

CS 344 – UNIX OS

Fundamentals – Lecture #7

Purushotham Bangalore
Department of Computer and Information
Sciences
University of Alabama at Birmingham (UAB)

Declaring and Using Variables in Shell

- Declaring variables: `variable_name=value`
- Display variables: `echo $variable_name`
- Examples
 - `a=Hello`
 - `b=date`
 - `$b`
 - `c=`date``
 - `d="You are currently logged on to `hostname`"`
 - `e="Today's date is: $b"`
 - `f="Today's date is: $c"`
 - `echo $e $f`

Overwriting Existing Files

- When output is redirected to a file, if file already exists, it is overwritten
- To avoid overwriting existing files set the noclobber variable: set `–o noclobber`
- When noclobber variable is set, the shell complains that the file exists
- To overwrite a file when noclobber variable is set, use `>!` instead of `>` for output redirection
- Example: `date >! filename`

3/14/2005

3

Avoiding Accidental Removal of Files

- Unlike Windows, when files are deleted, they cannot be undeleted
- noclobber feature is an instruction to the shell not commands such as `cp`, `mv`, `rm`
- To avoid accidental removal of files use `–i` option with `cp`, `mv`, `rm`
- To avoid using `–i` option everytime set alias for `cp`, `mv`, `rm`, etc
 - `alias rm='rm –i'`
 - `alias cp='cp –i'`
 - `alias mv='mv –i'`
- To override an alias use `\command` instead of `command` (e.g., `\rm filename`)

3/14/2005

4

Redirecting Error Messages

- We have seen output redirection using “>” and “>>”, to redirect error messages use “2>” or “2>>” (in bash)
- Utilities write to error stream when an error occurs during the execution of the utility
 - ls -l filename (if the filename does not exist or file permissions are not sufficient)
- The shell assigns 1 for standard output stream and 2 for standard error stream (“>” is same as “1>”)
- To redirect both error and output together use “2>&1” after the filename
 - ls -l myfile xyz > outerr 2>&1

3/14/2005

5

Communicative Shell

- When multiple commands or special characters are used to instruct the shell, the expansion and modification made by the shell can be viewed by starting the shell with `-x` option
- We can start a new shell by typing “`bash -x`” or execute a script “`bash -x myscript`” from current shell

```
$ bash -x
$ tr '\t' ':' < quizscores | sort
+ tr '\t' :
+ sort
.....
.....
```

```
$ bash -x
$ c="You are currently logged on to `hostname`"
++ hostname
+ c=You are currently logged on to blazer5
$ echo $c
+ echo You are currently logged on to blazer5
You are currently logged on to blazer5
```

3/14/2005

6

Shell Command Line Expansion, I

- Wildcard character – *
 - Matches zero or more characters
 - List all files that end with .java – ls *.java
 - List all files that start with cs – ls cs*
- Matching character – ?
 - Matches one character
 - List all files that start with chapter followed by any character – ls chapter?
- * and ? are special characters called meta-characters interpreted by the shell, these characters should not be used in filenames

3/14/2005

7

Shell Command Line Expansion, II

- To specify filename within a specified range use []
 - List all files that start with chapter and has a number following it within the range 1-5 – ls chapter[1-5]
 - Only one character is matched
- To match and create multiple filenames from one pattern use {}
 - List all files that start with chapter and has specific numbers following it – ls chapter{2,3,4}
 - Curly braces match files specified in the braces, but will NOT expand ranges
- All meta-characters can be combined together
 - ls *[0-9]?{java,c,cpp}

3/14/2005

8

Local & Environment Variables

- Use “export variable_name”
 - a=Hello
 - export a
 - export b=World
 - export PATH=\${PATH}:~/bin
- Remove variable – unset
 - unset a
- List all variables – set
- List environment variables – env

3/14/2005

9

Filename Completion with Shell

- When listing or editing files typing complete filenames accurately could be difficult
- The shell can be used to aid this problem
 - Using wildcard characters
 - Using file completion with TAB
- Enter part of the filename/directory and press TAB
- If a unique file/directory exists the shell will complete it, otherwise it will display all possible options

3/14/2005

10

Job Control Commands

- When shell executes a command it waits until the utility completes
- If we do not like to wait, we can instruct the shell to execute the job in background by adding & at the end of the command (e.g., sleep 10 &)
- To suspend current job, press CTRL-Z
- To kill a running job with interrupt signal, press CTRL-C
- To kill a running job with quit signal, CTRL-\
- To list all jobs background and stopped jobs use the command “jobs”
- To put suspended job into background use “bg %job_num”
- To bring a background job into foreground use “fg %job_num”
- To kill a job use “kill %job_num”

3/14/2005

11

Details about User Processes

- To list user processes use “ps” command (on some systems you can also use “top”)
- To obtain detailed information about all processes use “ps -ef”
- To list processes owned by you use “ps -u \$USER”
- To obtain a long listing of your processes use “ps -l”

```
$ ps -l
F S  UID  PID  PPID  C PRI NI  ADDR  SZ  WCHAN TTY  TIME CMD
8 S  511 22817 22815 0 51 20      ?  309      ? pts/1  0:00 bash
```

- To kill a particular process use “kill -9 PID”

3/14/2005

12