Introduction to Unix: Fundamental Commands

Ricky Patterson – UVA Library

Based on slides from Turgut Yilmaz – Istanbul Teknik University
• **What We Will Learn**

• The fundamental commands of the *Unix* operating system.

• Everything here is also applicable to the *Linux* operating system.
• What Is UNIX?

• UNIX is a computer operating system, a control program that works with users to
  – run programs,
  – manage resources, and
  – communicate with other computer systems.

• Several people can use a UNIX computer at the same time; hence UNIX is called a multiuser system. Any of these users can also run multiple programs at the same time; hence UNIX is called multitasking.
Shell Commands of UNIX

- Unix Commands

- When you first log into a unix system, you are presented with something that looks like the following:

  /home/ricky#  meander:>

- That “something” is called a prompt. As its name would suggest, it is prompting you to enter a command.

- Every unix command is a sequence of letters, numbers and characters. But there are no spaces.
• Unix is also case-sensitive. This means that *cat* and *Cat* are different commands.

• The prompt is displayed by a special program called the *shell*.

• **Shells accept** commands, and **run** those commands.

• They can also be programmed in their own language. These programs are called “**shell scripts**”. Shell scripts are extremely powerful, but beyond the scope of this introduction.
• When you first login, the prompt is displayed by bash, and you are running your first unix program, the bash shell.

• As long as you are logged in, the bash shell will constantly be running.

• Other shells are available, including csh, tcsh, and ksh.
Unix Commands

obtaining help

- The man command displays reference pages for the command you specify.
- The UNIX man pages (man is short for manual) cover every command available.
- To search for a man page, enter man followed by the name of the command to find.
- For example:

  ricky@meander:~>man ls
NAME

ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuSUX nor --sort.

-a, --all
   do not hide entries starting with .

-A, --almost-all
   do not list implied . and ..

-b, --escape
   print octal escapes for nongraphic characters
• **man** *(obtaining help)*

• There is also a **keyword function** in **man**.

• For example;
  – If you are interested in any commands that deal with **Postscript**, the printer control language for **Adobe**
  – Type **man -k ps** or **man -k Postscript**, you’ll get a listing of all commands, system calls, and other documented parts of **unix** that have the word “ps” (or “Postscript”) in their name or short description.

• This can be very useful when you’re looking for a tool to do something, but you don’t know it’s name-or if it even exists!
• **cat**

• **cat** command is used to concatenate or displays the contents of a file.

• To use it, type **cat**, and then press **enter** key:

  ```
  /home/larry# cat
  ```

• This produces the correct result and runs the cat program.

• To end many unix command, type end-of-file command (EOF) 
  
  ```
  [hold down the key labeled “Ctrl” and press “d” (Ctrl+d) ]
  ```
To display the contents of a file, type

```
cat filename
```
• To see Linux commands press **Tab** key,
• If you want to learn commands beginning with **c** you can write **c** then press **Tab** key

```
/home/larry#  c
```

<table>
<thead>
<tr>
<th>c++</th>
<th>chage</th>
<th>codepage</th>
<th>continue</th>
</tr>
</thead>
<tbody>
<tr>
<td>c++decl</td>
<td>charset</td>
<td>col</td>
<td>control-panel</td>
</tr>
<tr>
<td>c++filt</td>
<td>chattr</td>
<td>colcrt</td>
<td>convert_smbpasswd</td>
</tr>
<tr>
<td>c2ph</td>
<td>checkalias</td>
<td>collateindex.pl</td>
<td>cp</td>
</tr>
<tr>
<td>c_rehash</td>
<td>chfn</td>
<td>colrm</td>
<td>cpio</td>
</tr>
<tr>
<td>cal</td>
<td>chgrp</td>
<td>column</td>
<td>cpp</td>
</tr>
<tr>
<td>calibrate_ppa</td>
<td>chmod</td>
<td>comm</td>
<td>cproto</td>
</tr>
<tr>
<td>cancel</td>
<td>chown</td>
<td>command</td>
<td>crontab</td>
</tr>
<tr>
<td>captinfo</td>
<td>chsh</td>
<td>comp</td>
<td>csh</td>
</tr>
<tr>
<td>card</td>
<td>chut</td>
<td>comp_err</td>
<td>csplit</td>
</tr>
<tr>
<td>case</td>
<td>ci</td>
<td>compgen</td>
<td>ctags</td>
</tr>
<tr>
<td>cat</td>
<td>cjpeg</td>
<td>compile_et</td>
<td>cut</td>
</tr>
<tr>
<td>catchsegv</td>
<td>cksum</td>
<td>complete</td>
<td>cvs</td>
</tr>
<tr>
<td>cc</td>
<td>clear</td>
<td>composeglyphs</td>
<td>cvshug</td>
</tr>
<tr>
<td>cd</td>
<td>cmp</td>
<td>compress</td>
<td>cxpm</td>
</tr>
<tr>
<td>cdecl</td>
<td>cmuwmtopbm</td>
<td>consolechars</td>
<td>cytune</td>
</tr>
<tr>
<td>chacl</td>
<td>co</td>
<td>consolehelper</td>
<td></td>
</tr>
</tbody>
</table>
• **Storing information**

• Unix provides files and directories.
• A directory is like a folder: it contains pieces of paper, or files.
• A large folder can even hold other folders—directories can be inside directories.
• In unix, the collection of directories and files is called the file system. Initially, the file system consists of one directory, called the “root” directory.
• Inside the “root” directory, there are more directories, and inside those directories are files and yet more directories.
• Each file and each directory has a name.
• A short name for a file could be joe,
• while it’s “full name” would be /home/larry/joe. The full name is usually called the path.
• The path can be divide into a sequence of directories.
• For example, here is how /home/larry/joe is read:

The initial slash indicates the root directory. This signifies the directory called home. It is inside the root directory.

The second slash corresponds to the directory larry, which is inside home.

joe is inside larry.
- A **path** could refer to either a **directory** or a **filename**, so joe could be either.
- All the items before the short name must be directories.
• **Looking at directories with `ls`**

• The command `ls` lists files.

• If you try `ls` as a **command**, it will list the files (and directories) contained in the current directory.

If you have files, `ls` lists the names of files in the directory.
• If you want a list of files of a more active directory, try the root directory.

/my/home/larry#  ls /
    bin  etc   install  mnt  root  user  var
    dev  home  lib   proc  tmp  usr  vmlinux

“/” is a parameter saying what directory you want a list for. In this case, it is the top level directory “/”

Some commands have special parameters called options or switches. To see this try:

/my/home/larry#  ls –F /
    bin  etc/   install/  mnt/  root/  user/  var/
    dev/  home/  lib/   proc/  tmp/  usr/  vmlinux/

The -F is an option. It displays file types.
• An option is a special kind of parameter that starts with a dash “-”
• An option modifies how the program runs, but not what the program runs on.
• For ls, -F is an option that lets you see which things are directories, which ones are special files, which are programs, and which are normal files.
• Anything with a trailing slash “/” is a directory.
• ls -l file* displays files starting with “file”
• ls –l displays all details
• Many unix commands are like `ls`.

• They have **options**, which are generally one character after a dash, and they have **parameters**.

• Unlike `ls`, some commands *require* certain parameters and/or options. You have to learn these commands.
• **pwd**

  • **pwd** (present working directory) tells you your current directory.
  
    – Most commands act, by default, on the current directory.  
      
      *For instance, `ls` without any parameters displays the contents of the current directory.*  

• **cd**

  • **cd** is used to change directories.
  
    • The format of this command:
      
      `cd new-directory` (where new-directory is the name of the new directory you want).
• For instance, try:

```
/home/larry# cd /home
/home#
```

• If you **omit the optional parameter directory**, you’re **returned to your home**, or original directory (the same as typing `cd ~`). Otherwise, `cd` will change you to the specified directory.

• There are two directories used only for relative pathnames:
  • The directory “.” refers to the **current directory**
  • The directory “..” refers to the **parent directory** of the current directory
• The directory “..” is most useful moving back up a directory: `cd ..`
• The command “cd –” will return you to the most recent directory visited.
**mkdir**

`mkdir (make directory)` is used to create a new directory,
- It can take more than one parameter, interpreting each parameter as another directory to create.
- By default, it will create the new directory as a subdirectory of the current directory

**rmdir**

`rmdir (remove directory)` is used to remove a directory,
- `rmdir` will refuse to remove a non-existant directory, as well as a directory that has anything in it.
• **Moving Information**

• The primary commands for manipulating files under Unix are `cp`, `mv`, and `rm`. They stand for **copy**, **move**, and **remove**, respectively.

• **cp**

• `cp` is used to copy contents of file1 to file2

  \[
  \text{cp file1 file2 (contents of file1 is copied to file2 in the same directory)}
  \]

  \[
  \text{cp folder1/file1 folder2 (contents of file1 is copied to file1 in the inside of folder2 directory)}
  \]
• **rm**

  • `rm` is used to remove a file.
    – `rm filename`  --- > removes a file named `filename`

• **mv**

  • `mv` is used to move a file.
    – `mv filename /path/newname`  --- > moves a file named `filename` to a new location, with a new name

  • looks like `cp`, except that it deletes the original file after copying it.

  • `mv` will rename a file if the second parameter is a file. If the second parameter is a directory, `mv` will move the file to the new directory, keeping it’s shortname the same.
Some Other UNIX Commands

- The Power of Unix

- The power of unix is hidden in small commands that don’t seem too useful when used alone, but when combined with other commands produce a system that’s much more powerful, and flexible than most other operating systems.
- The commands include `sort`, `grep`, `more`, `cat`, `wc`, `spell`, `diff`, `head`, and `tail`. 
Operating on Files

In addition to the commands like `cd`, `mv`, and `rm`, you learned in shell section, there are other commands that just operate on files, but not the data in them.

- These include `touch`, `chmod`, `du`, and `df`.
- All of these commands don’t care what is in the file.
Some of the things these commands manipulate:

- **The time stamp**: Each file has three dates associated with it. These are *creation time*, *last modification time* and *last access time*.

- **The owner**: the owner of files

- **The group**: the group of users

- **The permissions**: read, write, execute permissions of files. The permissions tell Unix who can access what file, or change it, or, in the case of programs, execute it. Each of these permissions can be toggled separately for the owner, the group, and all the other users.
• touch will update the time stamps of the files listed on the command line to the current time.
• If a file doesn’t exist, touch will create it.
• **chmod**

• *chmod* (change mode) is used to change the permissions on a file.

  (owner) (group) (others)

  chmod [number][number][number] file1

  Number = (read)4 + (write)2 + (execute)1

• Example: chmod 754 file1

  for owner: read, write and execute permissions (4+2+1)
  for group: read and execute permissions (4+0+1)
  for others: only read permission (4+0+0)
• **chmod**

• **chmod** can also be set in alpha mode (non-octal)

  (owner)  (group)  (others)

  chmod [user/group/others/all]operator[permission]
  [file(s)]

  operator can be +, -, or =

• Example:   chmod u+rwx,g+rx,o+r file1

  for **owner**: *read, write and execute* permissions (u+rwx)
  for **group**: *read* and *execute* permissions (g+rx)
  for **others**: only *read* permission (o+r)
• System Statistics

• Commands in this section will display statistics about the operating system, or a part of the operating system.

• du

  du (disk usage) will count the amount of disk space for a given directory, and all its subdirectories take up on the disk.

• df

  df (disk filling) summarizes the amount of disk space in use. For each file system, it shows the total amount of disk space, the amount used, the amount available, and the total capacity of the file system that’s used.
• **What’s in the File?**

• There are two major commands used in unix for listing files, *cat*, and *more*.

• **cat**

• *cat* shows the contents of the file.

  ```bash
cat [-nA] [file1 file2 ... fileN]
  
  cat is not a user friendly command-it doesn’t wait for you to read the file, and is mostly used in conjunction with pipes.

• However, *cat* does have some useful command-line options. For instance, *n* will number all the lines in the file, and *A* will show control characters.
• more

• more is much more useful, and is the command that you’ll want to use when browsing ASCII text files

    more [-l] [+linenumber{]} [file1 file2 ... fileN]

• The only interesting option is l, which will tell more that you aren't interested in treating the character Ctrl-L} as a `new page" character. more will start on a specified linenumber.

• head

    head will display the first ten lines in the listed files.

    head [-lines{]} [l file1 file2 ... fileN]

• Any numeric option will be taken as the number of lines to print, so head -15 frog will print the first fifteen lines of the file frog
• **tail**
  
  Like *head*, **tail** display only a fraction of the file.
  *tail* accepts an option specifying the number of lines.
  
  ```
  tail [-lines] [file1 file2 ... fileN]
  ```

• **file**
  
  The **file** command attempts to identify what format a particular file is written in.
  
  ```
  file [file1 file2 ... fileN]
  ```

  Since not all files have extensions or other easy to identify marks, the *file* command performs some rudimentary checks to try and figure out exactly what it contains.
Information Commands

The commands that will alter a file, perform a certain operation on the file, or display statistics on the file.

grep

- grep is the generalized regular expression parser.
- This is a fancy name for a utility which can only search a text file.

```
grep [-nvwx] [-number] { expression} [file1 file2 ... fileN]
```
• **WC**

  • *wc* (word count) simply counts the number of words, lines, and characters in the file(s).

    ```
    wc [-clw] [file1 file2 ... fileN]
    ```

  • The three parameters, *clw*, stand for character, line, and word respectively, and tell *wc* which of the three to count.

• **spell**

  • *spell* is very simple unix spelling program, usually for American English. *spell* is a filter, like most of the other programs we’ve talked about.

    ```
    spell [file1 file2 ... fileN]
    ```
• **diff**

• The GNU version of **diff** has over twenty command line options. It shows you what the differences are between two files

• **diff file1 file2**
These three programs are used to **compress** and **decompress** data.

- **gzip**, or GNU Zip, is the program that reads in the original file(s) and outputs files that are compressed, and therefore smaller.

- **gzip** deletes the files specified on the command line and replaces them with files that have an identical name except that they have “.gz” appended to them.
More help

• Lynda.com course:
  – “UNIX for Mac OS X Users” (most applies to all flavors of UNIX)
  – Be sure to access it using via the Library’s Research Portal to take advantage of the UVa subscription: 
    http://www.library.virginia.edu/research/
  – Then click on Lynda.com link

• Lots of online resources
  – http://www.doc.ic.ac.uk/~wjk/UnixIntro/