CS-344 - Unix Operating System Fundamentals

Lecture 5
Using Multiple Utilities in Scripts
and
Accessing and Changing Previous Commands

Based on slides created by Dr. Bangalore for the Spring 2005 offering of the course

Shell Scripts (I)

- Enables execution of complex tasks by using multiple commands in a single file
- *.bashrc* or *.bash_profile* are such examples
- Create simple script using any editor

```
echo "Welcome" $USER
echo "Today's date is: "
date | cut -d' ' -f2-3
echo "You are logged in to: "
hostname
echo "There are"
who | wc –l
echo "user(s) currently logged in"
echo "Your PATH is set to the following directories:"
echo $PATH | tr ':' '\n'
```

Shell Scripts (II)

- To execute a script:
  - The script must have execute permission
  - File permissions can be set using "*chmod*"
- or
  - Use *source script_name*

Creating a complex script

- Read a file – myfile.in
- Output to the screen the total number of unique words
- Output the list of unique words to the file *words.out* along with the number of times each word appears ordered with the most-used words listed first
- Solution:

```
tr -d '?."!:,;' < myfile.in | tr 'A-Z' 'a-z' | tr ' \t' '\n\n' \
| sed '/^$/d' \
| sort | uniq -c | sort -rn \
| tee words.out | wc -l
```


Algorithm

- Delete punctuation characters
- Convert all characters to lowercase
- Move each word to a separate line
- Remove blank lines (if any)
- Sort the lines
- Remove duplicates
- Compute word frequency and output to file
- Compute the total number of unique words

Implementation (I)

- Delete punctuation characters
  tr –d '?."!:,();' < file
- Convert all characters to lowercase
  tr 'A-Z' 'a-z' < file
- Move each word to a separate line
  - replace space and tab with a new line
    tr ' \t' '\n\n' < file
- Remove blank lines (if any)
  sed '/^$/d' file
  - search for lines starting with ^ and ending $ with to text in between, and then delete those lines

Implementation (II)

- Sort the lines
  sort file
- Remove duplicates and compute word frequency
  uniq –c file
- Sort based on word frequency
  sort –rn file (-n numerical sort, -r reverse sort order)
- Output to file and another utility
  | *tee words.out*
- Compute the total number of unique words
  wc -l

Solution

```
tr -d '?."!:,;' < myfile.in \
| tr 'A-Z' 'a-z' \
| tr ' \t' '\n\n' \
| sed '/^$/d' \
| sort | uniq -c \
| sort -rn \
| tee words.out | wc -l
```

continuation character – no new line

Executing Shell Scripts

```
$ cat myscript
echo "Welcome" $USER
echo "Today's date is: " `date | cut -d' ' -f2-3`
echo "You are logged in to: " `hostname`
echo "There are" `who | wc -l` "user(s) currently logged in"
echo "Your PATH is set to the following directories:"
echo $PATH
$ ls -l myscript
-rwxr-xr-x   1 puri     staff       239 Feb 20 18:23 myscript
$ myscript
bash: myscript: command not found
$ ./myscript
Welcome puri
Today's date is:  Feb 20
You are logged in to:  hestia
There are 1 user(s) currently logged in
Your PATH is set to the following directories:
/usr/bin:/bin:/usr/sbin:/sbin:/hf/local/bin:/mz/mb/puri/bin
$ export PATH=$PATH:.
$ myscript
```

Shell did not find the script in it's PATH

You can also specify the full path

You will see the above output this time

Accessing & Changing Previous Commands

history (I)

- Displays a list of previous commands executed by the shell
- Each command has an associated number
- To repeat the last command enter *!!*
- To execute a command by event number use *!number* (e.g., *!19* executes the command associated with event # 19)
- To execute a command that begins with a specific string use *!string* (e.g., !ca will execute the last command starting with ca)

history (II)

- To add history event number to command prompt enter export *PS1='[\!] $ '*
- To select all arguments from previous command use *!** as argument of new command

```
cat quizscores homework
wc !*
```

- To select the last argument of previous command use *!$* as argument of new command
- To add an argument to a previous command use *!! NewArgument* or *!string NewArgument*

```
!! –l
!cat hw5
```


Filename Completion with Shell

- When listing or editing files typing complete filenames accurately could be difficult
- The shell can help in this problem:
  - Using wildcard characters
  - Using file completion with *TAB* (BASH)
- 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