INTRODUCTION TO UNIX

ADVANCED COMPUTING CENTER FOR RESEARCH AND EDUCATION

HTTP://WWW.ACCRE.VANDERBILT.EDU

What is Unix?

1. An operating system (un)like Windows; created in late 1960's at AT&T Bell Labs
2. Designed to be a programmer's operating system
3. Turned out to be a portable, multi-user, multi-tasking operating system - a first!
4. There are many different versions of Unix:
   - 4a Apple's OS X / macOS (and iOS!) is a user-friendly desktop
   - 4b Linux is a clone of Unix which offers extremely good performance and is free
   - 4c Therefore, Linux is the de facto standard for High Performance Computing (HPC) clusters

Ken Thompson (seated) and Dennis Ritchie, the creators of UNIX

All Versions of Unix Provide Some Sort of GUI, But…

ACCRE
Advanced Computing Center for Research & Education

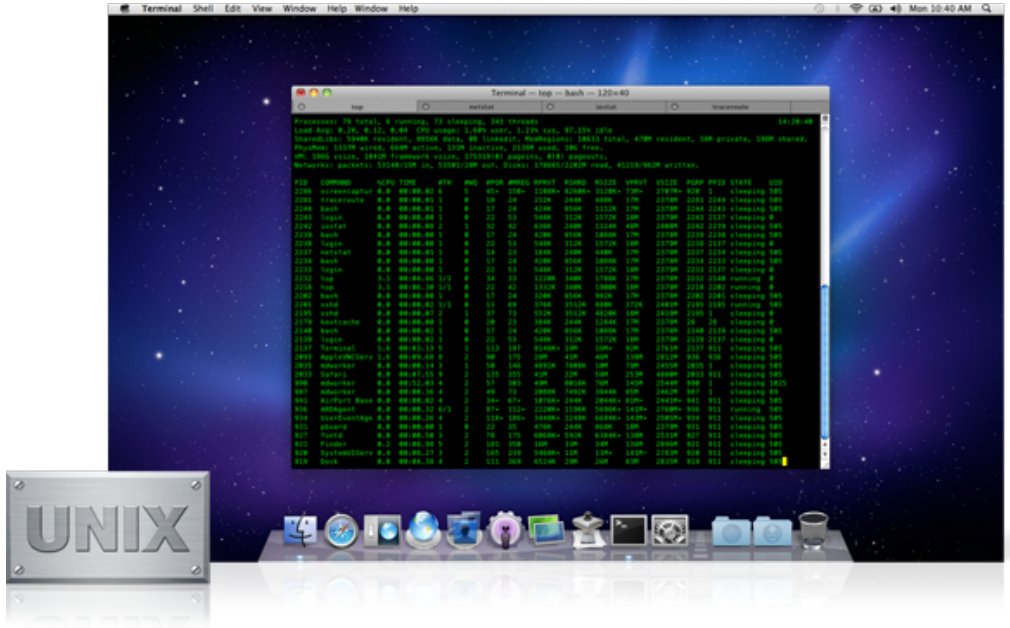
1. Users interact with the cluster via a shell

2. Yes, the command line! It's more lightweight, efficient, better suited for remote access, etc.

3. There are many different shells:

   3a. bash - most common; the default in OS X, Ubuntu Linux, and ACCRE

   3b. tcsh and zsh are examples of other shells some people prefer



The Format of Unix Commands

ACCRE
Advanced Computing Center for Research & Education

1. The format of Unix commands is:
   command [options] [arguments]

2. ls is a command

3. -l is an option to the ls command

4. example1 is an argument to the ls command

ken@vmps10:~ — ttys007

```
$ ls
example1  example2
$ ls -l
total 8
-rw-r-----. 1 ken accretraining 69 Oct 13  2008 example1
-rw-r-----. 1 ken accretraining 61 Oct 20  2008 example2
$ ls -l example1
-rw-r-----. 1 ken accretraining 69 Oct 13  2008 example1
$
```

The Most Important Command of All

ACCRE
Advanced Computing Center for Research & Education

1. The man command displays manual pages; example at left is output of man ls
2. Displays a synopsis of how to use the command and a description of each option / argument
3. Long options are preceded by two dashes
4. You cannot assume that an option does the same thing with different commands
   - 4a Example: -v means “verbose” with many commands, but it means “doesn’t match” with grep
5. command –help displays similar information

```
ken@vmps10:~ — ttys007
LS(1)                          User Commands                          LS(1)

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 -cftuvSUX nor --sort.

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

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

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

       --author
              with -l, print the author of each file

       -b, --escape
              print octal escapes for nongraphic characters

       --block-size=SIZE
              use SIZE-byte blocks.  See SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~

       -c     with -lt: sort by, and show, ctime (time of last modification
              of file status information) with -l: show ctime and  sort  by
              name otherwise: sort by ctime

       -C     list entries by columns

       --color[=WHEN]
              colorize  the  output.   WHEN  defaults to ‘always’ or can be
              ‘never’ or ‘auto’.  More info below

       -d, --directory
              list directory entries instead of contents, and do not deref-
              erence symbolic links

       -D, --dired
              generate output designed for Emacs’ dired mode

       -f     do not sort, enable -aU, disable -ls --color
:
```

Command History and Editing

1. The shell maintains a history of the commands you have previously executed (1,000 on ACCRE)
2. Up and down arrow keys scroll thru your history; left and right arrow keys move thru a command
3. Edits can be made by inserting or deleting text; pressing enter executes the command

Press the up arrow key once

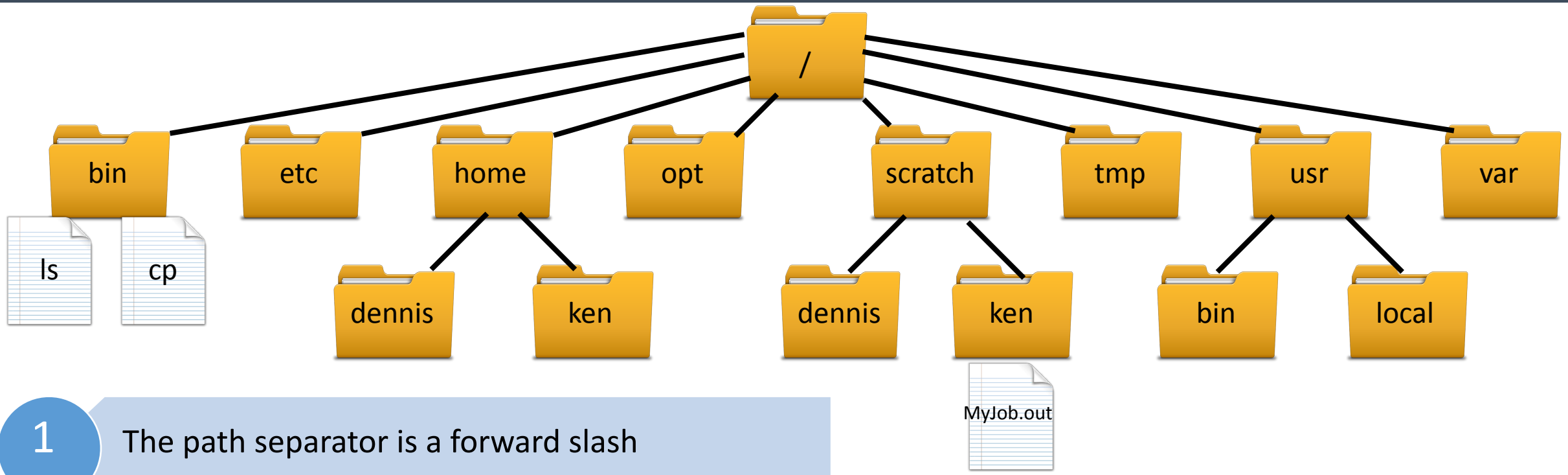
```
ken@vmps10:~ — ttys005
$ ls -l esample1
ls: cannot access esample1: No such file or directory
$ ls -l esample1
$ ls -l example1
-rw-r-----. 1 ken accretraining 69 Aug 19 15:48 example1
$
```

Use the left arrow key to move from here to here

Press the delete key to erase the “s” and type an “x”

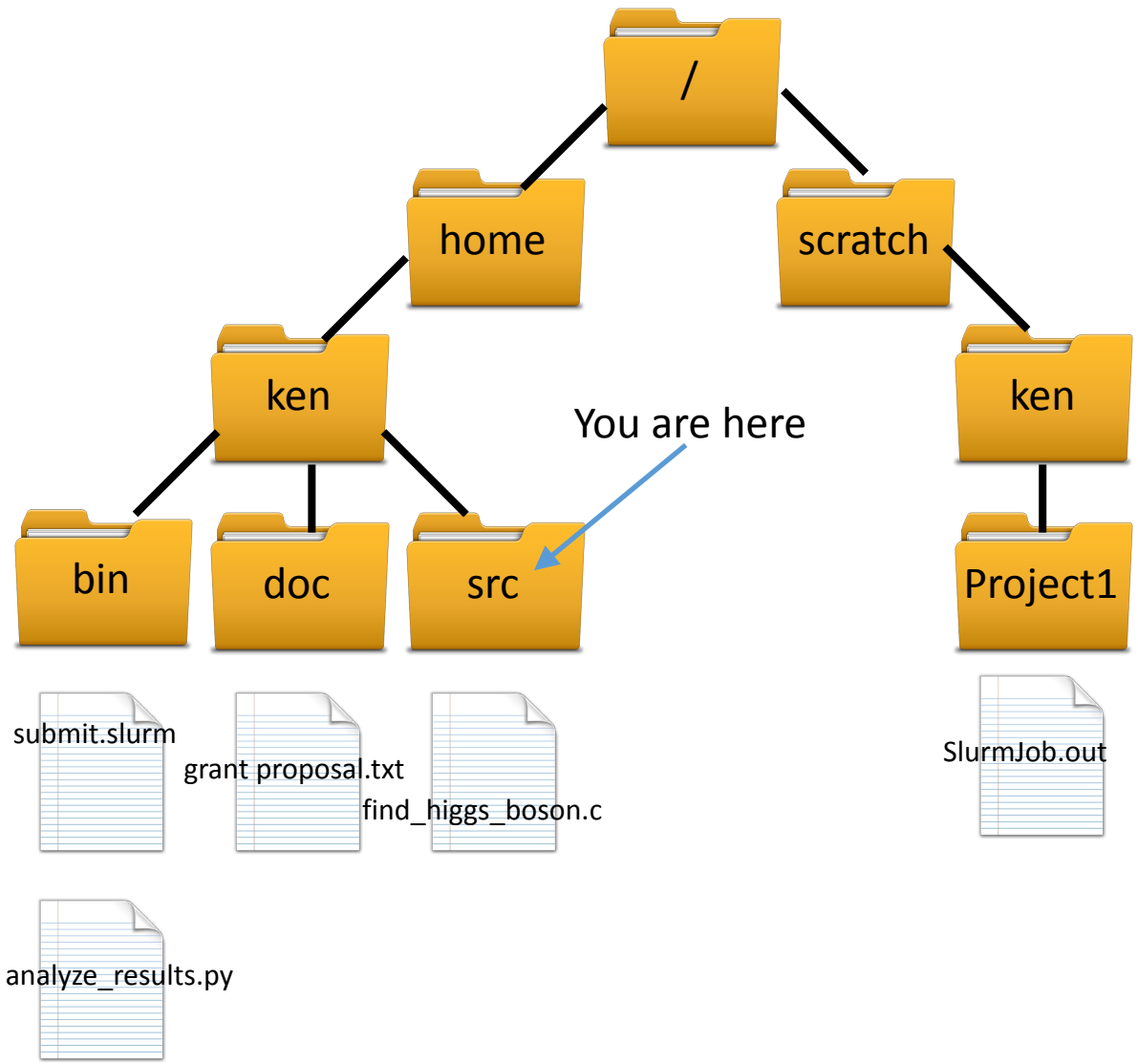
Press the return / enter key to execute the command (you don’t have to scroll to the end of the command line)

Hierarchical Filesystem



1. The path separator is a forward slash
2. There is no “C:” or “F:” drive - it’s all one big filesystem anchored at the root directory (“/“)
3. File names are case sensitive
4. File extensions, if they exist, have no meaning to the operating system

Absolute Versus Relative Paths



1. Assuming my current directory is /home/ken/src
2. The absolute path to grantproposal.txt is: /home/ken/doc/grantproposal.txt
3. The relative path to grantproposal.txt is: ../doc/grantproposal.txt
4. The absolute path to SlurmJob.out is: /scratch/ken/Project1/SlurmJob.out
5. The relative path to SlurmJob.out is: ../../../scratch/ken/Project1/SlurmJob.out
6. You should use whichever one is shorter … or easier for you to remember!

Commands for Working with Directories

ACCRE
Advanced Computing Center for Research & Education

1. pwd prints your present working directory
2. ls lists directories and files
3. cd changes directories
4. mkdir makes a directory
5. rmdir removes a (empty) directory

5a. rm -r recursively deletes a directory tree

5b. Be very careful using it!!!

ken@vmps10:~ — ttys007

```
$ pwd
/home/ken
$ ls -l
total 8
-rw-r-----. 1 ken accretraining 69 Oct 13  2008 example1
-rw-r-----. 1 ken accretraining 61 Oct 20  2008 example2
$ mkdir scripts
$ ls -l
total 8
-rw-r-----. 1 ken accretraining  69 Oct 13  2008 example1
-rw-r-----. 1 ken accretraining  61 Oct 20  2008 example2
drwxr-xr-x. 2 ken accretraining 512 Aug 10 16:12 scripts
$ cd scripts
$ pwd
/home/ken/scripts
$ ls -l
total 0
$ cd
$ pwd
/home/ken
$ rmdir scripts
$ ls -l
total 8
-rw-r-----. 1 ken accretraining 69 Oct 13  2008 example1
-rw-r-----. 1 ken accretraining 61 Oct 20  2008 example2
$
```

Commands for Working with Files

1. cat, more, or less display the contents of a file
2. cp copies files
3. mv moves (renames) files
4. rm removes files
5. The -i option makes cp, mv, and rm "interactive"

ken@vmps12:~ — ttys005

```
$ ls
example1  example2
$ more example1
This is a file called example1 for the "Introduction to Unix" class.
$ more example2
And this is another file called example2 for the same class.
$ cp example1 example3
$ ls
example1  example2  example3
$ more example3
This is a file called example1 for the "Introduction to Unix" class.
$ cp example2 example3
$ more example3
And this is another file called example2 for the same class.
$ cp -i example1 example3
cp: overwrite `example3'? y
$ more example3
This is a file called example1 for the "Introduction to Unix" class.
$ mv example3 example4
$ ls
example1  example2  example4
$ rm example4
$ ls
example1  example2
$
```

Autocompleting Filenames with the Tab Key

1. You only have to type enough of a filename to ensure uniqueness and then you can <TAB>
2. If you haven't typed enough to uniquely identify the file, press <TAB> twice for a list
3. Type enough additional to uniquely identify the file and then press <TAB> to complete!

Press TAB here

Press TAB twice here

Press TAB here

ken@vmps12:~ — ttys005

```
$ ls -l
total 8
-rw-r-----. 1 ken accretraining 69 Oct 13  2008 example1
-rw-r-----. 1 ken accretraining 61 Oct 20  2008 example2
-rw-r--r--. 1 ken accretraining  0 Aug 18 15:06 job1fordennis.slurm
-rw-r--r--. 1 ken accretraining  0 Aug 18 15:06 job2fordennis.slurm
-rw-r--r--. 1 ken accretraining  0 Aug 18 15:06 job3bfordennis.slurm
-rw-r--r--. 1 ken accretraining  0 Aug 18 15:06 job3fordennis.slurm
$ ls -l job1fordennis.slurm
-rw-r--r--. 1 ken accretraining 0 Aug 18 15:06 job1fordennis.slurm
$ ls -l job3
job3bfordennis.slurm  job3fordennis.slurm
$ ls -l job3
job3bfordennis.slurm  job3fordennis.slurm
$ ls -l job3bfordennis.slurm
-rw-r--r--. 1 ken accretraining 0 Aug 18 15:06 job3bfordennis.slurm
$
```

Editing Files

1. There are 3 editors commonly available: emacs, nano, and vim
2. emacs is very popular with programmers
3. vim has the steepest learning curve, but is the fastest of the three
4. nano is the easiest to learn and is good for basic editing - nano .bashrc
5. The bottom two lines of the screen are reserved for nano
6. The arrow keys let you move around in the file, as does Control-Y and Control-V
7. You can easily insert and delete text
8. Control-O outputs (saves) a file; Control-X exits nano

nano .bashrc

```
ken@vmps12:~ — ttys005
  GNU nano 2.0.9                File: .bashrc

# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and variables
alias ll="ls -la"
alias mroe=more

export PS1="$ "                  # Redefine shell prompt
export PATH=$PATH:/usr/lpp/mmfs/bin

                              [ Read 13 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
```

File Permissions

1. All files have one owner (user) and one group associated with them
2. Only the user may change the user, group, or permissions
3. Permissions are read, write, and execute; they apply to the user, group, and others

User ken has read, write, and execute permission

Group members have read and execute, but not write permission

Others have no permission

ken@vmps12:~ — ttys005

```
$ ls -la
total 668
drwxr-x---.    2 ken  accretraining   2048 Aug 19 14:49 .
drwxr-xr-x. 5475 root root          262144 Aug 19 08:25 ..
-rw-------.    1 ken  accretraining  11168 Aug 18 18:41 .bash_history
-rw-r--r--.    1 ken  accretraining     24 Oct 13  2008 .bash_logout
-rw-r--r--.    1 ken  accretraining    176 Oct 13  2008 .bash_profile
-rw-r--r--.    1 ken  accretraining    237 Jan 14  2015 .bashrc
-rw-r-----.    1 ken  accretraining     69 Oct 13  2008 example1
-rw-r-----.    1 ken  accretraining     61 Oct 20  2008 example2
-rw-r--r--.    1 ken  accretraining      0 Aug 18 15:06 job1fordennis.slurm
-rw-r--r--.    1 ken  accretraining      0 Aug 18 15:06 job2fordennis.slurm
-rw-r--r--.    1 ken  accretraining      0 Aug 18 15:06 job3bfordennis.slurm
-rw-r--r--.    1 ken  accretraining      0 Aug 18 15:06 job3fordennis.slurm
-rw-------.    1 ken  accretraining   3779 Jan 14  2015 .viminfo
$
```

Read, Write, and Execute

	Read	Write	Execute
Files	You can look at the file contents	You can modify the file contents	You can run the program
Directories	You can ls the directory	You can create new files, rename existing files, and <u>delete</u> files	You can cd to the directory

Changing Permissions

1. Use the chmod (change mode) command

2. Alphabetic method - add or take away (r)ead, (w)rite, e(x)ecute from (u)ser, (g)roup, (o)ther

3. Numeric method - read = 4, write = 2, execute = 1; total up for user, group, and other

ken@vmps12:~ — ttys005

```
$ ls -l example*
-rw-r--r--. 1 ken accretraining 69 Oct 13  2008 example1
-rw-r--r--. 1 ken accretraining 61 Oct 20  2008 example2
$ chmod ug+x,o-r example1
$ chmod 750 example2
$ ls -l example*
-rwxr-x---. 1 ken accretraining 69 Oct 13  2008 example1
-rwxr-x---. 1 ken accretraining 61 Oct 20  2008 example2
$
```

Pattern Matching

ACCRE
Advanced Computing Center for Research & Education

- `*` — Matches zero or more instances of any character
- `?` — Matches one instance of any character
- `[abc]` — Matches any one character within the brackets
- `[0-9]` — Matches any one character within the range defined in the brackets
- `[A-z]` — Matches all letters, plus most punctuation characters, because this is an ASCII range

ken@vmps12:~ — ttys005

```
$ ls
Example  example1  example12  example1a  example2  example3  exampleA
$ ls example*
example1  example12  example1a  example2  example3  exampleA
$ ls example?
example1  example2  example3  exampleA
$ ls example[12]
example1  example2
$ ls example[0-9]
example1  example2  example3
$ ls example[0-9]?
example12  example1a
$ ls example[0-9][0-9]
example12
$ ls example[A-Za-z]
exampleA
$
```

Finding Files and Folders with Find

ACCRE
Advanced Computing Center for Research & Education

1. Syntax is: find “where to start looking” -“criteria” “what to look for” -“what to do with it”

2. Find all files starting at the current directory whose names are example followed by another character

3. Same as the 1st example, but instead of printing their filenames, ls them

```
ken@vmps10:~ — ttys005
$ ls
Example  example1  example12  example1a  example2  example3  planets.sh
$ find . -name "example?" -print
./example1
./example3
./example2
$ find . -name "example?" -ls
1624059    4 -rw-r-----   1 ken      accretraining       69 Aug 19 15:48 ./example1
17536674    0 -rw-r-----   1 ken      accretraining      260 Aug 23 15:30 ./example3
1624062    4 -rw-r-----   1 ken      accretraining       61 Aug 19 15:48 ./example2
$ find . -mtime -3 -ls
1686964    4 drwxr-x---   2 ken      accretraining     2048 Aug 23 15:54 .
1596647   28 -rw-------   1 ken      accretraining    12801 Aug 23 18:40 ./.bash_history
17536671    8 -rw-------   1 ken      accretraining     3906 Aug 23 15:54 ./.viminfo
17536674    0 -rw-r-----   1 ken      accretraining      260 Aug 23 15:30 ./example3
6392867   24 -rw-r--r--   1 ken      accretraining    12288 Aug 22 16:01 ./.bashrc.swp
17536676    0 -rwxr-x---   1 ken      accretraining      241 Aug 23 15:54 ./planets.sh
$ ls /scratch/ken
job1.err  job1.out  job2.err  job2.out
$ find /scratch/ken -name "*.err" -print
/scratch/ken/job2.err
/scratch/ken/job1.err
$ find /scratch/ken -name "*.err" -exec rm {} \;
$ ls /scratch/ken
job1.out  job2.out
$ find . -name "example?" -size +100c -ls
17536674    0 -rw-r-----   1 ken      accretraining      260 Aug 23 15:30 ./example3
$
```

4. Find all files starting at the current directory whose modification time is less than 3 days ago

5. Find all files starting at /scratch/ken whose name ends in “.err” and print them

6. Same as the previous example, but instead of ls’ing them, rm them!

7. Two criteria used: 1) file name, 2) file size (all files larger than 100 characters in this example)

Aliases

ACCRE
Advanced Computing Center for Research & Education

1 alias name="some value"

2 alias rm="rm -i"

2 Now when you type rm, the shell will automatically replace it with rm -i

4 alias ll="ls -la"

5 alias mroe=more

6 alias login="ssh ken@login.accre.vanderbilt.edu"

Shell Variables

1. Many variables are set for you; env lists them
2. System variables are in all CAPITAL letters
3. export my_var=some_value - creates a variable
4. The echo command can be used to display the value of a variable
5. When referencing a variable, precede the name with a $ sign

ken@vmps12:~ — ttys005

```
$ export claim2fame="Inventor of UNIX"
$ echo claim2fame
claim2fame
$ echo $claim2fame
Inventor of UNIX
$ echo $PATH
/usr/scheduler/slurm/sbin:/usr/scheduler/slurm/bin:/usr/lpp/mmfs/bin:/usr/local/
bin:/usr/local/common/bin:/usr/bin:/bin:/usr/scheduler/slurm/sbin:/usr/scheduler
/slurm/bin:/usr/lpp/mmfs/bin:/usr/local/bin:/usr/local/common/bin:/usr/bin:/bin:
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/var/cfengine/bin:/
usr/lpp/mmfs/bin:/home/ken/bin
$
```

Shell Initialization Files

1. Any aliases or variables you define on the command line are in effect only until you logout

2. To make them permanent, simply add them to your .bashrc file in your home directory

ken@vmps10:~ — ttys005

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and variables
alias ll="ls -la"
alias mroe=more

export PS1="$ "                 # Redefine shell prompt
export PATH=$PATH:/usr/lpp/mmfs/bin
export claim2fame="Inventor of UNIX"
~
~
~
~
~
~
~
~
                                                14,36         All
```

Command Substitution

1. Any command enclosed in grave accents is executed first and its' output substituted in

1a. $(command) does the same thing as `command`

2. This can be used with other commands or to assign a value to a variable

Not single quotes!

ken@vmps09:~ — ttys005

```
$ echo "Today's date and time is `date`"
Today's date and time is Tue Aug 23 15:09:11 CDT 2016
$ export right_now=`date`
$ echo $right_now
Tue Aug 23 15:09:53 CDT 2016
$
```

Input / Output Redirection

ACCRE
Advanced Computing Center for Research & Education

- Any shell has 3 filehandles open by default

0 stdin - standard input, defaults to keyboard, file descriptor 0

1 stdout - standard output, defaults to screen, file descriptor 1

2 stderr - standard error, defaults to screen, file descriptor 2

Input / Output Redirection

ACCRE
Advanced Computing Center for Research & Education

1. Input redirection - e-mail yourself a file: `mailx ken.thompson@att.com < example1`

2. Output redirection - `myprogram > output.log`

3. Error redirection - `myprogram 2> error.log`

4. Output and error redirection to different files - `myprogram > output.log 2> error.log`

5. Output and error redirection to the same file - `myprogram > combined.log 2>&1`

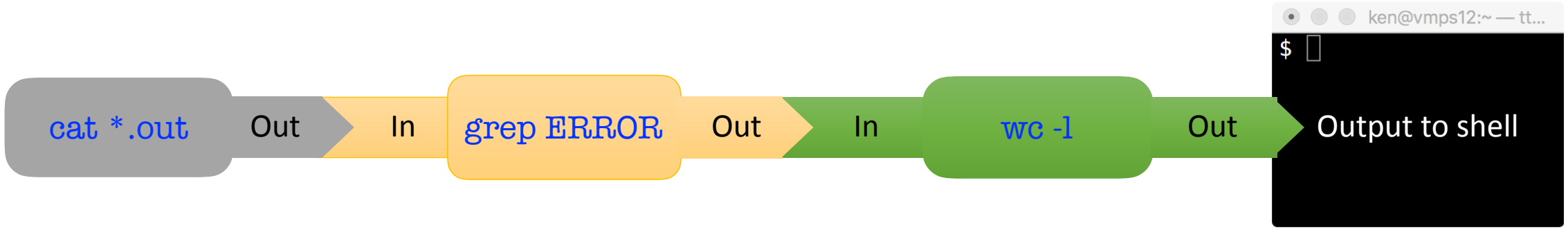
6. But don't forget the Unix philosophy!

ken@vmps09:~ — ttys005

```
$ ls example?
example1  example2  example3
$ more example?
::::::::::::::
example1
::::::::::::::
This is a file called example1 for the "Introduction to Unix" class.
::::::::::::::
example2
::::::::::::::
And this is another file called example2 for the same class.
::::::::::::::
example3
::::::::::::::
And this is yet a 3rd file for the "Introduction to Unix" class.
$ cat example1 example2 > example3
$ more example3
This is a file called example1 for the "Introduction to Unix" class.
And this is another file called example2 for the same class.
$ cat example1 example2 >> example3
$ more example3
This is a file called example1 for the "Introduction to Unix" class.
And this is another file called example2 for the same class.
This is a file called example1 for the "Introduction to Unix" class.
And this is another file called example2 for the same class.
$
```

Pipes and Filters

1. Pipes take the output of one command and make it the input to another command
2. Analogous to plumbing pipes
3. Filters are commands which can accept input from another command and also produce output
4. Syntax is `command | filter_command`
5. Multiple pipes and filters can be strung together:
   `cat *.out | grep ERROR | wc -l`



Some Useful Filter Commands

1. wc - word count
2. grep - get a regular expression and print it
3. sort - very powerful sort utility
4. uniq - filter duplicate lines
5. cut - cuts specific fields or columns
6. sed - stream editor, does search and replace

ken@vmps09:~ — ttys005

```
$ cat example3
This is a file called example1 for the "Introduction to Unix" class.
And this is another file called example2 for the same class.
This is a file called example1 for the "Introduction to Unix" class.
And this is another file called example2 for the same class.
$ cat example3 | wc
       4      46     260
$ cat example3 | grep Unix
This is a file called example1 for the "Introduction to Unix" class.
This is a file called example1 for the "Introduction to Unix" class.
$ cat example3 | sort
And this is another file called example2 for the same class.
And this is another file called example2 for the same class.
This is a file called example1 for the "Introduction to Unix" class.
This is a file called example1 for the "Introduction to Unix" class.
$ cat example3 | sort | uniq
And this is another file called example2 for the same class.
This is a file called example1 for the "Introduction to Unix" class.
$ cat example3 | sed "s/class/tutorial/g"
This is a file called example1 for the "Introduction to Unix" tutorial.
And this is another file called example2 for the same tutorial.
This is a file called example1 for the "Introduction to Unix" tutorial.
And this is another file called example2 for the same tutorial.
$
```

QUESTIONS?

ACCRE
Advanced Computing Center for Research & Education

Or are you ready to go forth and compute?!?