Lecture 7

Introduction to Shell Scripts

COP 3353 Introduction to UNIX

What is a shell script?

An executable file containing

- Unix shell commands
- Programming control constructs (if, then, while, until, case, for, break, continue, while,)
- basic programming capabilities (assignments, variables, arguments, expressions,)

The file entries are the *script*

The file is interpreted rather than compiled and executed

- The first line of the script indicates which shell is used to interpret the script

Simple script (egshell.sh)

```
#!/bin/sh
#this is the script in file egshell.sh
cal
date
who | grep liu
exit
```

The #! is used to indicate that what follows is the shell used to interpret the script

The exit command immediately quits the shell script (by default it will also quit at the end of the file)

Executing shell scripts

```
sh myscript      #uses Bourne shell
tcsh myscript    #uses t-cshell
```

Note that the above explicitly invoke the appropriate shell with the file containing the commands as a parameter. The file does not need to be executable.

You can also make the file executable and then simple run as a command

```
chmod 755 myscript (or chmod +x myscript)
myscript
```

Shell scripts

Advantages

- Can quickly setup a sequence of commands to avoid a repetitive task
- Can make several programs work together

Disadvantages

- Little support for large and complicated programming semantics
- Shell scripts need to be interpreted hence are slower programs

Which shell to use?

- csh shell and tcsh shell are recommended for use at the command line
- sh (Bourne) shell and bash shell are recommended for writing shell scripts
- Examples will generally use the Bourne shell

Printing a line to standard output

Use the echo command to print a line to stdout

Form of command:

echo <zero or more values>

Examples

```
echo "Hello World"
echo "hello" "world" #two values
echo hello  #need not always use quotes
echo "please enter your name"
```

Shell Environment Variables

These are variables provided as part of the shell s operational environment

They exist at startup but can be changed

Examples are: USER, HOME, PATH, SHELL, HOSTNAME

The setenv command (in tcsh) is used to set these, for example, by:

```
setenv PATH $PATH:/home/here/bin
```

(this sets the PATH variable so that it s current value is appended by :/home/here/bin)

Note that setenv is how tcsh sets the environment variables

User defined variables

You can also specify variables yourself and these can also be used inside a script

In tcsh, the set command is used to set a variable to a string value

Form:

set <name> = <value>

Examples:

```
set alpha = ñany stringò
set beta = 3
set mypath = /home/special/public_html
```

Once a variable has been defined, it s value can be used by dereferencing it with $.

```
ls ïal $mypath
```

Note that using setenv or set without any parameters simply displays the current settings

Shell variables (Bourne shell)

Note that for all shells, variables need not be declared explicitly, but simply used

For the Bourne shell, the use is as follows (note that there should be no blanks before and after the equals sign and no need for the set command.

Form:

<name>=<value>

Example

```
alpha=ñhello worldò
beta=45
echo $alpha $beta ñthird argumentò
```

Note that $alpha is the value of the variable alpha

Reading values into shell variables

The read statement is used to read a line of standard input, split the line into fields of one or more strings, and assign those strings to shell variables. Any strings not assigned are assigned to the last variable.

Form:

read <var1> <var2> <varn>

Examples

```
read num
read field1 field2 rest
read field1 field2 < ifile.txt
```

Shell arguments

Arguments on the command line can be passed to a shell script, just as you can pass command line arguments to a program

$1, $2, , $9 are used to refer to up to nine command line arguments (similar to C s argv[1], argv[2], , argv[9]).

Note that $0 contains the name of the script (argv[0])

Example:

```
shprog.sh john 40
shprog.sh bob 45 ñnew yorkò
```

Example using shell arguments

Script:

```
#!/bin/sh
#script name is greet.sh
#friendly display of today's date
echo "Hello " $1 $2 "pleased to meet
  you "
echo "The date is "
date
exit
```

Usage:

```
greet.sh john smith
```