Shell Scripting Continued

Common Scripting languages

- sh (Bourne Shell)
- bash (Bourne Again Shell)
- csh (C Shell)
- ksh (Korn Shell)
- tcsh (Tenex C Shell)
- Other languages : Perl, Python
- cat /etc/shells
    - lists the shells available on the system
    - UNIX can have as many as 280 shells
- System boot scripts (/etc/init.d)

Variable Comparison

- Strings
  - = equal
  - != not equal
- Numerical
  - -eq equal
  - -ne not equal
  - -lt less than
  - -le less than or equal
  - -gt greater than
  - -ge greater than or equal

for Command

```
for VARIABLE in LIST OF VALUES ; do
  COMMANDS ;
done

for FILENAME in im1 im2 im3 ; do
  bet $FILENAME ${FILENAME}_brain ;
done
```

while Command

```
while CONDITION ; do
  COMMANDS ;
done

a=1
while [ $A -lt 4 ] ; do
  bet im$A brain$A ;
  a=`echo $A + 1 | bc` ;
done
```

case Command

```
case $VAR in
  OPTION1)
    COMMAND ;;
  OPTION2)
    COMMAND ;;
esac
```

Redirecting output to a file

```
# Output to a file
ls > file.txt
# Append to a file
ls >> file.txt
sort < file.txt
wc < file.txt > wordcount.txt
#  Output Error 2 to file or stdout
ls /nonexistentdir 2> error.txt
ls /nonexistentdir > output.txt 2>&1
```

Functions

- Allow for modularizing code in shell scripts
- Can be called like independent scripts

```
function NAME {
  COMMANDS
}

greet() {
  echo "Hello, $1"
}
greet "Bob"
```

Wildmasks

- Matching patterns in filenames
    - * matches any string
    - ? matches any one character
    - [abgj] matches any one character in range/list

```
$ ls
sub1_t1.nii.gz sub1_t2.nii.gz sub2_t1.nii.gz
sub2_t2.nii.gz sub3_pd.nii.gz
$ ls sub1*
sub1_t1.nii.gz sub1_t2.nii.gz
$ ls sub*t1*
sub1_t1.nii.gz sub2_t1.nii.gz
$ ls sub[13]*
sub1_t1.nii.gz sub1_t2.nii.gz sub3_pd.nii.gz
$ ls sub?_t2.nii.gz
sub1_t2.nii.gz sub2_t2.nii.gz
# Wildmasks and variables substituted before echo prints
$ echo sub*t1*    sub1_t1.nii.gz    sub2_t1.nii.gz
```

Command Line Arguments

- Variables $1 $2 $3 etc store value of command line arguments
- `./backup.sh CS211 2024-04-07.zip
  - $0 = name of the script (often including the path)
    - $1=CS211, $2=2024-04-07.zip
  - $# = number of command line arguments given
  - $@ = all the command line arguments
  - $$ = process ID number (unique to this process)

File Redirection

- Command input can be taken from a file with: <
- Command output can be redirected to a file with: >
- Command output can be appended to a file with: >>

```
$ echo "smoothing=10mm" > settings.txt
$ echo "No lowpass" >> settings.txt
$ cat settings.txt
```

```
smoothing=10mm
No lowpass
```

awk Command

- Very general pattern matching facility.
- Simple but capability to pick out columns of text

sed Command

- Performs string substitutions
- Used to add, remove or change parts of a string
- Often invaluable for modifying variables

```
sed s/STRING1/STRING2/g
```

Regular Expressions

- Very flexible and not quickly learnt
- Special characters used in regular expressions include:
  - . matches any one character
  - * matches zero or more of the last character
  - .* matches any string
  - [ ] matches any character in the range
  - ^ represents the start of the line
  - $ represents the end of the line
  - [^] matches any character not in the range

```
$ echo "Hello world" | sed 's/w.*/X/g'     Hello X
$ echo "Hello world" | sed 's/^.* /X/g'    Xworld
$ echo "Hello world" | sed 's/.$/X/g'      Hello worlX
$ echo "Hello world" | sed 's/[wo]/X/g'    HellX XXrld
$ echo "Hello world" | sed 's/[^wo]/X/g'   XXXXoXwoXXX
```

File Type Testing

Tests	Command
-b block file	-p exists and named pipe
-c character file	-r exists and readable
-d directory	-s exists and size greater than 0
-e file exists	-u exists and SUID bit set
-f exists and regular file	-w exists and writable
-g exists and SGID bit set	-0 exists and owned by effective user ID
-h exists and symbolic link	-k exists and sticky bit SUID or SGID
-x exists and executable	chmod 1644 filename

Commands	Description
basename	removes all leading directory info and specified extensions
dirname	just returns the directory path to specified file
sort	sorts files according to alphabetic or numerical order
which	reports where an executable file can be found
head	prints the first n lines of a file
tail	prints the last n lines of a file
touch	creates an empty file
paste	merges files together (horizontally)
grep	finds patterns in strings
pipe	chain commands together, each input is previous output

System Administration

- Automate backups
- User management
- Files management and access
- System monitoring

Data Processing

- Process large datasets
- Automate data manipulation tasks

Web Development

- Manage server configurations
- Automate deployment processes

Disadvantages

- Prone to costly errors
- One mistake can change the command, be harmful
- Slow execution speed, compared to compiled languages
- Design flaws within language syntax or implementation
- Not well suited for large and complex task
- Minimal data structure

Questions?