoutput
7 barry.txt
barry.txt
bob
example.png
firstfile
foo1
myoutput
video.mpeg
user@bash:
```

Input Output Redirection

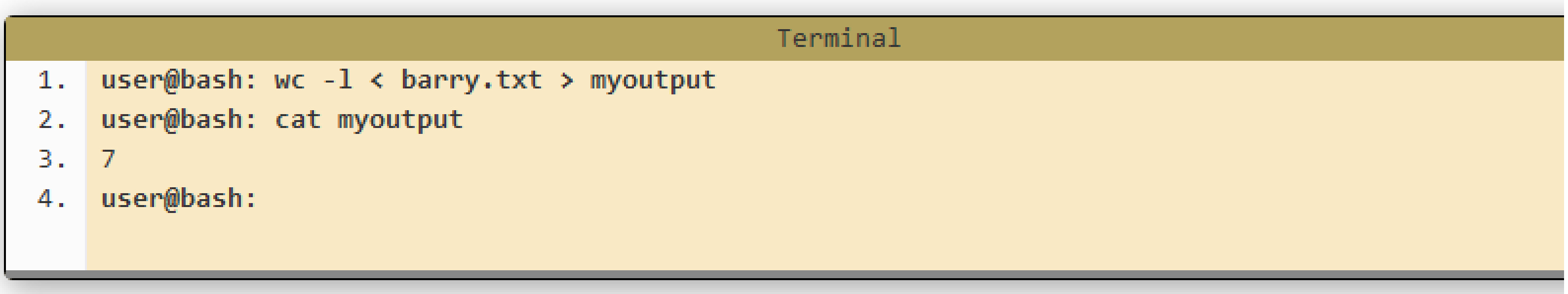
Input Redirection Example

via it's STDIN stream.



```
user@bash: wc -l myoutput
8 myoutput
user@bash: wc -l < myoutput
8
user@bash:
```

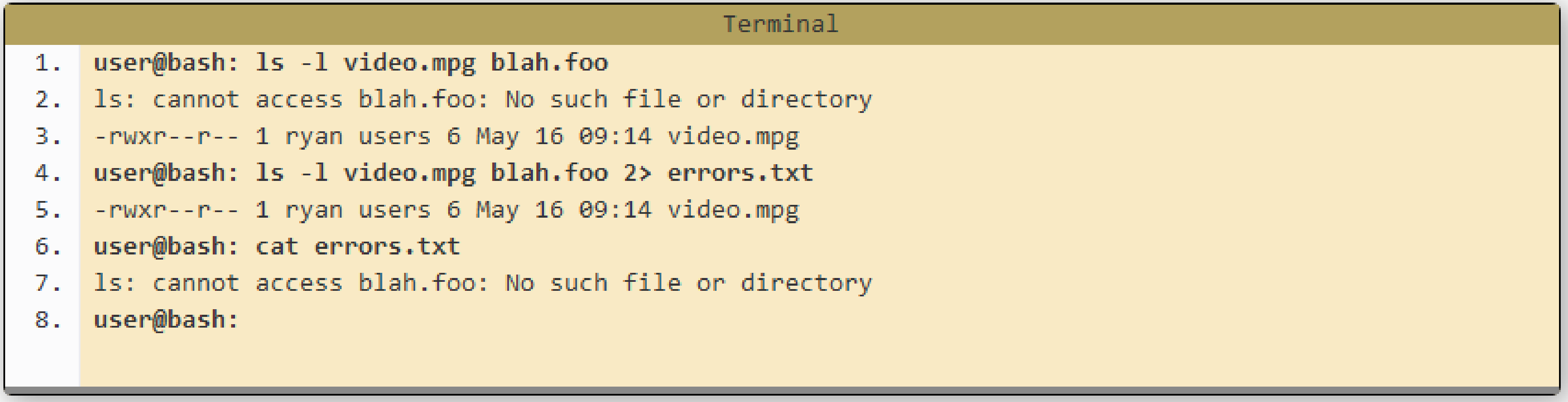
Input Output Redirection combined Example



```
user@bash: wc -l < barry.txt > myoutput
user@bash: cat myoutput
7
user@bash:
```

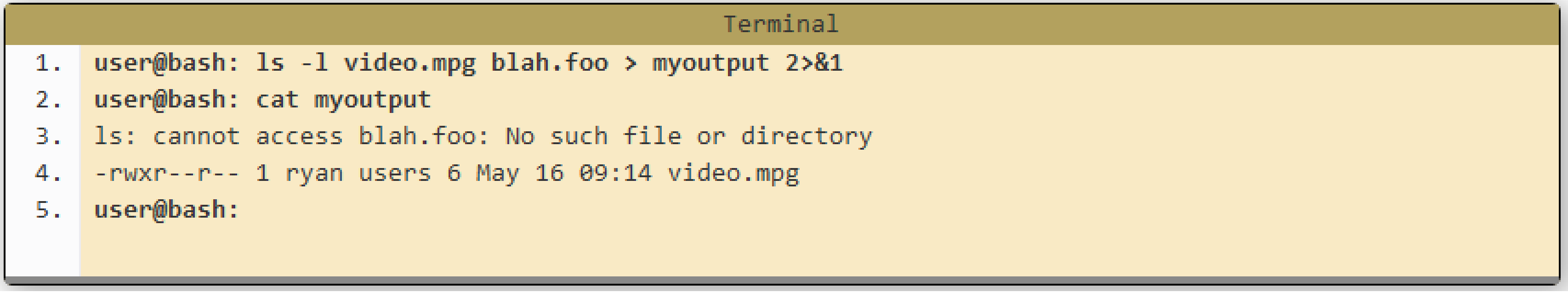
STDERR Redirection

Save error message from terminal in error.txt file



```
user@bash: ls -l video.mpg blah.foo
ls: cannot access blah.foo: No such file or directory
-rwxr--r-- 1 ryan users 6 May 16 09:14 video.mpg
user@bash: ls -l video.mpg blah.foo 2> errors.txt
-rwxr--r-- 1 ryan users 6 May 16 09:14 video.mpg
user@bash: cat errors.txt
ls: cannot access blah.foo: No such file or directory
user@bash:
```

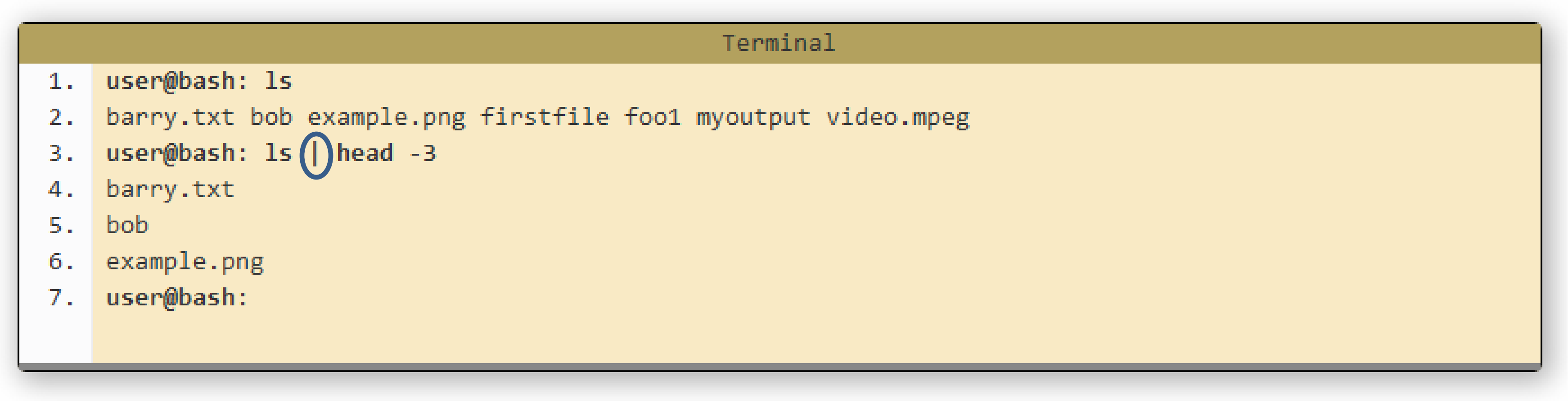
Save both output of ls and error message in myoutput



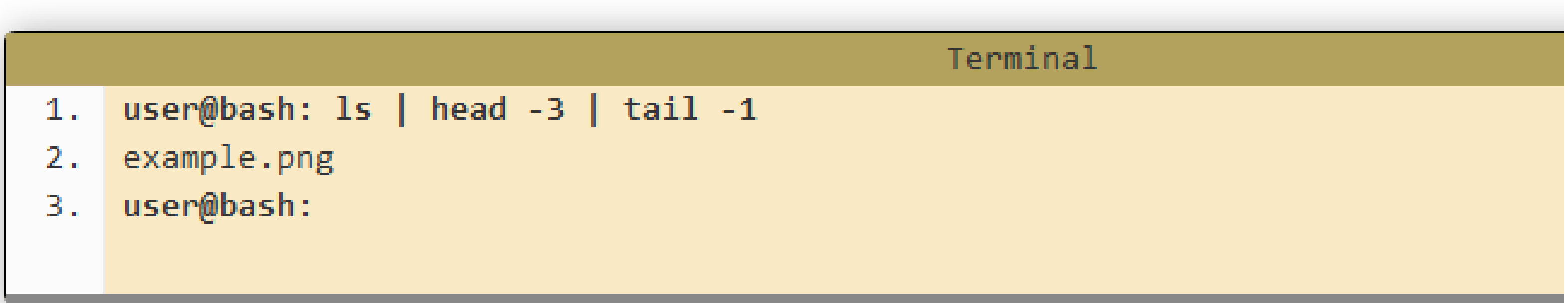
```
user@bash: ls -l video.mpg blah.foo > myoutput 2>&1
user@bash: cat myoutput
ls: cannot access blah.foo: No such file or directory
-rwxr--r-- 1 ryan users 6 May 16 09:14 video.mpg
user@bash:
```

Piping

- Piping provides the mechanism for sending data from one program to another. Operator used for this purpose is (|)



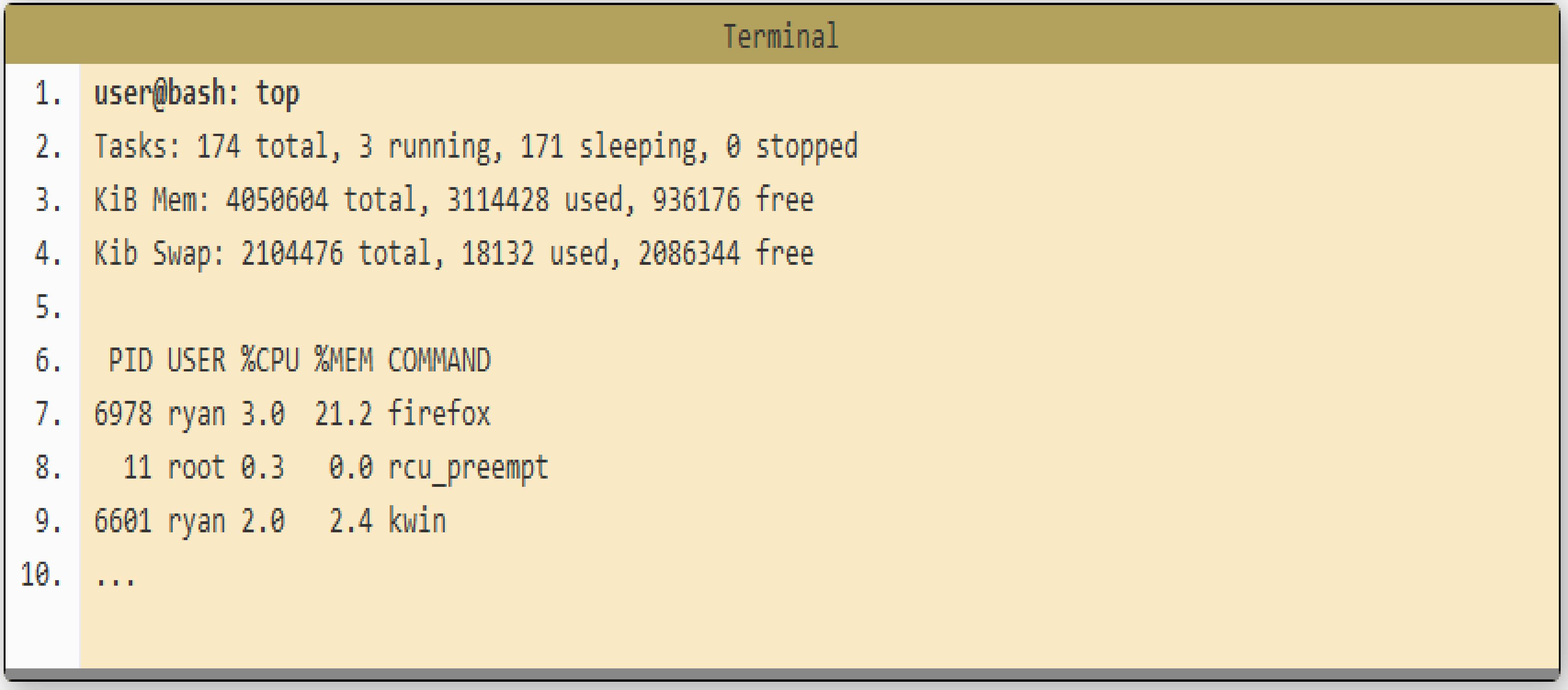
```
user@bash: ls
barry.txt bob example.png firstfile foo1 myoutput video.mpeg
user@bash: ls | head -3
barry.txt
bob
example.png
user@bash:
```



```
user@bash: ls | head -3 | tail -1
example.png
user@bash:
```

Process Management Commands

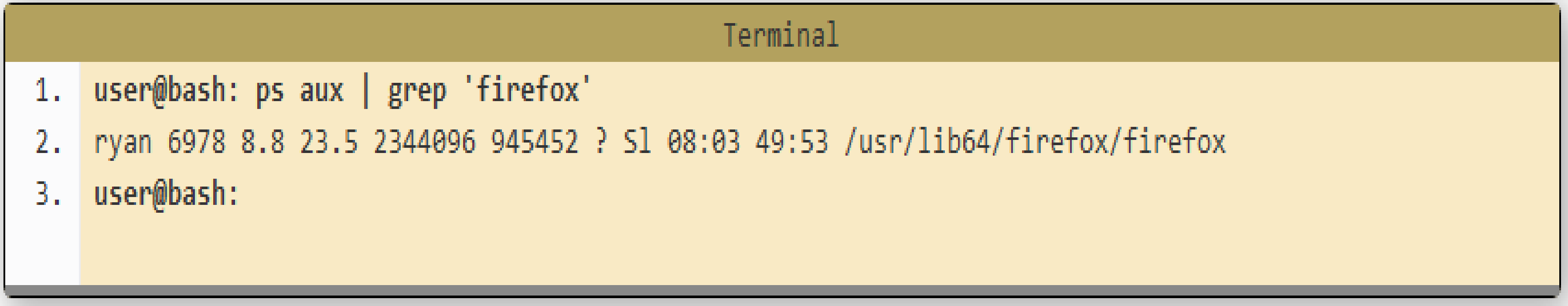
- **top** View real-time data about processes running on the system.



```
user@bash: top
Tasks: 174 total, 3 running, 171 sleeping, 0 stopped
KiB Mem: 4050604 total, 3114428 used, 936176 free
Kib Swap: 2104476 total, 18132 used, 2086344 free

 PID USER %CPU %MEM COMMAND
6978 ryan 3.0  21.2 firefox
  11 root 0.3   0.0 rcu_preempt
6601 ryan 2.0   2.4 kwin
...
```

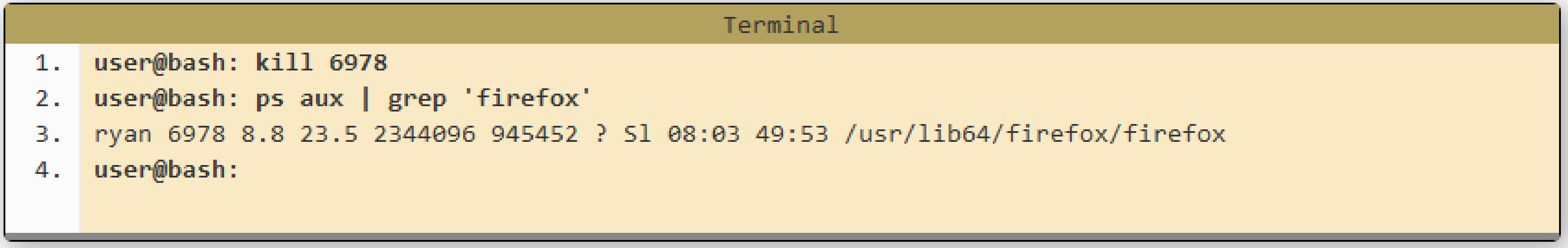
Process Management Commands

- **ps:** shows the processes running on the current terminal, to get a complete list add **aux (auxilary)**

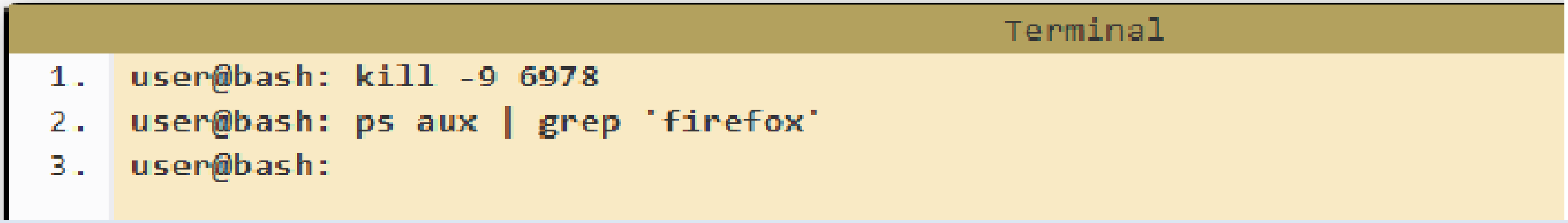


```
user@bash: ps aux | grep 'firefox'
ryan 6978 8.8 23.5 2344096 945452 ? Sl 08:03 49:53 /usr/lib64/firefox/firefox
user@bash:
```

- **kill :** to terminate a process



```
user@bash: kill 6978
user@bash: ps aux | grep 'firefox'
ryan 6978 8.8 23.5 2344096 945452 ? Sl 08:03 49:53 /usr/lib64/firefox/firefox
user@bash:
```



```
user@bash: kill -9 6978
user@bash: ps aux | grep 'firefox'
user@bash:
```

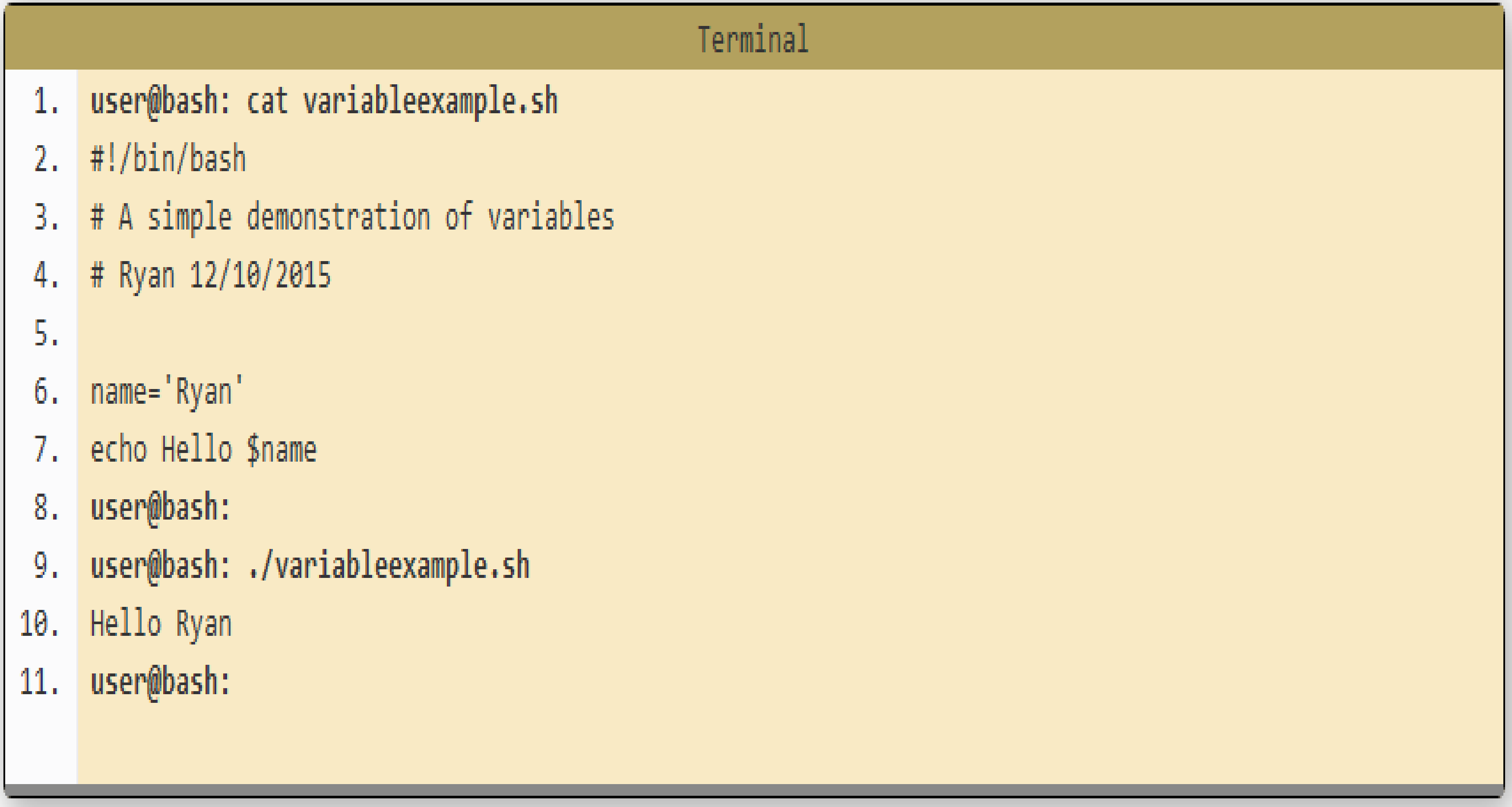
Shell scripts

- A shell is a program which reads and executes commands for the user.

- Shells also usually provide features such job control, input and output redirection and a command language for writing *shell scripts*.

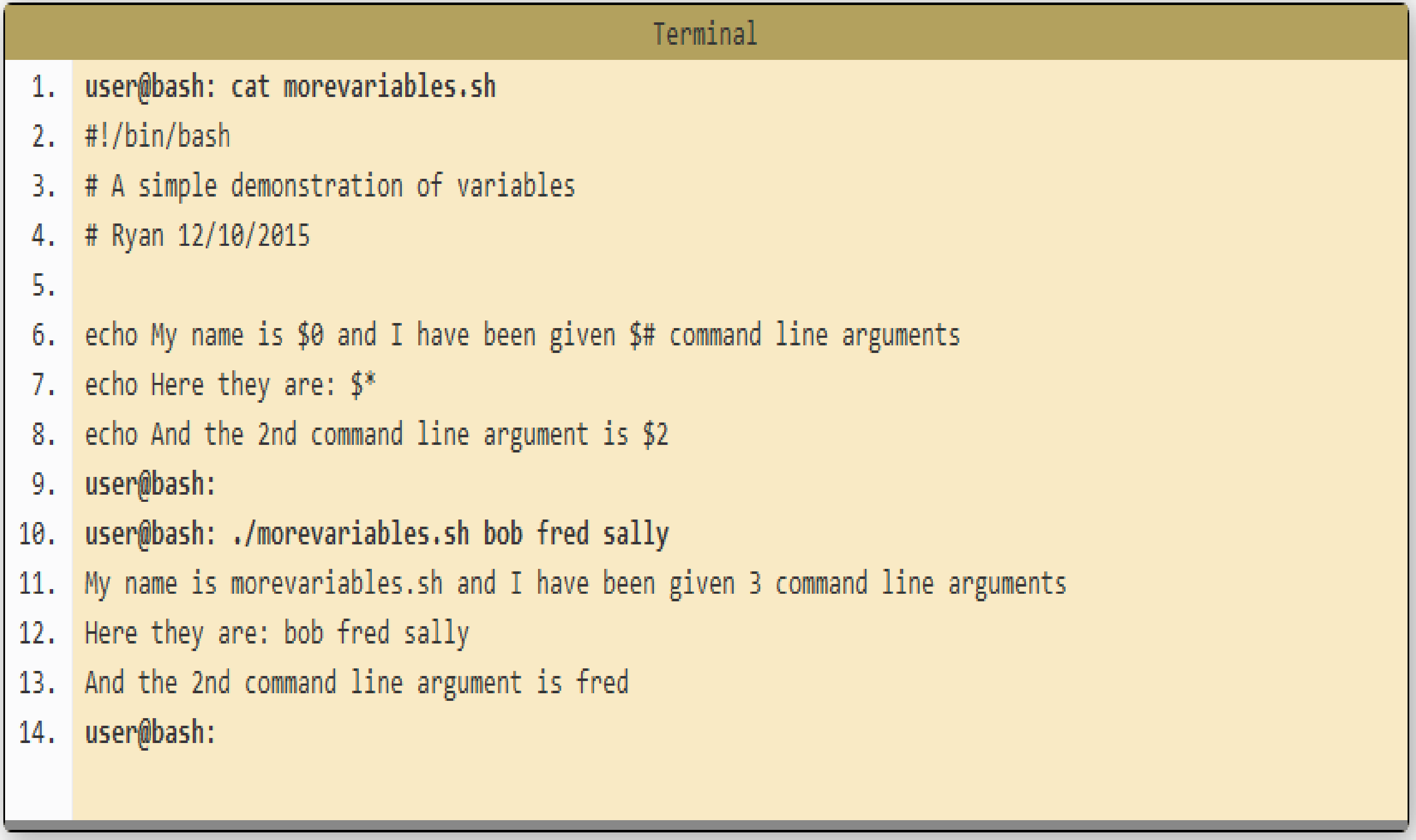
- Different shells available on Linux(e.g. sh, bash, csh, ksh, tcsh etc.), and they each support a different command language.

Shell scripts Examples



```
user@bash: cat variableexample.sh
#!/bin/bash
# A simple demonstration of variables
# Ryan 12/10/2015

name='Ryan'
echo Hello $name
user@bash:
user@bash: ./variableexample.sh
Hello Ryan
user@bash:
```

Shell scripts Examples



```
user@bash: cat morevariables.sh
#!/bin/bash
# A simple demonstration of variables
# Ryan 12/10/2015

echo My name is $0 and I have been given $# command line arguments
echo Here they are: $*
echo And the 2nd command line argument is $2
user@bash:
user@bash: ./morevariables.sh bob fred sally
My name is morevariables.sh and I have been given 3 command line arguments
Here they are: bob fred sally
And the 2nd command line argument is fred
user@bash:
```

Shell scripts Examples

- **$0** - The name of the script.

- **$1 - $9** - Any command line arguments given to the script. $1 is the first argument, $2 the second and so on.

- **$#** - How many command line arguments were given to the script.

- **$*** - All of the command line arguments.

Some Useful Applications:Text editor vi

- vi ("vee-eye", short for visual, or perhaps vile) is a display-oriented text editor based on an underlying line editor called ex.

- It also uses standard alphanumeric keys for commands, so it can be used on almost any terminal or workstation without having to worry about unusual keyboard mappings.

- To start vi, enter:

  $ vi *filename*

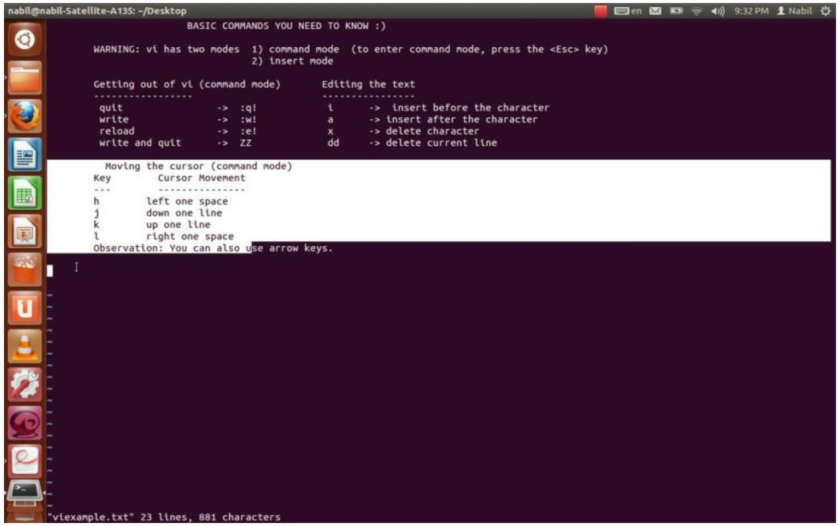
where *filename* is the name of the file you want to edit.

Some Useful Applications:Text editor vi

- mode-based operation:vi has two modes: command mode and input mode.

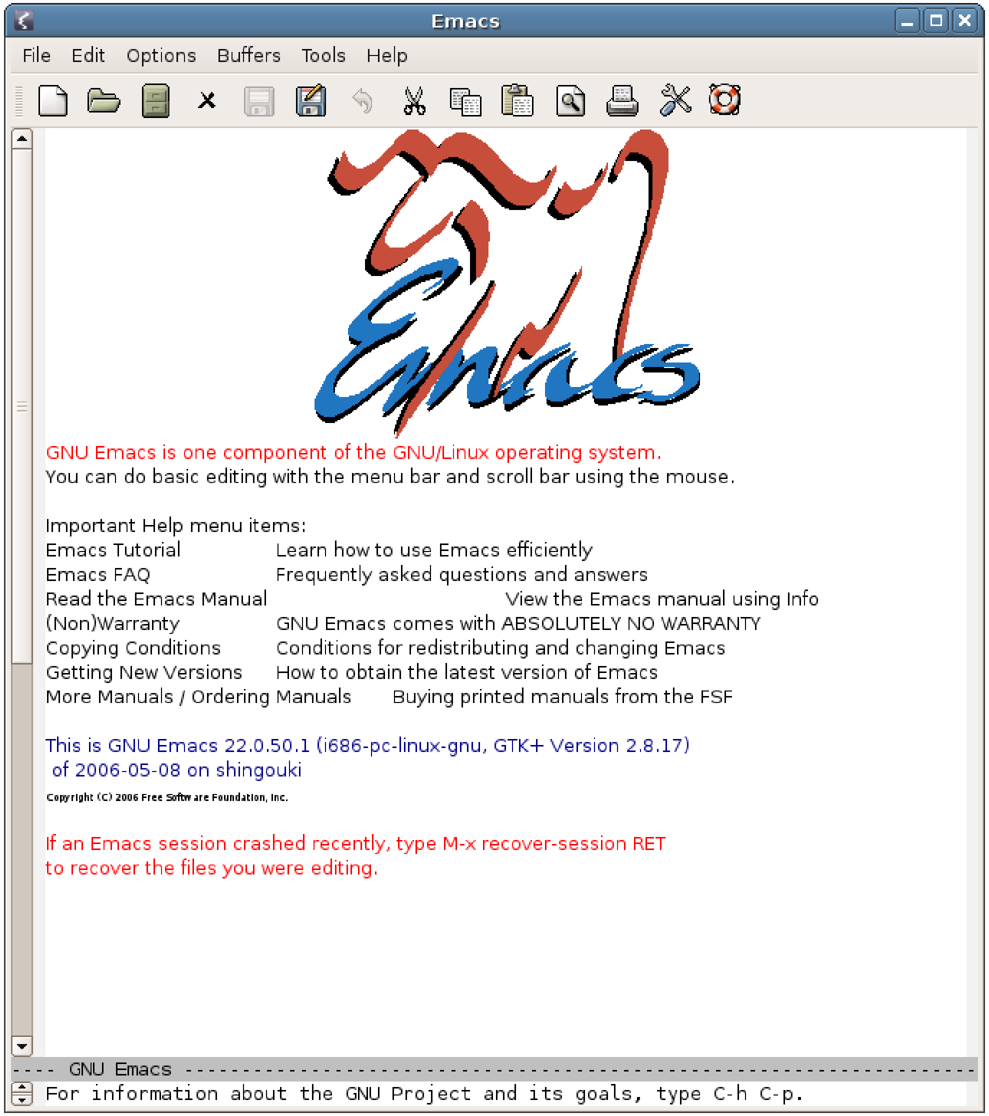
- In command mode, characters you type perform actions (e.g. moving the cursor, cutting or copying text, etc.)

- In input mode, characters you type are inserted or overwrite existing text.



```
                    BASIC COMMANDS YOU NEED TO KNOW :)

        WARNING: vi has two modes  1) command mode  (to enter command mode, press the <Esc> key)
                                   2) insert mode

        Getting out of vi (command mode)       Editing the text
        -----------------                      ----------------
         quit                   ->  :q!         i      ->  insert before the character
         write                  ->  :w!         a      -> insert after the character
         reload                 ->  :e!         x      -> delete character
         write and quit         ->  ZZ          dd     -> delete current line

          Moving the cursor (command mode)
        Key        Cursor Movement
        ---        ---------------
        h        left one space
        j        down one line
        k        up one line
        l        right one space
        Observation: You can also use arrow keys.

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"viexample.txt" 23 lines, 881 characters
```

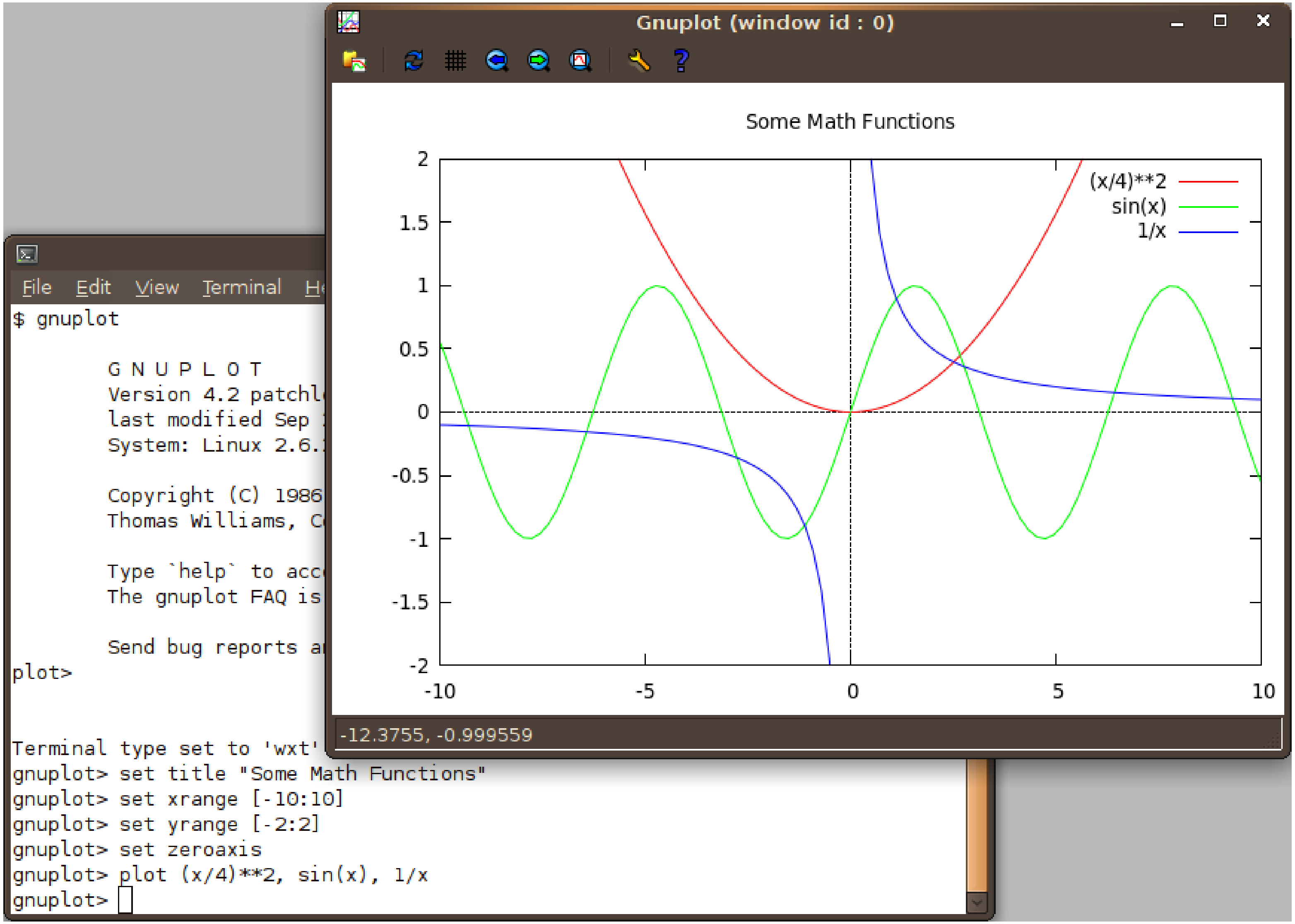
Some Useful Applications: emacs



```
GNU Emacs is one component of the GNU/Linux operating system.
You can do basic editing with the menu bar and scroll bar using the mouse.

Important Help menu items:
Emacs Tutorial          Learn how to use Emacs efficiently
Emacs FAQ               Frequently asked questions and answers
Read the Emacs Manual                          View the Emacs manual using Info
(Non)Warranty           GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions      Conditions for redistributing and changing Emacs
Getting New Versions    How to obtain the latest version of Emacs
More Manuals / Ordering Manuals       Buying printed manuals from the FSF

This is GNU Emacs 22.0.50.1 (i686-pc-linux-gnu, GTK+ Version 2.8.17)
 of 2006-05-08 on shingouki
Copyright (C) 2006 Free Software Foundation, Inc.

If an Emacs session crashed recently, type M-x recover-session RET
to recover the files you were editing.
```

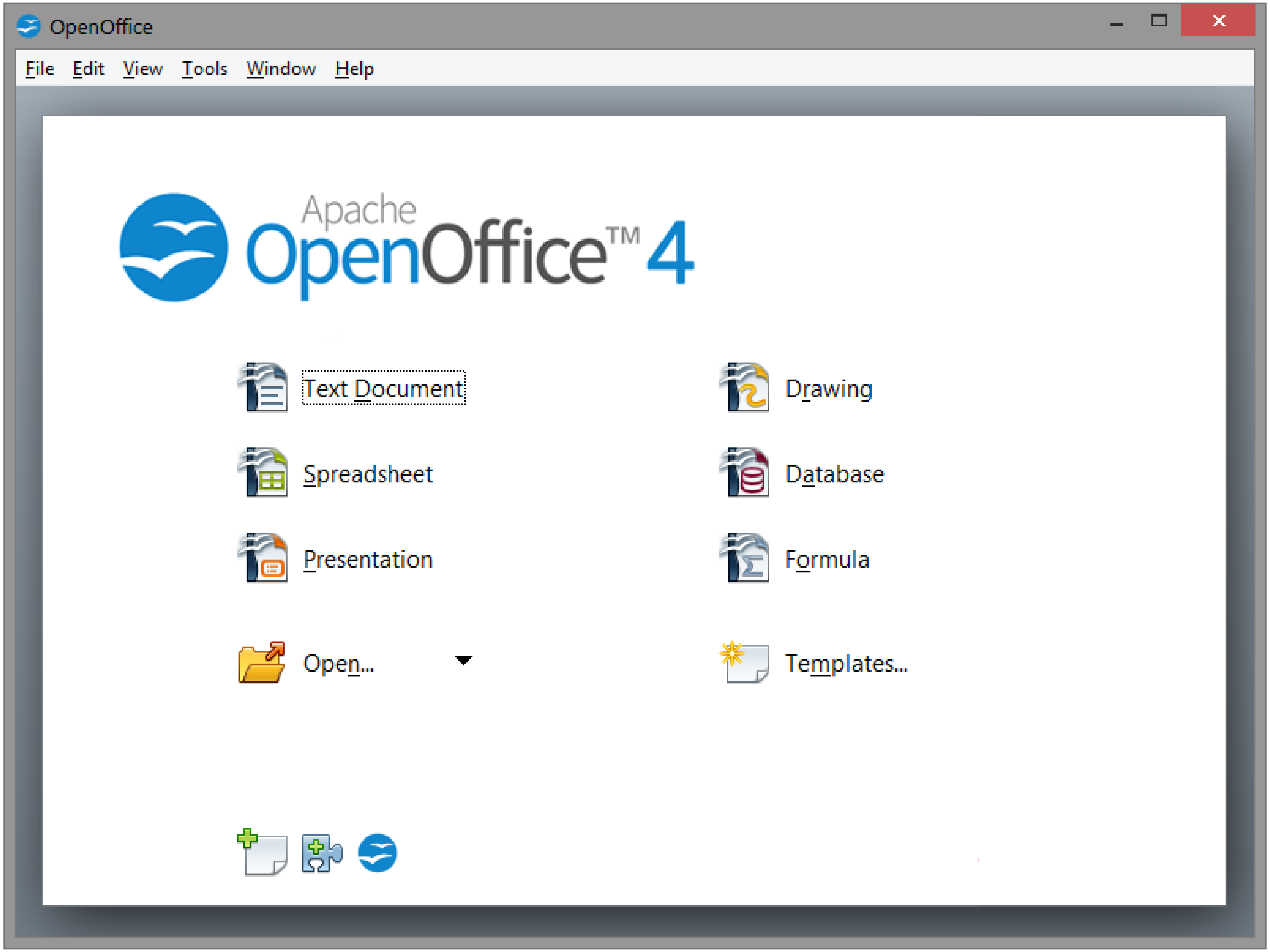
```
---- GNU Emacs ------------------------------------------------------------
For information about the GNU Project and its goals, type C-h C-p.
```

Some Useful applications: Gnuplot

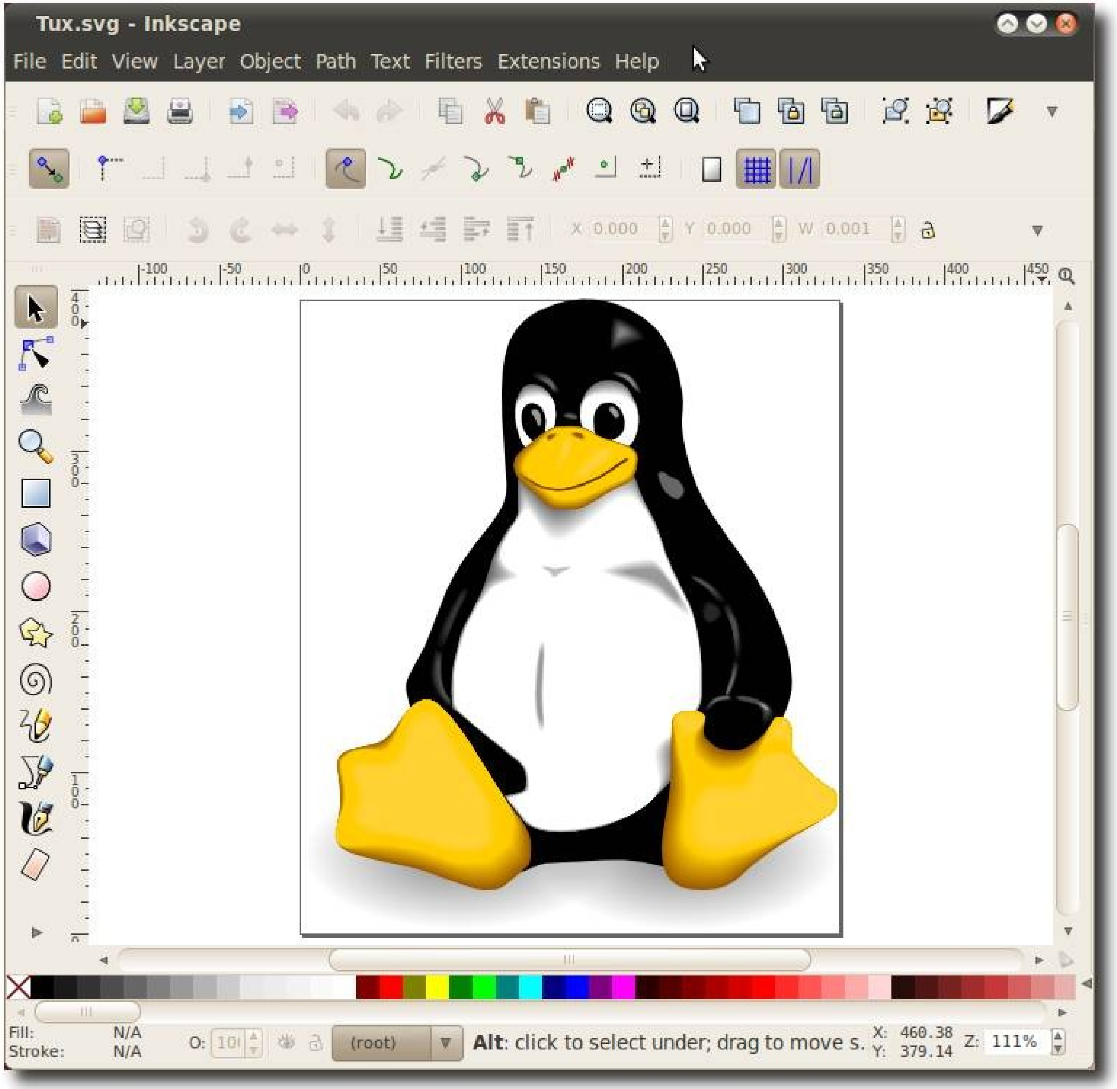


```
$ gnuplot

        G N U P L O T
        Version 4.2 patchl
        last modified Sep 
        System: Linux 2.6.

        Copyright (C) 1986
        Thomas Williams, C

        Type `help` to acc
        The gnuplot FAQ is

        Send bug reports a
plot>


Terminal type set to 'wxt'
gnuplot> set title "Some Math Functions"
gnuplot> set xrange [-10:10]
gnuplot> set yrange [-2:2]
gnuplot> set zeroaxis
gnuplot> plot (x/4)**2, sin(x), 1/x
gnuplot> 
```

Some Useful applications: OpenOffice



Some Useful applications: Inkscape



Alt: click to select under; drag to move s.