

CS-344 - Unix Operating System Fundamentals

Lecture 3
Using Basic UNIX Utilities
and
Using Multiple Utilities in Scripts

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

Few points about the Shell (I)

- Determine which shell you currently have
 - One option, use "ps" command
- How to determine where the command or utility is present?
 - Use "which" command

```
$ ps
PID TTY TIME CMD
2665 pts/2 0:00 bash
```

```
$ which bash
/usr/bin/bash
```

Department of Computer and Information Sciences UAB

Few points about the Shell (II)

- ❑ How does the shell know that a command is in a particular directory?
 - Using an environment variable – *PATH*
- ❑ How to determine what *PATH* variable is set to?


```
$ echo $PATH
/usr/bin:/bin:/usr/sbin:/sbin:/hf/local/bin
```
- ❑ What other environment variables are there?
 - Use "*env*" (also "*printenv*" on Linux) to display all the environment variables and their corresponding values

6/16/2005 4

Department of Computer and Information Sciences UAB

Few points about the Shell (III)

- ❑ How to change one of the environment variable?
 - For bash, type: *PS1='someprompt'*

```
puri@blazer1:~[501]$ PS1='[hostname]:[date] '$ '
[blazer1:Sun Feb 13 19:14:06 CST 2005] $
```
- In general, type: *variable=value*

```
$ echo $PATH
/usr/bin:/bin:/usr/sbin:/sbin:/hf/local/bin
$ PATH=$PATH:$HOME/bin
$ echo $PATH
/usr/bin:/bin:/usr/sbin:/sbin:/hf/local/bin:/mz/mb/puri/bin
```
- ❑ How to make these changes be persistent?
 - Include these changes in the file *.bash_profile* or *.bashrc* (for bash shell)

6/16/2005 5

Department of Computer and Information Sciences UAB

cut

- ❑ To cut out fields from each line of a file
- ❑ Each field must be separated by a delimiter
 - default delimiter is a tab
 - other delimiters use *-d<character>* (*-d' '* or *-d:*)
- ❑ Other options include:
 - By fields: *cut -f1,4 myfile*
 - By character position: *cut -c5-10 myfile*
- ❑ What do you see when you type:
 - *cut -f5,6 -d: /etc/passwd*
 - *ls -l | cut -c55-*

6/16/2005 6

paste

- Puts data together by concatenating corresponding lines in specified files
- A tab character is added before the corresponding line from the 2nd file
- To change the default field separator use *-d* option (*paste -d':' file1 file1*)
- Paste can also be used to combine multiple lines in a single file using *-s* option (*paste -s -d'' myfile*)
- Multiple files can be specified as arguments for the paste utility (*paste file1 file2 file3 > newfile*)

6/16/2005

7

diff

- Compares two files and displays changes required to the first file to make it match the second file
- Lines unique to each file are marked between '`<`' and '`>`' characters
- No output is produced when the two files are identical
- Usage: *diff <options> file1 file2*
- Other useful options:
 - Ignore case: *-i*
 - Ignore all blanks: *-w*

6/16/2005

8

grep

(global regular expressions print)

- To search for a pattern in a file
- Usage: *grep <options> <pattern> <filename>*
- Each line with the matching pattern is displayed on the console by default
- To display lines not matching the specified pattern use *-v* option
- Specify pattern with in single quotes when using non-alpha numerical characters
- Other useful options:
 - Ignore case during comparison: *-i*
 - Display line number for matching lines: *-n*
 - Display only the filenames that contain the pattern: *-l* (useful when searching multiple files)

6/16/2005

9

sort (I)

- Sort all input lines based on specified order (default in ASCII order)
- Usage: `sort <options> <filename>`
- To sort:
 - Dictionary order: `sort -d myfile`
 - Ignore case: `sort -f myfile`
 - Numerical value: `sort -n myfile`
 - Reverse sort: `sort -r myfile`
- Output sent to standard output by default, use `-o` option to send output to a specified file
 - `sort myfile.in -o myfile.out`

6/16/2005

10

sort (II)

- To sort using data after specific fields use:
`sort -k n file`
- To sort using a specific field range use:
`sort +n -m file` (start after n delimiters and stop after m delimiters)
- To specify a secondary key for sorting use:
`sort +n -m +p -q file` (field range n-m for primary key, p-q for secondary key)
- A different sort order can be specified for the secondary key (see pages 265-266)
- Useful when sorting data based on a specific field and breaking ties with a secondary key

6/16/2005

11

join

- To combine two files based on a common field (input files must be sorted)
Usage: `join <options> file1 file2`
- By default all fields from both files are sent to the standard output after the common field
- To display only specific fields from each file the `-o` option can be used:
`join -o 2.2 1.2 1.1 file1 file2`
- By default field 1 is used as the common key, a different field can be specified using `-j` option:
`join -j1 n -j2 m file1 file2`

6/16/2005

12

Department of Computer and Information Sciences UAB

sed

(stream editor)

- Stream editor – works on individual lines instead of reading the entire file
- Useful when working on large files or making changes from a script file
- Sample usage scenarios (output sent to standard output):
 - Replace all instances of a specific string:
`sed 's/abc/ABC/g' myfile`
 - Search specific string and then make a replacement:
`sed '/xyz/s/abc/ABC/g' myfile`
 - Delete lines:
`sed '/abc/d' myfile`

6/16/2005 13

Department of Computer and Information Sciences UAB

tr

- Reads standard input, deletes or translate characters based on the input options, and displays output to standard output
- Usage: `tr <options> string1 string2`
- Examples:
 - `tr a A < myfile`
 - `tr abc XYZ < myfile` or `cat myfile | tr abc XYZ`
 - `tr -d 'xyz' < myfile`
 - `cat /etc/passwd | tr ':' ''`
 - `tr '\n' '' < myfile`

6/16/2005 14

Department of Computer and Information Sciences UAB

tee

- Note that output redirection with `>` and `|` we can send output to either a file or a utility not both
- `'tee'` is used to send output to both a file and another utility (similar to a plumber's tee)
- The data is not modified in anyway by tee utility
- Example:
`ls -l | tee myfile | wc -l`

6/16/2005 15

Miscellaneous

- ❑ Calculator
 - bc – uses a C language like syntax
 - dc – uses reverse polish notation
- ❑ 'file' utility can be used to determine the file type

```
$ bc
scale=10
12.0+2.1
14.1
$ bc -l
12.0+2.1
14.1
```

```
$ dc
12.0
2.1
+
$ bc -l
p
14.1
$
```

```
$ file /etc/motd
/etc/motd:  ascii text
$ file /bin/bash
/bin/bash:  ELF 32-bit MSB executable SPARC Version 1, dynamically
linked, stripped
$ file t.c
t.c:        c program text
```

6/16/2005

16

uniq

- ❑ Uniq discards a duplicate line if adjacent lines in the input are same
- ❑ Duplicate lines in the input will not be detected if they are not adjacent
- ❑ If input is sorted only one copy of all duplicate and unique lines will be displayed
- ❑ Options:
 - To output unique lines only use *-u*
 - To output lines that have duplicates use *-d*
 - To output number of times a line is duplicated use *-c*
 - To ignore first n fields on each input line when comparing use *-f n*

6/16/2005

17

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
```

6/16/2005

18

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`

6/16/2005

19

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 '\n' '\n\n' \
| sed '/^$/d' \
| sort | uniq -c | sort -rn \
| tee words.out | wc -l
```

6/16/2005

20

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

6/16/2005

21

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

6/16/2005

22

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`

6/16/2005

23

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

6/16/2005

24
