Scribe notes for Lecture 11 - 2/8/2016

Perpared by Aditya Naik, Nnamdi Nnamdi-Emetarom, Mengyu Mo and Shilin Ni

Overview

In lecture 10, we covered svn, UNIX basics, what is shell scripting, environment variables and shell expansion. Lecture 11 was a continuation of that discussion and went over advanced ideas including text processing, redirection, piping, alias, grep, gawk, associative arrays, and regular expressions.

We explored the following uses of bash scripting:

- Word Processing
- Redirection
- Piping
- Text Processing
- Grep
- Shell scripts

We then covered Gawk and gave a brief overview of shell scripts.

1. Word Processing

   Sort

   Sort is a simple and very useful command which will rearrange the lines in a text file so that they are sorted, numerically and alphabetically. By default, the rules for sorting are:

   Lines starting with a number will appear before lines starting with a letter;

   Lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet;

   Lines starting with a lowercase letter will appear before lines starting with the same letter in uppercase.

However, the rules for sorting can be changed according to the command provided. For example:

```
$ sort -r -u <filename>
$ sort -n -k 2 -t <filename>
```

Flags:

-r
Reverse the comparison results

-u
Delete duplicate lines

-c
Check for strict ordering

-n
"Numeric Sort": compare according to numeric string value

-k
Select key value

Count

Count can be used for counting

```
$ wc -l -w -m -c <text>
```

Flags:

-l
counts the number of lines

-w
counts the number of words

-m
count the number of characters

-c
count the number of bytes

These commands, combined with the idea of piping, provide programmers with a very powerful tool.

uniq

uniq reports or filters out repeated lines in a file. However, uniq does not detect repeated lines unless they are adjacent.

```
uniq -c <filename>
```

Flags:

-c
Prefix lines with a number representing how many times they occurred.

2. Redirection

   Redirection can be used in the command line to redirect keyboard input to standard input stream using " < "

   Redirection can also be used to redirect a program's standard output or standard error output using " > "

3. Piping

   Piping can be used to use the output of one process as the input of another process.

   Piping implements shared memory between the program - a part of the virtual memory allocated to each process is reserved to be shared with another process.

   Example:

   ```
   $ process1 | process2 | process3
   $ ls -al | more
   ```

   .

4. Text Processing

   translate

   ```
   $ tr [options] <set1> <set2>
   ```

   If sets are strings of characters, by default, translate searches through the string in set1 and replaces them with set2.
   Example:

   ```
   $ echo <filename> | tr AEIOU aeiou
   ```

   This replaces the all uppercase vowel with lower case. Remember from previous lecture, the () symbol indicated no special characters are evaluated.
   Flags:

   -d

   Delete all characters that are in the set

-c

Complements set1 before replacing it with set2. This is commonly used as a combination of both c and d as

```
$ tr -cd [a-zA-Z] <filename>
```

which removes all non-letters from the file.

Translate and redirect:

Note: Because bash processes I/O from left to write, the commands can be chained:

```
$tr -cd 0-9 < file1 > file2
```

which deletes everything but numbers from file1 and stores the resulting output in file2

5. Grep

grep, which stands for global regular expression print, processes text line by line and prints any lines which match a specified pattern.

```
$ grep [OPTIONS] PATTERN [FILE...]
```

Flags:

-i

Perform a case-insensitive match.

-v

Inverts the match

-o

Stands for –only-matching, print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.

-n

Stands for –line-number, prefix each line of output with the 1-based line number within its input file.

Examples:

```
$ grep [Mm]onster Frankenstein.txt | wc -l:
```

Count lines contain the word monster or Monster in file Frankenstein.txt

in [Mm]onster means the first character in a word must be M or m.

```
$ grep <c[A-Za-z]*d\>
$ grep <c.*d>
$ gerp -o \<c...d>
```

[a-zA-Z]*: * stands for [a-zA-Z] appearance 0 or more times.
. stands for single character.

6. Gawk

Awk is a type of programming language that is designed for text processing, including sort, counting, uniq, associative arrays and regular expression. It is one of the most powerful tools for unix operation systems. Gawk, therefore, stands for the GNU implementation of awk.

Awk has a pattern-action structure - every time a line of input is read by the processor and the associated action of any pattern that matches the input line will be executed.

Format:

```
Pattern1 {action1}
Pattern2 {action2}
Pattern3 {action3}
```

Advantages of gawk:

- complicate text processing tasks can be done with a few instructions.
- access fields within lines easily
- own sophisticated and numerous built-in functions such as print, string and arithmetic.
- have variables and control flow in the actions

Examples:

```
$ gawk /[Ss]tudent/ {print} hw1.txt
$ gawk /[Ss]tudent/ hw1.txt
$ gawk /[Ss]tudent/ /print $0} hw1.txt
```

Gawk defaults to print the whole line if no action is specified. Therefore, the above three lines are equivalent - searching and printing Student or student when one is found in the file hw1.txt. Also, Extended regular expressions like ? and ˆare accepted.

Begin and End The actions of these two patterns, just like their names, are only executed once before and after the whole process of inputting and matching.

Example:

```
$ gawk BEGIN {print "Starting search for a student}
/[Ss]tudent/ {print; count++}
END {print "Search completed, there are
  " count " students in the book.} hw1.txt
```

Input fields:

- Each input line can be automatically separated into several fields via field separator (FS).
- FS defaults to whitespace. It can be changed by using begin pattern.
- Each field is referred to by a number.

More Gawk features:

- If no pattern is given, the action will be executed by default for every input line
- NF - number of fields in the current line
- NR - number of lines read so far
- The above line will the print all the words in infile.
- -The for loop searches for each field in the current line and prints that field. -There is no pattern so the code is executed for each line.
- RS - record separator, defaults to newline
- gawk -F - the same as FS=
- toupper(), tolower() - convert lowercase/uppercase to uppercase/lowercase
- gawk - matching command
- ! gawk - not matching command

Associative Arrays Arrays in awk is quite different from arrays in other programming languages.

- No need to specify array size
- Index can be numbers or strings. In other words, it extends the choices of index.
- Format: arrayname[index]=value
- Remove an element: delete array[index]

7. Shell Scripts

A script is a text file that an interpreter can read and execute. Bash is a shell and an interpreter. A simple bash script is a text file with lines of commands.

One of the most important uses of shell scripts is automation of tasks. Shell scripts also tend to be cleaner and simplistic than other programming languages.

Example: helloworld.sh

```bash
#! /bin/bash
echo "Hello, my name is $USER"
echo "Greetings to the world from my $SHELL shell"
echo "on $HOSTNAME which is a machine of type $MACHTYPE."
```

Use chmod +x “file”.sh to make the shell script executable
To run the script, use ./“file”.sh

LaTeX