Objectives

- Command Line
  - Window Anatomy
  - Command Structure
  - Command Examples
- Help
- Files and Directories
- Permissions
- Wildcards and Home (~)
- Redirection and Pipe
- Create and Edit files
Command Line
Unix Basics

- Multi-user
- Case-sensitive
- Bash shell, command-line
- Commands
Command Window Anatomy

Click in the title bar to bring the window to the front and make it active.
Command Window Anatomy

Appears as the first line of a login shell.
Command Window Anatomy

Prompts
Appears at the beginning of a line and usually ends in $.
Command Window Anatomy

Command input

Place to type commands, which may have options and/or arguments.
Command Window Anatomy

Place for command response, which may be many lines long.
Command Window Anatomy

Input cursor

Typed text will appear at the cursor location.
Scroll Bar

Will appear as needed when there are more lines than fit in the window.
Use the mouse to change the window size from the default 80x24.
Command Structure

*command* [arguments]

- Commands are made up of the actual *command* and its *arguments*

*command* -options [arguments]

- The arguments are further broken down into the command *options* which are prefixed by a “-” and other *arguments* that identify data for the command.
Command Examples

Try these commands:

```
[train@centos ~]$ date
Tue Dec 14 14:11:42 EST 2010
[train@centos ~]$ cal
    December 2010
   Su Mo Tu We Th Fr Sa
          1 2 3 4
   5 6 7 8 9 10 11
  12 13 14 15 16 17 18
  19 20 21 22 23 24 25
  26 27 28 29 30 31

[train@centos ~]$ hostname
centos.css.udel.edu
[train@centos ~]$ pwd
/home/train
[train@centos ~]$ whoami
train
[train@centos ~]$ ps
  PID TTY          TIME CMD
19189 pts/2 00:00:00 bash
19217 pts/2 00:00:00 ps
[train@centos ~]$ uptime
14:12:01 up 7 days, 17 min, 3 users, load average: 0.00, 0.00, 0.00
[train@centos ~]$
```
More Command Examples

Try these commands:

```
[train@centos ~]$ history
  1  date
  2  cal
  3  hostname
  4  pwd
  5  whoami
  6  ps
  7  uptime
  8  history

[train@centos ~]$ cal 10 2010
   October 2010
 Su Mo Tu We Th Fr Sa
       1 2
  3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
 31

[train@centos ~]$ uname
Linux
```

```
[train@centos ~]$ uname -a
Linux centos.css.udel.edu 2.6.18-194.26.1.el5 #1 SMP Tue Nov 9 12:54:20 EST 2010
 x86_64 x86_64 x86_64 GNU/Linux
```
Command Input Keys Summary

At the command prompt ($), use

- A to insert a character.
- return to enter the command.
- delete to erase character to left.
- ▼► to move to the left or right.
- ▲ to retrieve previous command.
- Tab for command line completion.
Command Input Keys Summary

At the command prompt ($), use

- **Control-C** or **Ctrl-c** or just **C-c**: interrupts a running program
- **Control-U**: deletes last line typed
- **Control-D** or **Ctrl-d** or just **C-d**: ends text input for many UNIX programs
Command Output Summary

- Lines longer than the column width are wrapped.

Use the mouse in the resize handle to increase the width of the screen and unwrap the lines.

- Lines are scrolled off the top and the prompt will appear at the bottom of the screen.

Use the mouse in the scroll bar to see lines that have scrolled off the screen.
Help: man, info, apropos
Getting Help

**man** command
- formats and displays on-line manual pages (i.e., documentation) for *command*.

**info [subject]**
- formats and displays online documentation for a particular subject that are easily searchable.

**apropos** *keyword*
- searches the whatis database for commands containing the *keyword*
Getting Help: man

man bash

```
train@centos:~$ man bash

bash(1)

NAME
bash - GNU Bourne-Again SHell

SYNOPSIS
bash [options] [file]

COPYRIGHT
Bash is Copyright (C) 1989-2005 by the Free Software Foundation, Inc.

DESCRIPTION
Bash is an sh-compatible command language interpreter that executes
commands read from the standard input or from a file. Bash also incor-
porates useful features from the Korn and C shells (ksh and csh).

Bash is intended to be a conformant implementation of the Shell and
Utilities portion of the IEEE POSIX specification (IEEE Standard
1003.1). Bash can be configured to be POSIX-conformant by default.

OPTIONS
In addition to the single-character shell options documented in the
description of the set builtin command, bash interprets the following
```
Useful Keys: man

- **F**: “Space bar” or 'f' move forward one screenful
- **B**: ‘b’ move back one screenful
- **Enter**: "Enter" or "Return" move down one line.
- **Q**: ‘q’ to quit.
Bash Features
***************

This text is a brief description of the features that are present in the Bash shell (version 3.2, 28 September 2006).


Bash contains features that appear in other popular shells, and some features that only appear in Bash. Some of the shells that Bash has borrowed concepts from are the Bourne Shell (`sh`), the Korn Shell (`ksh`), and the C-shell (`csh' and its successor, `tcsf`). The following menu breaks the features up into categories based upon which one of these other shells inspired the feature.

This manual is meant as a brief introduction to features found in Bash. The Bash manual page should be used as the definitive reference on shell behavior.
Useful Keys: info

“Space bar” to read linearly through all sections.

Back up after “Space bar”.

‘h’ special link to help.

‘l’ return from help link.

‘q’ to quit.
Useful Keys: info

- **tab**: Jump to next link or cross reference.
- **return**: Follow the link at the cursor.
- **esc**: Back up to previous link.
- **M**: ‘m’ to go to the menu.
- **L**: ‘l’ return from link.
- **Q**: ‘q’ to quit.
Getting Help: apropos

apropos bash
Files and Directories
Files

- Data in UNIX is stored in files.

Directories

- Files are organized hierarchically into directories.
- Top-level directory is called “root” represented as a slash (/).
- Every file (and directory) is listed and separated by a slash (/).
Files and Directories

The UNIX file system is like an inverted tree.

/home/its/ug1/ee51vn/report.doc
Files and Directories

List the contents of the current working directory including hidden files with \(-a\)

\(\text{ls}\)
\(\text{ls -a}\)

Create a directory called \textit{name} and change to it, \texttt{cd} by itself always changes to your home directory

\(\text{mkdir name}\)
\(\text{cd name}\)
\(\text{cd}\)

Print the path of your current working directory

\(\text{pwd}\)
Working with Directories

Try these commands:

```
$ ls
$ ls -a
.              .bash_profile .dbus     .ssh     .xauthAfjiN7
..             .bashrc       .lesshst  .vim     .Xauthority
.bash_history .bash_udit    .sav      .viminfo .zshrc
.bash_logout  .config       .smpd     .vimrc
$ mkdir Mytest
$ ls
Mytest
$ cd Mytest
$ ls
$ pwd
/home/1201/Mytest
$ cd
$ pwd
/home/1201
$ cd Mytest
$ cd
```
Working with Directories

Rename a directory called \textit{name} to \textit{newname}

\texttt{mv \textit{name} \textit{newname}}

Remove a directory called \textit{name} as long as it is empty

\texttt{rmdir \textit{name}}

Remove the directory called \textit{name} and everything in it including all files and subdirectories.

\texttt{rm -r \textit{name}}

* Note if \texttt{rm} is aliased to "\texttt{rm -i}" , then you'll be prompted to examine each file before removing. To avoid this use \texttt{\backslash rm -r \textit{name} } to override this behavior.
Working with Directories

Try these commands:

```bash
$ pwd
/home/1201/Mytest
$ mkdir Mysub1 Mysub2
$ ls
 Mysub1  Mysub2
$ mv Mysub1 Mysub3
$ ls
 Mysub2  Mysub3
$ rmdir Mysub2
$ ls
 Mysub3
$ cd
$ ls Mytest
 Mysub3
$ rmdir Mytest
rmdir: failed to remove `Mytest/': Directory not empty
$ rm -r Mytest
$ 
```
Changes the timestamp of the file called *name* or creates it:

```
touch name
```

Copy the file called *name* to *newname*:

```
cp name newname
```

Rename a file called *name* to *newname*:

```
mv name newname
```

Remove the file called *name* or interactively prompt with `-i`:

```
rm name
rm -i name
```
Working with Files

Try these commands:

```bash
$ cd
$ touch mytest
$ ls
mytest
$ cp mytest mytest.new
$ ls
mytest  mytest.new
$ mv mytest mytest.old
$ ls
mytest.new  mytest.old
$ rm mytest.old
$ ls
mytest.new
$ rm -i mytest.new
rm: remove regular empty file `mytest.new'? n
$ ls
mytest.new
$
```
Permissions
Permissions

Each file or directory has access (rwx) permissions defined for the owner (user), group and others.

- **read access** - file can be read, directory can be listed
- **write access** - file contents can be changed, directory contents can be changed (create new, delete and rename files)
- **execute access** - file can be executed, directory can be changed to (cd into that directory)
- **- access** - permission denied for that access

```
-rwxr--r--
```

`man chmod`
Try these commands:

```bash
$ mkdir Mytest
$ ls
Mytest  mytest.new
$ ls -l mytest.new
-rw-r--r-- 1 traine it_css 0 Aug 21 11:58 mytest.new
$ ls -lad Mytest/
drwxr-xr-x 2 traine it_css 6 Aug 25 08:43 Mytest/
$ 
```

The items in red tell us `mytest.new` is a file and `Mytest` is a directory. Both show permissions for user (traine) and group (it_css).
Wildcards and Home (~)
Try these commands:

```
$ ls
Mytest  mytest.new
$ cp mytest.new Mytest/mytest1.new
$ cd Mytest
$ ls
mytest1.new
$ ls ~
Mytest  mytest.new
$ cp ~/mytest.new mytest2.new
$ ls
mytest1.new  mytest2.new
$ cp *.new ~/
$ cd
$ ls
Mytest  mytest1.new  mytest2.new  mytest.new
$ rm mytest?.new
$ 
```
Wildcards and Home (~)

Try these commands:

```
$ ls
Mytest  mytest.new
$ cp -r Mytest Mytest1
$ ls
Mytest  Mytest1  mytest.new
$ rm -r Mytest*
$ ls
mytest.new
```

* `rm -r name` allows you to remove a directory and all its contents recursively. If you forget the `-r` it won't work because it is a directory.
Redirection and Pipe
Creating Files: Redirection (STDOUT)

Try these commands:

```
[train@centos ~]$ cat > animals-sm.txt
mouse
finch
hamster
[train@centos ~]$ cat > animals-lg.txt
elephant
buffalo
rhinoceros
[train@centos ~]$ more animals*
```

^D (Ctrl-d for EOF)

^D (Ctrl-d for EOF)
Creating Files: Redirection

Try these commands:

```
$ cat animals-sm.txt animals-lg.txt > animals.txt
mouse
finch
hamster
elephant
buffalo
rhinoceros
$ more animals.txt

buffalo
elephant
finch
hamster
mouse
rhinoceros
$ sort animals.txt

buffalo
elephant
finch
hamster
mouse
rhinoceros
$ sort animals.txt > animals-sorted.txt

buffalo
elephant
finch
hamster
mouse
rhinoceros
$ more animals-sorted.txt
```
Creating Files: Append and Pipe

Try these commands:

```
[train@centos ~]$ cat >> animals-lg.txt
cow
horse
[train@centos ~]$ more animals-lg.txt
elephant
buffalo
rhinoceros
cow
horse
[train@centos ~]$ cat animals-lg.txt animals-sm.txt | sort > animals-sorted.txt
[train@centos ~]$ ls
animals-lg.txt animals-sm.txt animals-sorted.txt animals.txt mytest.new
[train@centos ~]$ more animals-sorted.txt
buffalo
cow
elephant
finch
hamster
horse
mouse
rhinoceros
```

^D (Ctrl-d to stop)
Creating Files: Redirection (STDIN) and Pipe

Try these commands:

```
[train@centos ~]$ more animals.txt
mouse
finch
hamster
elephant
buffalo
rhinoceros
[train@centos ~]$ sort < animals.txt > animals-orig-sort.txt
[train@centos ~]$ more animals-orig-sort.txt
buffalo
elephant
finch
hamster
mouse
rhinoceros
[train@centos ~]$ grep e animals-orig-sort.txt
elephant
rhinoceros
horse
[train@centos ~]$ grep -c e animals-orig-sort.txt
3
[train@centos ~]$ cat animals-orig-sort.txt | wc -l
5
[train@centos ~]$ 
```
Create and Edit Files
Editors

nano
  - Quick and easy to use

vim (vi)
  - Uses modes, usually smaller and faster to load, but not as customizable

emacs
  - Highly customizable including a large number of features
The VIM editor has two modes:

- **Command**: interprets a letter or sequence of letters as a command.
- **Insert**: puts anything typed into the file. The ESC key ends insert mode and returns you to command mode.

Command line entry at the bottom of the screen appears when the command “:” is typed. VIM starts up in command mode.
## vim Basics

### Start vim

- `vi filename`
- `vi -r filename`

### End vim

- `:wq`, `:q`, `:q!`
- `ZZ`

### Moving around

- `h`, `j`, `k`, `l`
- `w`, `e`, `b`

### Insert

- `i`, `a`, `o`, `A`, `O`
- `ESC` - to return to command mode

### Delete

- `x`, `dw`, `dd`

### Undo

- `u`
The Vim(vi) tutor is useful for people that want to learn their first Vim(vi) commands.

To start the Vim(vi) tutor, type

```
$ vimtutor
```

It copies the tutor file first, so that it can be modified without changing the original file. Takes ~ 25-30 minutes to complete.
Exercises
Exercise 1

Review the Unix/Linux Tutorial for Beginners

- Use the exercises at the end of *Tutorial One* through *Five* to test your understanding of the material.

- [http://www.ee.surrey.ac.uk/Teaching/Unix/](http://www.ee.surrey.ac.uk/Teaching/Unix/)
Exercise 2

Choose an editor and complete the tutorial.

- **nano**
  [http://www.udel.edu/001401](http://www.udel.edu/001401)

- **vi (vim)**
  [http://www.jerrywang.net/vi/](http://www.jerrywang.net/vi/)

- **emacs**
Need Help?

- **Email:** consult@udel.edu
  If you make the first line of the email message
  
  Type=Cluster-Mills

  your question will be routed more quickly.

- **Phone:** (302) 831-6000
- **Text:** (302) 722-6820