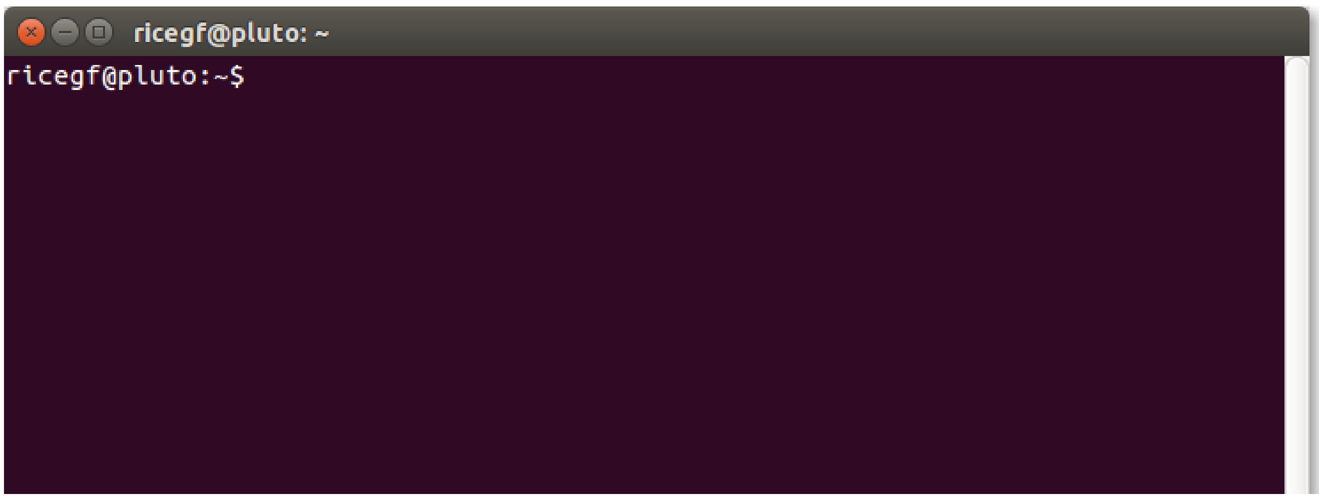
Using bash in 5 Pages

(the “Bourne-Again SHell”)

1 Starting bash

- In Ubuntu, press **Ctrl-Alt-t**. Or press the “super” key ( ) and type “terminal”. Or double-click LXTerminal on the desktop, or select Start → System Tools → LXTerminal.
- In Mac OS X, select Applications → Utilities → Terminal.
- In Windows, you have 2 options – run bash under Windows or run bash under Linux. Or both!
    - Install Cygwin (recommended – works on all Windows versions) or MinGW’s MSYS. Both work with native Windows applications, and allow you to script Windows activities.
    - Install the Windows Subsystem for Linux (Windows 10 Anniversary Edition or later), which actually runs all of Ubuntu under Windows similar to a virtual machine. Bash here runs only Ubuntu applications, and cannot script Windows activities, but it’s compatible with our class development environment for the first portion of the semester.



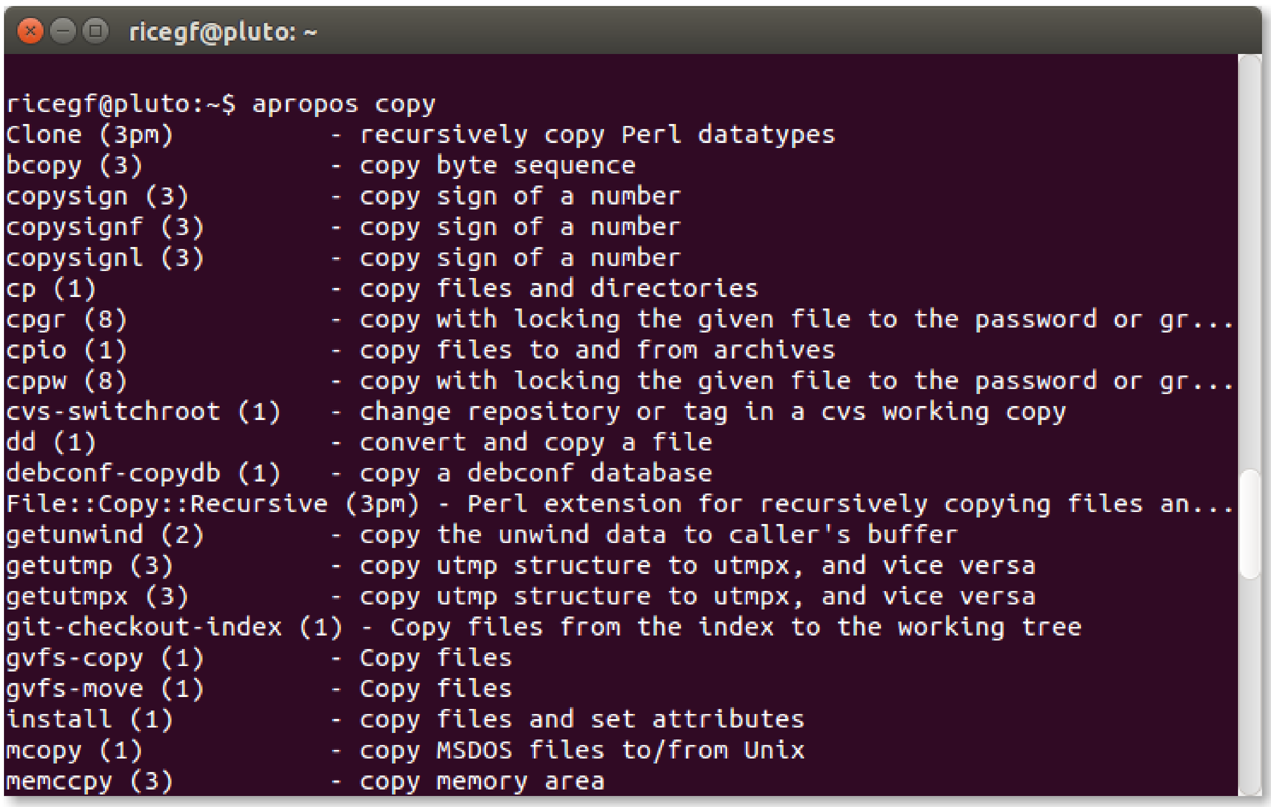
```
ricegf@pluto:~$
```

Ubuntu Terminal / Bash Tips and Tricks

- Use **View** → **Zoom In** to make text bigger, **View** → **Zoom Out** to make text smaller.
- The **up-arrow** key will step through previous commands, which may be edited and re-entered.
- The **mouse scroll wheel** and the **scroll bar** on the right review previous work.
- Select text, then **right click** → **Copy** to copy text (such as earlier command output) to the clipboard. **Right click** → **Paste** pastes the text from the clipboard onto the command line for editing and submission.
- In Linux, you can also select text to copy it to the special “X buffer”, and middle-click to any window to paste the text there. This is faster than the usual method, but only works for text.
- Type part of a command or filename, and press **Tab** to complete it.
- **Control-Z** stops the current command. Use **fg** to continue it, or **bg** to run it in the background. Or add a **&** to the end of a command to run it in the background from the start. Or select File → Open Terminal to just open a new terminal.

2 Getting help in bash

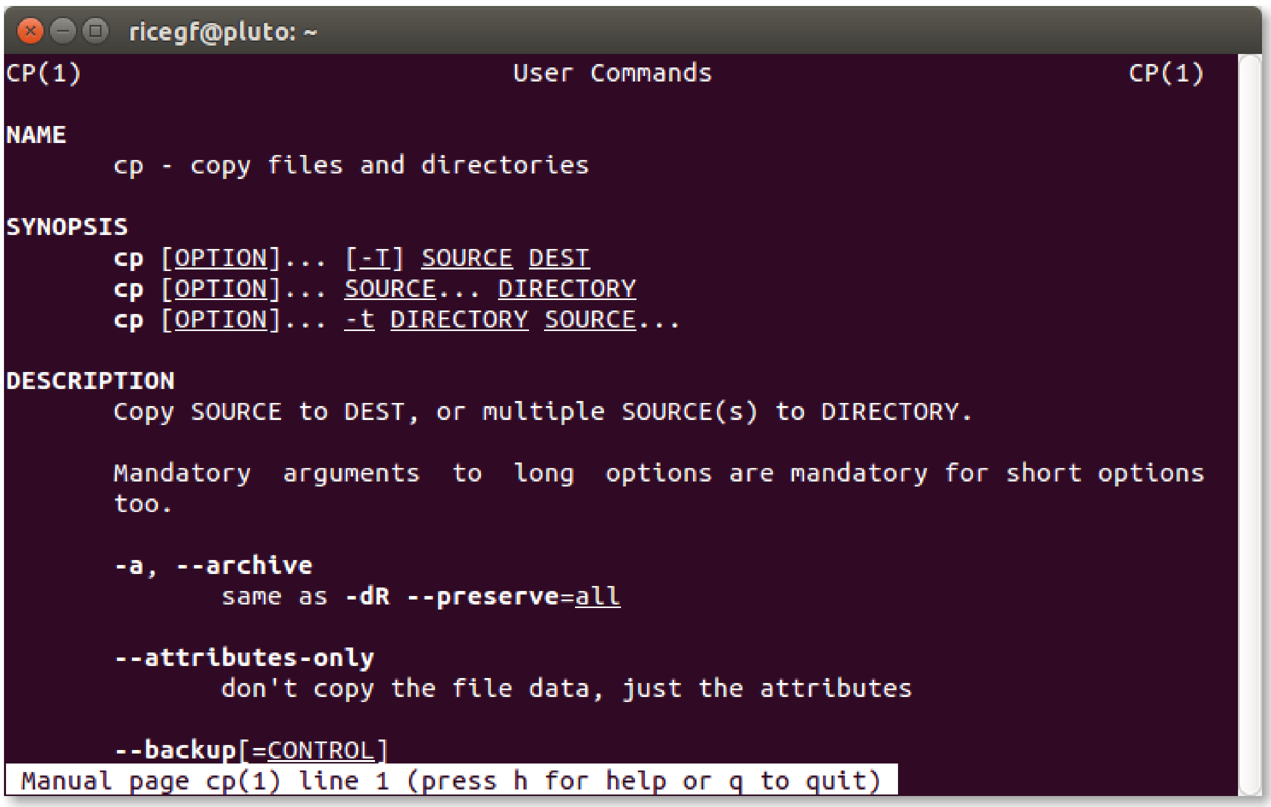
1. **An alphabetized list of bash commands is available on-line at http://ss64.com/bash/.**
2. **apropos [topic]** lists all commands related to the specified topic.



ricegf@pluto: ~

```
ricegf@pluto:~$ apropos copy
Clone (3pm)            - recursively copy Perl datatypes
bcopy (3)              - copy byte sequence
copysign (3)           - copy sign of a number
copysignf (3)          - copy sign of a number
copysignl (3)          - copy sign of a number
cp (1)                 - copy files and directories
cpgr (8)               - copy with locking the given file to the password or gr...
cpio (1)               - copy files to and from archives
cppw (8)               - copy with locking the given file to the password or gr...
cvs-switchroot (1)     - change repository or tag in a cvs working copy
dd (1)                 - convert and copy a file
debconf-copydb (1)     - copy a debconf database
File::Copy::Recursive (3pm) - Perl extension for recursively copying files an...
getunwind (2)          - copy the unwind data to caller's buffer
getutmp (3)            - copy utmp structure to utmpx, and vice versa
getutmpx (3)           - copy utmp structure to utmpx, and vice versa
git-checkout-index (1) - Copy files from the index to the working tree
gvfs-copy (1)          - Copy files
gvfs-move (1)          - Copy files
install (1)            - copy files and set attributes
mcopy (1)              - copy MSDOS files to/from Unix
memccpy (3)            - copy memory area
```

3. **man [command]** displays a concise, interactive manual of any command. Try **man man**.



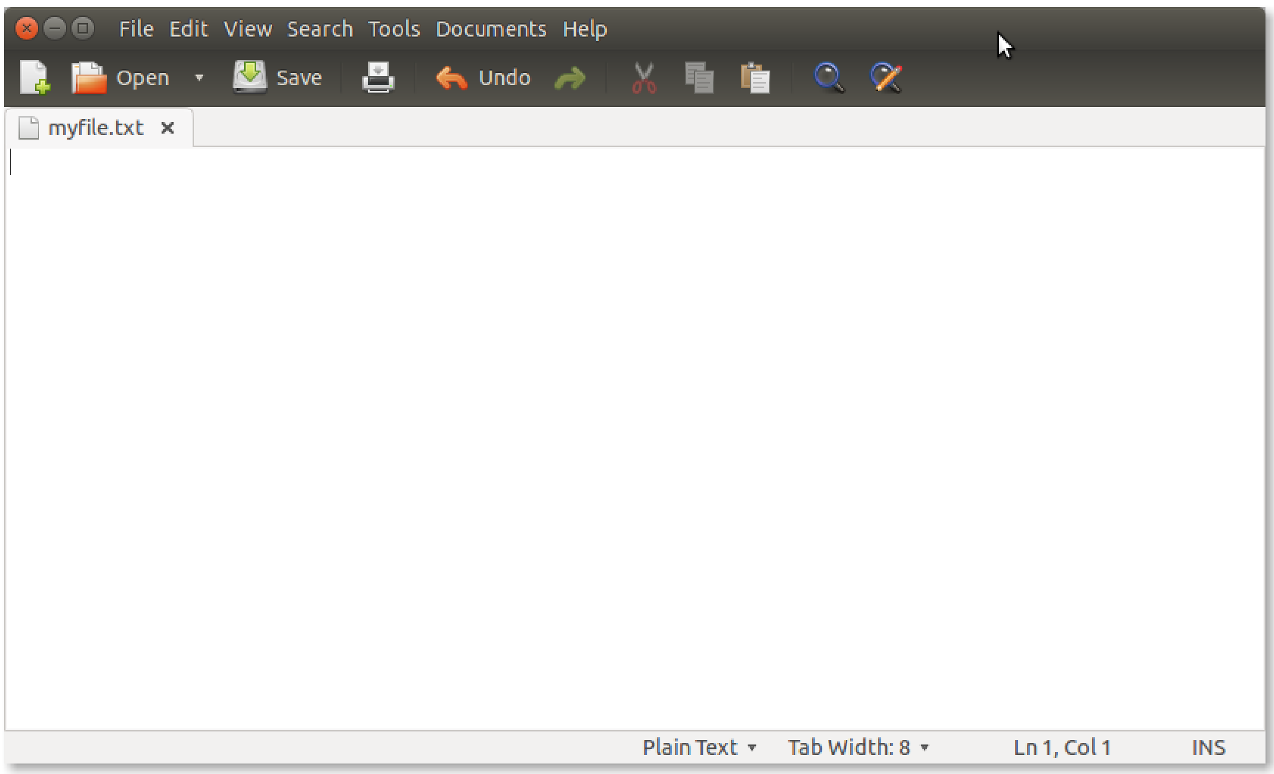
ricegf@pluto: ~

```
CP(1)                            User Commands                            CP(1)

NAME
       cp - copy files and directories

SYNOPSIS
       cp [OPTION]... [-T] SOURCE DEST
       cp [OPTION]... SOURCE... DIRECTORY
       cp [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
       Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --archive
              same as -dR --preserve=all

       --attributes-only
              don't copy the file data, just the attributes

       --backup[=CONTROL]
 Manual page cp(1) line 1 (press h for help or q to quit)
```

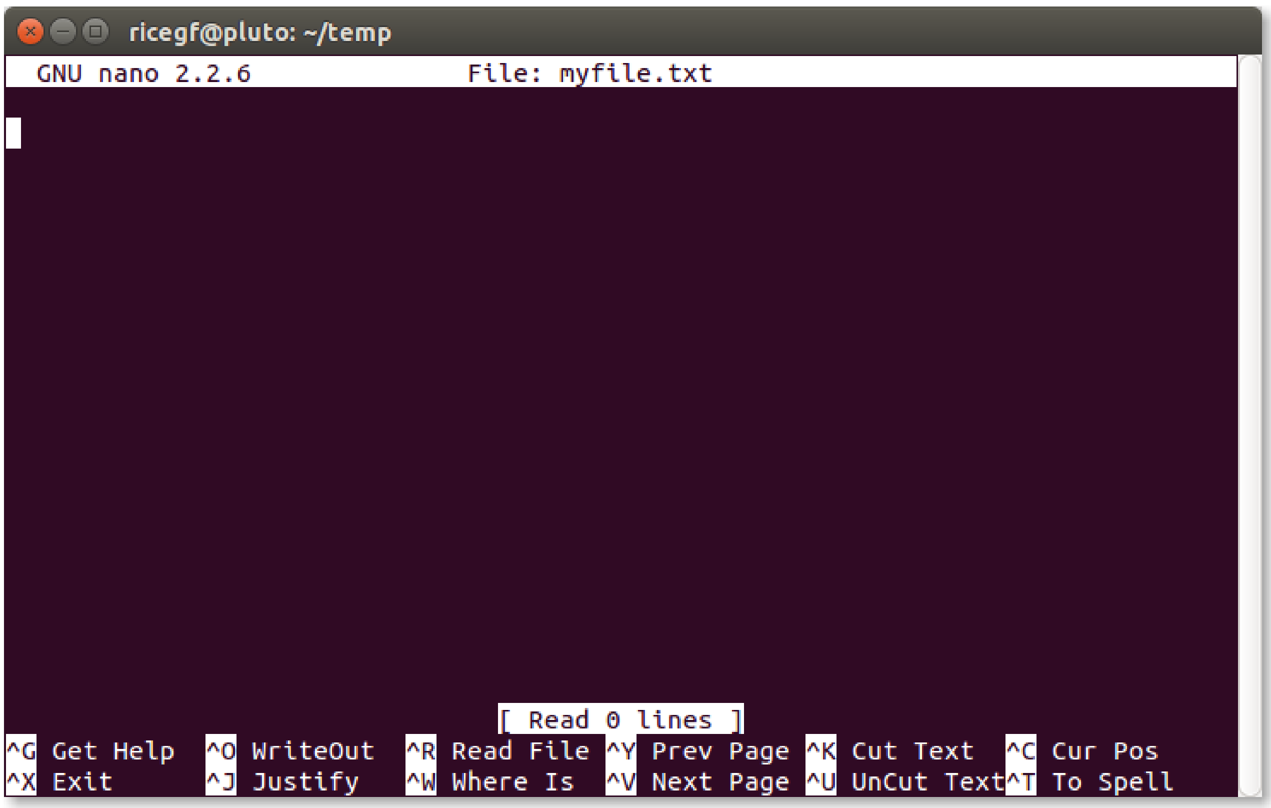
- **Page Up** and **Page Down** or the **mouse wheel** will move through the manual.
- **/[word]** (slash followed by a word) searches for the first occurrence of “word”. **n** then successively moves to each occurrence of “word”.
- **q** will quit the manual.

3 Editing a text file

- The default editor is gedit (roughly similar to notepad++), which opens in a GUI window. **gedit [filename]** will open the file in the editor, or use File → Open, or click the Open button. See https://wiki.gnome.org/Apps/Gedit for more information.



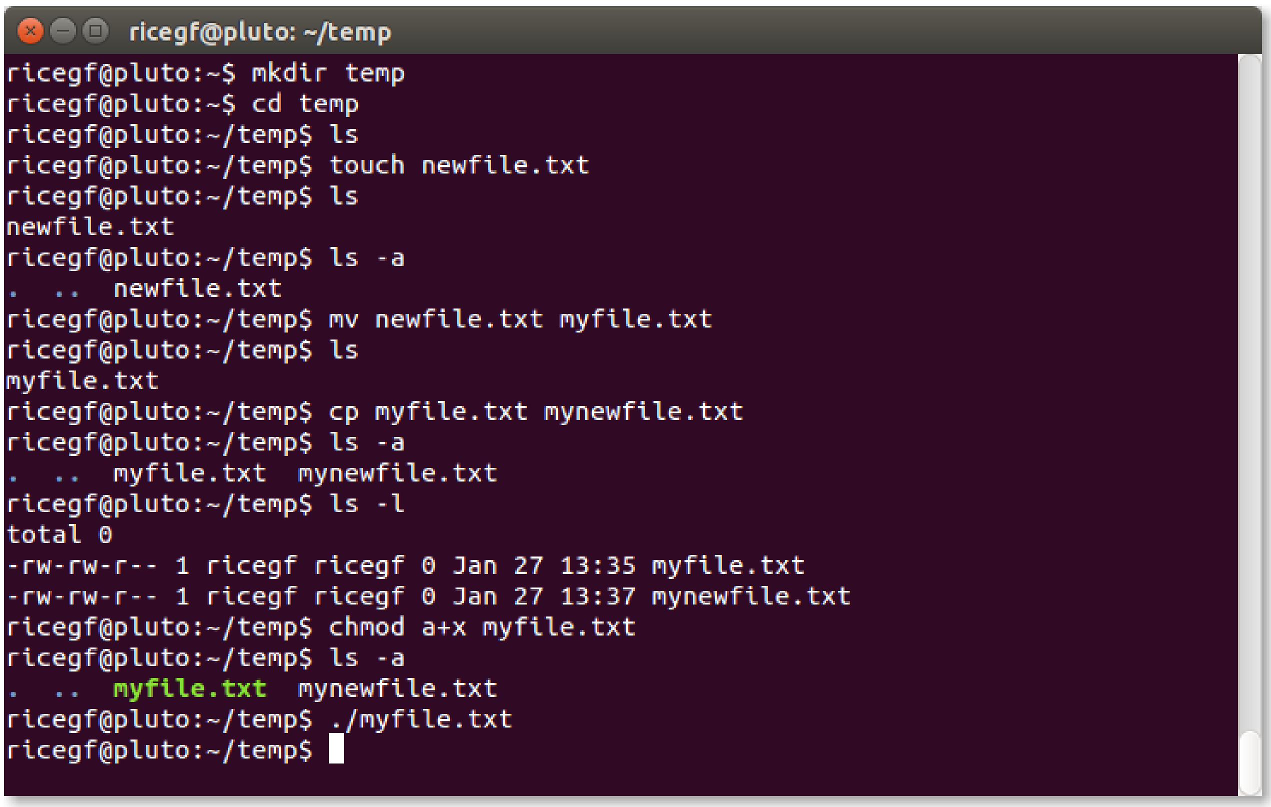
- To edit within the terminal, use nano. This is less similar to typical Windows or Mac editors. See https://www.nano-editor.org/dist/v2.2/nano.html for more information.



```
  GNU nano 2.2.6              File: myfile.txt




                                [ Read 0 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

4 Navigating directories and using files

- Paths are separated with forward slashes (/home/ricegf/), not backslashes as in Windows. No drive letters exist – all paths start with a slash (a "unified file system").
- **ls** will list the files in the current directory (like dir in Windows' cmd.exe)
    - **ls -l** will display a "long" listing with extra information
    - **ls -a** will show all files, including those that are hidden (e.g., start with a period)
- **mkdir [name]** will create a new directory with the given name (same as cmd.exe).
- **cd [directory]** will change to the specified directory (same as cmd.exe).
- **pushd [directory]** will change to the specified directory, but remember the current directory.
    - **popd** will return to the most recently remembered directory.
- **rmdir [directory]** will remove a directory, but only if it's empty (same as cmd.exe).
  rm -fr [directory] will remove a directory *and all of its contents*, no questions. **Be careful!**
  rm [file] removes a file permanently (no trash can).
- **mv [directory] [new_name]** will move a directory (or file) to a new name or directory.
  cp -r [directory] [copied_name] will copy a directory and all of its contents to a new directory.
  cp [file] [new_name] will copy a file to a new name or directory.
- **locate [partial_name]** will list all files on the computer that contain the partial_name.
- **grep [string] [filename(s)]** will search the filenames and list those containing the string.
- **cat [file(s)]** concatenates (types) the contents of all listed files to the console.
    - **head [file]** shows the first few lines of the file. **tail [file]** shows the last few lines.
    - **less [file]** pages through the file one screenful at a time, with Page Up and Page Down.
- **chmod a+x [file]** will make a file "executable" (like a .EXE in Windows). Gcc will automatically make programs it builds executable. **chmod** in general sets file permissions.
- **gnome-screenshot -a** will allow you to capture any area of the screen with the mouse.
  gnome-screenshot -i will allow you to select options interactively via a GUI.



```
ricegf@pluto:~$ mkdir temp
ricegf@pluto:~$ cd temp
ricegf@pluto:~/temp$ ls
ricegf@pluto:~/temp$ touch newfile.txt
ricegf@pluto:~/temp$ ls
newfile.txt
ricegf@pluto:~/temp$ ls -a
.  ..  newfile.txt
ricegf@pluto:~/temp$ mv newfile.txt myfile.txt
ricegf@pluto:~/temp$ ls
myfile.txt
ricegf@pluto:~/temp$ cp myfile.txt mynewfile.txt
ricegf@pluto:~/temp$ ls -a
.  ..  myfile.txt  mynewfile.txt
ricegf@pluto:~/temp$ ls -l
total 0
-rw-rw-r-- 1 ricegf ricegf 0 Jan 27 13:35 myfile.txt
-rw-rw-r-- 1 ricegf ricegf 0 Jan 27 13:37 mynewfile.txt
ricegf@pluto:~/temp$ chmod a+x myfile.txt
ricegf@pluto:~/temp$ ls -a
.  ..  myfile.txt  mynewfile.txt
ricegf@pluto:~/temp$ ./myfile.txt
ricegf@pluto:~/temp$ 
```

5 Combining commands via pipes and redirection

- **g++ --std=c++14 foo.gcc ; ./a.out** compiles and runs foo.gcc. The ; executes the left command, and when it exits, executes the right command.
- **./a.out > output.txt** sends the standard output (via cout) to the file named output.txt.
- **./a.out >> output.txt** appends the cout text to the existing file named output.txt.
- **./a.out > output.txt 2> errors.txt** sends the error output (via cerr) to the file named errors.txt.
- **./a.out < input.txt > output.txt** feeds the text from input.txt to standard input (aka cin).
- **./a.out | tee output.txt** sends the standard output (via cout) to both the console and output.txt. The | (pipe) connects cout from the left program to cin of the right program.

6 Loops, conditionals, and programmerish features

- **for f in $(ls) ; do mv $f $f.txt ; done** renames (moves) all files in the current directory to the same name with .txt appended. $([command]) is replaced by bash on the command line with the standard output of [command]. $f recalls the value of the f variable.
- **for i in $(seq 1 10) ; do echo $i ; done** counts from 1 to 10, once per line. echo (like print) just repeats its parameters to standard out.
- **while read line ; do echo $line >> myfile.txt ; done** appends each line of text entered at the console to the text file myfile.txt until EOF (end of file), which is control-d.
- **while :; do echo “This is the song that never ends” ; done** repeats the annoying song forever.
- **g++ --std=c++14 foo.cpp ; if [$? -eq 0]; then ./a.out ; fi** compiles foo.cpp and then runs it only if the compile succeeded.
- **time ./a.out** prints how long your program runs before exiting
- **zip -r [directory]** creates a ZIP archive of the named directory named directory.zip.
  - **unzip file.zip** unzips the zip file to the current directory.
  - The name of the current directory is a dot (“.”), and the parent is two dots (“..”).
- **diff file1.cpp file2.cpp** displays all differences between the two files. Lines in file1.cpp that aren't in file2.cpp will be preceded by “<”, while lines in file2.cpp not in file1.cpp will be preceded by “>”.
- **ps** lists all processes (commands) with their process id (“pid”) running in the current bash shell.
  ps -ef lists all processes / pids running on the computer.
  top periodically lists the “heaviest” processes running on your computer (**q** exits).
- **kill [pid]** terminates the process with the specified process id (the “pid”).
  kill -9 [pid] terminates the process with the specified pid with extreme prejudice.
  xkill terminates the next GUI program you click. **Be careful!**
- **which [command]** lists the full pathname of the command specified**.**
- **sudo [command]** executes the command as the administrator. **Be careful!**
  sudo apt-get install [program] installs the requested program from the Ubuntu app store.
- Edit .bashrc, and add *exactly* this at the end:
  ```
  alias backup='DIR=../$(basename $PWD)-$(date +%Y%m%d-%H%M%S);mkdir -p $DIR;cp -ru . $DIR'
  ```
  Then type **backup** anytime to make a perfect timestamped copy of the current directory alongside it in the parent. This will include a snapshot of the local git repository, if any.