Leveraging Linux Capabilities

- **The Shell Prompt**
- ***file*** **Command**
- **Pipes**
- ***date*** **Command**
- **Text File Conversion**
- **File Compression/Storage**
- **Getting File Information**
- **Getting User Information**

The Shell Prompt

- In the last class I mentioned **shell variables**
- We'll talk a lot more about them in a few weeks
- But right now I'd like to say a few words about one particular variable
- PS1 is the variable that determines your Unix prompt
- This variable can be customized in many ways to provide all sort of useful information
- By default PS1 tells you three things
  - Your Unix username
  - The machine to which you are connected
  - The location of your current directory

The Shell Prompt

- Some of you may have noticed that my prompt looks different from yours
- That's because I have customized the value of PS1 for my account
- But before I did this, my prompt looked something like this

```
ckelly@vm75:~/it244$
```

- Prompt parts:
  - The characters **before the @** show my Unix username
  - The characters **between @ and :** are the **hostname** of the machine I am using
  - The characters **from : to $** show my current directory

The Shell Prompt

- The ~ symbol indicates your home directory
- We'll talk more about this in a future class
- So when I got this prompt, I was in the directory

  `/home/ckelly/it244/work`

- If you look carefully at your prompt, you can always see where you are in the Unix filesystem

file - Show the File Type

- The *file* utility takes an argument of one or more files and shows the type of each:

```
$ file *
bin:                    directory
cars.txt:               ASCII text
cmds:                   directory
dead.letter:            ASCII news text
downloads:              directory
exercises_it244:        directory
f11_it244_class_web:    directory
hw_it244:               directory
it244:                  symbolic link to
`/courses/it244/s12/ghoffmn/'
java:                   directory
```

Pipes - Stringing Programs Together

- **Pipes** are one of the most powerful features of Unix
- A pipe takes the output of one Unix command and feeds it into the input of another
- Using pipes you can string programs together so they can perform a task that none of them could do separately
- A pipe consists of two or more Unix commands each one separated from the one that came before by the | symbol
- Using a pipe, we can build up a string of commands to get exactly what we want

Pipes - Stringing Programs Together

- Let's say we want to find all Red Sox wins **against the Rays** *sorted in reverse order by date* from the following file

```
$ cat red_sox.txt
2011-07-02   Red Sox @  Astros     Win 7-5
2011-07-03   Red Sox @  Astros     Win 2-1
2011-07-04   Red Sox vs Blue Jays  Loss 7-9
2011-07-05   Red Sox vs Blue Jays  Win 3-2
2011-07-06   Red Sox vs Blue Jays  Win 6-4
2011-07-07   Red Sox vs Orioles    Win 10-4
2011-07-08   Red Sox vs Orioles    Win 10-3
2011-07-09   Red Sox vs Orioles    Win 4-0
2011-07-10   Red Sox vs Orioles    Win 8-6
2011-07-15   Red Sox @  Rays       Loss 6-9
2011-07-16   Red Sox @  Rays       Win 9-5
2011-07-17   Red Sox @  Rays       Win 1-0
...
```

```
2011-07-17   Red Sox @  Rays       Win 1-0
2011-07-18   Red Sox @  Orioles    Win 15-10
2011-07-19   Red Sox @  Orioles    Loss 2-6
2011-07-20   Red Sox @  Orioles    Win 4-0
2011-07-22   Red Sox vs Mariners   Win 7-4
2011-07-23   Red Sox vs Mariners   Win 3-1
2011-07-24   Red Sox vs Mariners   Win 12-8
2011-07-25   Red Sox vs Royals     Loss 1-3
2011-07-26   Red Sox vs Royals     Win 13-9
2011-07-27   Red Sox vs Royals     Win 12-5
2011-07-28   Red Sox vs Royals     Loss 3-4
2011-07-29   Red Sox @  White Sox  Loss 1-3
2011-07-30   Red Sox @  White Sox  Win 10-2
2011-07-31   Red Sox @  White Sox  Win 5-3
```

Pipes - Stringing Programs Together

- First, let's find all games against the Rays

```
$ grep Rays red_sox.txt
2011-07-15        Red Sox @  Rays          Loss 6-9
2011-07-16        Red Sox @  Rays          Win 9-5
2011-07-17        Red Sox @  Rays          Win 1-0
```

- Now let's *feed this into* another command that selects the games the Sox won

```
$ grep Rays red_sox.txt | grep Win
2011-07-16        Red Sox @  Rays          Win 9-5
2011-07-17        Red Sox @  Rays          Win 1-0
```

Pipes - Stringing Programs Together

- Now we can use *sort* to get the results in the order we want

```
$ grep Rays red_sox.txt | grep Win | sort -r
2011-07-17      Red Sox @  Rays          Win 1-0
2011-07-16      Red Sox @  Rays          Win 9-5
```

- The Unix tool philosophy is:
  - simple programs that...
  - ...do one thing well
- Pipes are essential in making this philosophy work
- As we progress through this course, you will have many opportunities to use pipes

date - Get the Date and Time

- *date* displays the time and date

```
$ date
Tue Aug  7 20:02:48 EDT 2012
```

- You can change the way the date is displayed by following *date* with a + and a format string

```
$ date +"%Y-%m-%d %r"
2012-08-07 08:19:44 PM
```

- The format string consists mostly of pairs of character pairs the first character of which is a **%**

date - Get the Date and Time

- In the string on the previous slide, **%Y** stands for the four digit year
- To get more information on the various formatting options:

```
info date
```

- Now move the cursor down to the line that reads

```
* Date conversion specifiers::      %[aAbBcCdDeFgGhjmuUVwWxyY]
```

- Then hit the *Enter* key

Text File Conversion Programs

- Text files on Unix differ from those on Windows machines by the characters used to mark the end of a line
- There are two packages that provide software to convert between these two formats
  - ***tofrodos***
  - ***unix2dos***
- We won't be using either of them in this course but you should know about them

Text File Conversion Programs

- The ***tofrodos*** package provides:
  - *todos* to convert Unix files to Windows format, and...
  - *fromdos* to go the other way
- The ***unix2dos*** package uses:
  - *unix2dos* to convert Unix text files to Windows text, and...
  - *dos2unix* to go the other way
- However, you will not be expected to know this for a quiz or exam

Compressing Files with *bzip2*

- On the Internet there are many files which are free for the taking
- Many of these files are huge and copying them to a machine can take a long time
- To speed up the process, big files are usually ***compressed***
- Compression utilities are used to do this
- **`bzip2`** is one such utility

Compressing Files with *bzip2*

- It achieves the highest compression ratio of all common compression utilities
- You run *bzip2* like this

```
bzip2 FILENAME
```

- *bzip2* compresses the file creating a new file with the extension .bz2 and deletes the original file
- If you need to keep the original file run *bzip2* with the -k (for keep) option

Compressing Files with *bzip2*

- To decompress a file created by *bzip2* use *bunzip2* like this

```
bunzip2 FILENAME.bz2
```

- *bunzip2* will:
  - decompress the .bz2 file and...
  - ...create a new file with the .bz2 extension removed
- The compressed file is also deleted

Compressing Files with *bzip2*

- A file that *bunzip2* has compressed is unreadable
- If you want to look at the contents of a .bz2 compressed file *without* uncompressing it use *bzcat*
  - *bzcat*, will print the uncompressed contents of a file to the terminal
  - It ***does not*** alter the original, compressed file

gzip **- the GNU Compression Utility**

- The GNU project created *gzip* to compress files
- It is older, and less efficient, than *bzip2*
- But many open source packages are compressed with this program
- It is similar in operation to *bzip2*
- The compressed files *gzip* creates have a .gz extension

gzip **- the GNU Compression Utility**

- To *display* the contents of a .gz file without converting it, use *zcat*
- To *decompress* a gzipped file, use *gunzip*
- These utilities have nothing to do with the zip and unzip programs that are frequently used on Wintel machines
  - (**Tip:** What does "*Wintel*" mean? Look it up!)

tar **- Packaging Files for Transfer or Storage**

- Most software packages consist of many files
- But...to distribute the package efficiently you really want to turn these many files into a single file
- This also needs to be done when backing up a directory
- *tar* (**t**ape **ar**chive) is the Unix utility used for this purpose
- *tar* does not compress files
  - It stuffs multiple files into a single file, often called a **tarball**
  - *tar* is usually used along with a compression program

tar **- Packaging Files for Transfer or Storage**

- First you run *tar* to create a single file then you run another utility to compress the file
- You also use *tar* to unpack the files
- You run *tar* with different options to pack, or unpack, a tarball
- To "tar up" a set of files, run

```
tar -cvf ARCHIVE_NAME.tar DIRECTORY_NAME
```

- The options stand for **c**reate, **v**erbose, **f**ile
  - The .tar extension is a convention
  - Though you do not ***have*** to use this extension, it would be foolish not to

tar **- Packaging Files for Transfer or Storage**

- To see the files contained in a tar file without unpacking them use

```
tar -tvf ARCHIVE_FILE
```

- To unpack a tarball use

```
tar -xvf ARCHIVE_FILE
```

- The x option stands for e**x**tract
- Normally, you *create a tarball* and then *run a compression program* on the archive file

tar - Packaging Files for Transfer or Storage

- If **bzip2** is used for compression the new file will often have a `.tar.bz2` or `.tbz` extension
- If **gzip** is used the extensions usually are `.tar.gz`, `.tgz`, or sometimes simply `.gz`
- Again, these are simply conventions
  - That said, don't violate these conventions unless you have a **very** good reason
  - Life is complicated enough as it is
  - Plus, you don't want to create confusion for others

which - Finding a Program File

- Unix commands are programs
  - which exist as binary files
  - containing numeric codes that the computer's processor understands
  - that are located somewhere in the filesystem
- You can use the Unix utility *which* to find the exact location of any binary program file

which - Finding a Program File

- To find the location of the *tar* program file we would run

```
$ which tar
/bin/tar
```

- *which* shows that the executable file for *tar* is located in the /bin directory
  - *which* uses the **PATH** system variable to find the location of the file
  - We'll discuss **PATH** in a future class

whereis - Finding Files Used by a Program

- *whereis* is another program that can be used to locate program files
- *whereis* takes an approach different from that of *which*
- Every Unix or Linux system has certain standard places where it stores programs and the files they use
  - *whereis* searches these locations
  - It returns a list of all files associated with a program
  - The list gives the name of the file as well as it's location

***whereis* - Finding Files Used by a Program**

- When we run *whereis* on *tar*, we get more information than *which* returned

```
$ whereis tar
tar: /bin/tar /usr/include/tar.h /usr/share/man/man1/tar.1.gz
```

- We get a fuller view of the command:
  - The *first* entry is the executable file /bin/tar
  - The *second* entry is a header file /usr/include/tar.h
  - The program needs the header file to get certain information
  - The *third* entry is the file that *man* uses to provide information about *tar*

whereis - Finding Files Used by a Program

- A word of caution about using *which* and *whereis*
- Some commands are actually built into the shell itself
- These command are called built-ins and we will talk about them in a future class
- If you run *which* or *whereis* on these programs you will get nothing back

```
$ which cd
$
```

locate - Search for Any File

- *which* and *whereis* only work on programs
- *locate* can be used to find any file
- You don't need to know the full name of a file to use *locate*
- *locate* will search on a partial file name

```
$ locate foot
/etc/update-motd.d/99-footer
/usr/share/doc/java-common/debian-java-faq/footnotes.html
/usr/share/emacs/23.3/lisp/mail/footnote.elc
/usr/share/emacs/23.3/lisp/org/org-footnote.elc
/usr/share/libparse-debianchangelog-perl/footer.tmpl
/usr/share/xml-core/catalog.footer
...
```

locate **-** sample output continued

```
...
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/Kconfig
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/Makefile
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/Makefile.boot
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/debug-macro.S
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/entry-macro.S
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/hardware.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/io.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/irqs.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/isa-dma.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/memory.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/system.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/timex.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/uncompress.h
/usr/src/linux-headers-3.0.0-12/arch/arm/mach-footbridge/include/mach/vmalloc.h
```

locate **- Search for Any File**

- *locate* does not actually search the file system itself
  - That would take too long
  - Instead, it uses a database of all files on the system
    - This database is created by another program ***updatedb***
    - *updatedb* is usually run automatically in the background to update the database
- For some reason, in the past, the *locate* command **only** worked on it244a
- It may work on others, now

find **- Search for Files Using Different Criteria**

- The most powerful Linux/Unix command for finding files is ***find***
- Unfortunately, it's power makes it harder to use that *locate*
- *find* can be used to search for a file based on many different things such as:
  - The name of the file
  - The last time the file was used
  - The last time the file was changed
  - The access permission of the file
- An in-depth discussion of *find* is beyond the scope of this course, but since *locate* may not always be available on the UMB machines, I need to talk a bit about it

find - Search for Files Using Different Criteria

- The simplest way to use *find* is searching for files by name
- You do this using the following format

```
find  DIRECTORY  -name  FILENAME
```

- Here is an example

```
$ find  /home/ghoffmn  -name red_sox.txt

/home/ghoffmn/course_files/it244_files/red_sox.txt

/home/ghoffmn/course_files/it441_files/red_sox.txt
```

- Unlike *locate*, the *find* command will not accept a partial file name

```
$ find  /home/ghoffmn  -name red
$
```

find - Search for Files Using Different Criteria

- You can get the same affect by using the * character but since * has special meaning on the command line you have to escape it

```
$ find /home/ghoffmn -name memo.\*
/home/ghoffmn/memo.bak
/home/ghoffmn/memo.txt
/home/ghoffmn/tmp/memo.bak
/home/ghoffmn/tmp/memo.txt
/home/ghoffmn/tmp/memo.backup
```

who - See Users Logged On

- *who* prints a list of all users currently logged on to the machine

```
$ who
ghoffmn  pts/0        2012-08-12 13:41 (dsl092-066-161.bos1.dsl.speakeasy.net)
rouilj   pts/1        2012-08-12 04:25 (pool-74-104-161-
40.bstnma.fios.verizon.net)
eb       pts/2        2012-08-12 08:19 (pool-96-237-251-
11.bstnma.fios.verizon.net)
```

- *who* also provides information about each user's login session
  - It shows the time they logged in
  - It also shows the machine from which the user connected

who - See Users Logged On

- *who am i* will show the user who is logged into a specific terminal

```
$ who am i
ghoffmn  pts/0   2012-08-12 13:41 (dsl092-066-161.bos1.dsl.speakeasy.net)
```

- This can be useful if you find an unattended terminal
- You can run the same command without the spaces but it gives less information

```
$ whoami
ghoffmn
```

finger - Get information on Users

- *finger* provides information about Unix accounts:

```
$ finger ghoffmn
Login: ghoffmn                          Name: Glenn Hoffman
Directory: /home/ghoffmn                        Shell: /bin/bash
On since Wed Sep 17 16:09 (EDT) on pts/1 from dsl092-066-
161.bos1.dsl.speakeasy.net
   1 second idle
Mail forwarded to glennhoffman@mac.com
Mail last read Thu Sep  4 15:12 2014 (EDT)
Plan:
Office:          McCormack M-3-607                      Fall 2014
Office Hours:    Tuesday & Thursday, 10:00 - 12:00 PM and by appointment
Classes:
    IT 341-2  Introduction to System Administration  TuTh  12:30-1:45  S3-148
(IT Lab)
    IT 244-1  Introduction to Linux/Unix             TuTh   2:00-3:15  S3-028
(Web Lab)
...
```

finger - Get information on Users

- *finger*, like *mv*, has two functions
- When used without an argument *finger* shows every user currently logged in

```
$ finger
Login     Name                    Tty       Idle  Login Time
Office     Office Phone
ghoffmn   Glenn Hoffman           pts/0           Aug 18 11:13
(dsl092-066-161.bos1.dsl.speakeasy.net)
rouilj    John P. Rouillard       pts/1     4:34  Aug 18 06:44
(pool-74-104-161-40.bstnma.fios.verizon.net)
ubuntu    Ubuntu Dummy           *tty1       14d  Aug  4 04:53
```

finger - Get information on Users

- You can also use a last name with *finger*

```
$ finger hoffman
    Login: ghoffmn                  Name: Glenn Hoffman
    Directory: /home/ghoffmn                Shell: /bin/bash
    On since Wed Sep 17 16:09 (EDT) on pts/1 from dsl092-066-161.bos1.dsl.speakeasy.net
       1 second idle
    Mail forwarded to glennhoffman@mac.com
    Mail last read Thu Sep  4 15:12 2014 (EDT)
    Plan:
    Office:         McCormack M-3-607               Fall 2014
    Office Hours:   Tuesday & Thursday, 10:00 - 12:00 PM and by appointment
    Classes:
        IT 341-2  Introduction to System Administration  TuTh  12:30-1:45  S3-148  (IT Lab)
...

    Login: it244gh                  Name: Dummy for Glenn Hoffman
    Directory: /home/it244gh                Shell: /users/nologin
    Never logged in.
    Mail forwarded to glennhoffman@mac.com
    No mail.
    Plan:
    This account is a test account for Glenn Hoffman teaching it244
```

finger - Get information on Users

- Or a first name

```
$ finger hoffman
    Login: ghoffmn                  Name: Glenn Hoffman
    Directory: /home/ghoffmn                Shell: /bin/bash
    On since Wed Sep 17 16:09 (EDT) on pts/1 from dsl092-066-161.bos1.dsl.speakeasy.net
       1 second idle
    Mail forwarded to glennhoffman@mac.com
    Mail last read Thu Sep  4 15:12 2014 (EDT)
    Plan:
    Office:         McCormack M-3-607                Fall 2014
    Office Hours:   Tuesday & Thursday, 10:00 - 12:00 PM and by appointment
    Classes:
        IT 341-2  Introduction to System Administration  TuTh  12:30-1:45  S3-148  (IT Lab)
...

    Login: it244gh                  Name: Dummy for Glenn Hoffman
    Directory: /home/it244gh                Shell: /users/nologin
    Never logged in.
    Mail forwarded to glennhoffman@mac.com
    No mail.
    Plan:
    This account is a test account for Glenn Hoffman teaching it244
```