EECS 2031

Bash

Topics

- Functions
- User Interface

Special Substitution

- Functions are mainly to reuse code.
- Instead of repeating few lines of codes many times, put them in a function.
- A function must be defined in the script before it is used.
- Function with the same name as a command ?

no arguments

```
function_name() {
     <commands>
    <commands>
    }
```

```
function function_name() {
     <commands>
    <commands>
    }
```

Function

```
1. #!/bin/bash
2. function say_hello() {
3.     echo Hello World
4. }
5. say_hello   #no parenthesis
6. say_hello
```

```
tigger 184 % f1.sh
Hello World
Hello World
tigger 185 %
```

Passing Parameters

```
1. #!/bin/bash
2. function say_something() {
3.     echo $1
4. }
5. say_something  good
6. say_something very bad
```

```
tigger 184 % f2.sh
good
very
tigger 185 %
```

return value

- typically, bash functions do not return any values.
- However they return status (integers)
- Typically a return value of 0 means success (no errors), nonzero could be an error code
- One way to overcome this limitation is to use cat in the function and command substitution in the caller x=$(function)

Variable scope

- variables are global (in the script)

```
#!/bin/bash
#The scope of a variable
function do_var() {
    echo function x=$x
    x=22
}
x=1
echo main: before $x
do_var
echo main: After  $x
```

```
tigger 184 % f3.sh
main: before 1
function x=1
main: After 22
tigger 185 %
```

Variable scope --local

- variables are global (in the script)

```
#!/bin/bash
#The scope of a variable
function do_var() {
    echo function x=$x
    local x=22
}
x=1
echo main: before $x
do_var
echo main: After  $x
```

```
tigger 184 % f3.sh
main: before 1
function x=1
main: After 1
tigger 185 %
```

Sed -- Introduction

- Sed is a stream editor
- Line by line goes through the editor (filter) where every line may or may not change
- There is an interactive editor *ed* that accepts the same commands
- All editing commands (could be in a script file) are applied to each line in the file.
- The output is sent to the standard output (may be redirectred to a file).

How sed works

- Every line of the input file is read into the “pattern space”
- Sed commands are applied to he line one by one.
- After all the commands are applied to the line, the line is sent to the output (some of these commands may result in discarding the line).
- Each command is on the form of address and action
- The address decides if the action will be applied to the line or not.
- If 2 commands are applied at he same line, the second command will be applied to the “possibly”
- modified line by the first command

Sed Commands

- The address can be either a line number or a pattern enclosed between two slashes */pattern/*
- If no pattern, the command is applied to every line
- if one address, the command applied to that line, if 2 addresses, the command applied to the range of addresses.
- take a look at man sed, here are few useful flags
- -n Suppress automatic printing of pattern space
- -e script to follow
- -f script file

Addresses -- Example

- d Delete all th elines
- 2d Delete line 2
- 1,4d Delete lines 1 through 4
- /^$/d Delete all blank lines
- 7,/^$/d Delete lines 7 through the first blank line
- /^$/,$d Delete from the first blank line to the last line
- /a*b/,/[0-9]$/d Delete from the line that contains b, ab, aab, ... to the first line that ends with a digit

Sed Commands

- a\ Append one or more line to the current line
- c\ Change current line with new text
- d Delete line
- h Copy pattern space to holding buffer
- H Append content of pattern space to holding buffer
- g move holding buffer to pattern space (overwrite)
- G like g but append
- p print line
- s substitute
- n,q,r,!

Delete Command

- sed '3d' file delete the 3rd line
- sed '$d' file delete the last line
- sed '/north/d' file delete all lines that contains north

Substitute Command

- sed ‘s/west/north/’ file replace the first occurrence of west in every line to north
- sed ‘s/west/north/g’ replace each occurrence of west by north in each line.
- sed –n ‘s/west/north/p’ print only line that contains the word after replacing it by north
- sed –n ‘s/west/north/gp’ print only line that contains the word after replacing it by north but replace every occurrence (g for globally)
- sed –n ‘s/\(Mar\)got/\1ianne/p’ What is that?
- Can have multiple commands sed –e ‘---’ –e ‘-----’ file

Reading and Writing

- sed '/James/r newfile' file Looks for lines that contains James and right after it, sed read and includes the contents of "newfile"
- sed –e '/James/p' –e '/james/r newfile' file
- sed –e '/james/w newfile' file it writes the lines that contain James into new file

Changing the file

- Appending a line after a specific line
- sed –n '/north/ a <---Moved-------->' file It will append the string "<---Moved--->" after each line that contains "north
- sed –n '/north/ a\

  > <---Moved-------->' Another way to do it
- If you want north followed by white space /north[[:space:]] or north[\t]
- Use i\ instead of a\ to insert before the line
- sed '/western/c\

> changed' file change the line contains western to "changed"

Other commands

- sed '/east/{n; s/aa/bb/;} datafile the n commands matches the patter following it to the next line not the current one
- the y command is similar to Unix tr
- sed '1,3y/abcdef/ABCDEF/' datafile Capitalize letters a-f in the first three lines