The Shell

Ned Nedialkov

McMaster University
Canada

SE 3F03
January 2015

Outline

Shells

- A Unix shell is a command-line interpreter
- Interface to Unix/Linux OS
- Bourne bash shell; default on many systems
  On my system
  ```
  vc4:~%which bash
  /bin/bash
  ```
  which tells the location of a program; searches in the directories of the `$PATH` variable
- Also C shell csh, tcsh, Korn shell ksh, ...

Shell scripts

- Text file containing shell commands
- First line specifies the shell to be used
  - `#!/bin/bash` Bash shell
  - `#!/bin/sh -x` calls `bash` in debug mode; prints each line
  - `#!/bin/csh` Csh shell

Shell variables

- `name=value`
  No space around `=`
- To access the value, use `$`, e.g. `$name`
- Examples
  - `name=/usr/lib/filename`
  - `name=`**`ls`**
    The output of **ls** is assigned to `name`
    To see the value of `name`, **echo** `$name`

Command-line arguments

- `$0` name of the script/command
- `$1` first argument
- `$2` second argument and so on
- Try

```
#!/bin/bash
echo "My name is $0"
echo "First argument is $1"
echo "Second argument is $2"
```

For loops

```
for i in list
do
#  command(s)
done
```

means comment

Try

```
#!/bin/bash
files=`ls `
for i in $files
do
 echo "Echoing␣file␣name:␣" $i
done
```

Foor loops

```
for i
do
#   command(s)
done
```

Default is the list of input arguments

Conditionals

```
if command
then
# command(s)
else
# command(s)
fi
```

- **command** is any command or command sequence
- true is when the return value is 0
- false is when $\neq 0$

- **test** evaluates a conditional expression
  - `0` if true, `1` if false
- same as `[ args ]`, `args` is an expression

```
if [ args ]
then
# command(s)
else
# command(s)
fi
```

expression	true if
`str1 = str1`	`str1` equals `str2`
`str1 != str1`	`str1` does not equal `str2`
`-r file`	`file` exists and readable
`-w file`	`file` exists and writable
`-d file`	directory
`-f file`	regular file
`-s file`	file size $>$ 0
`expr1 -a expr2`	`expr1` and `expr2` are true
`expr1 -o expr2`	`expr1` or `expr2` is true

Examples

```
#!/bin/bash
if [ $1 = "foo" ]
then
    echo "First␣argument␣is␣foo"
else
    echo First argument is not foo
fi
```

- **if** `[ -r file.txt ]` is readable
- **if** `[ "$1"= "foo"-a -r file.txt ]` if the first argument is `foo` and `file.txt` is readable

Shell customization

- `.bashrc` executes when Unix starts a new shell
- `.bashrc_profile` executes on login; `.bashrc` runs first
- When an xterm is created, `.bashrc` is executed, but not `.bashrc_profile`
- C shell
  - `.cshrc`
  - `.login`

Environment variables

- `PATH` specifies where the shell searches for commands
- **export** `PATH=$PATH:/usr/`**local**`/bin`
- **export** defines an environment variable
- `HOME` home directory
- `EDITOR` default editor
- To see all such variables, `printenv`

Aliases

- Store them in `.bash_profile`
- **alias** `newname='`**command**`'`
- Examples
  - **alias** **rm**`='`**rm** `-i'` to prompt before removing a file
  - **alias** **cp**`='`**cp** `-i'` to prompt before copying a file

Example: `showfiles` script

```
#!/usr/bin/env bash
EXPECTED_ARGS=1
E_BADARGS=1
if [ $# -ne $EXPECTED_ARGS ]
then
  echo "Usage: `basename $0` {arg}"
  exit $E_BADARGS
fi
if [ ! -e $1 ]
then
 echo "file $1 does not exist"
 exit $E_BADARGS
fi
for myfile in $1/*
do
    if [ -d "$myfile" ]
    then
      echo "$myfile (DIR)"
    elif [ -f "$myfile" ]
    then echo "$myfile"
    fi
done
```

Example: svn_clean

```
#!/bin/bash
export PATH=$PATH:/Users/ned/bin
name=`ls `;
for i in $name
do
  if [ -d $i ]
  then
  cd $i
  svn_clean
  cd ..
  fi
  if [ -f $i ]
  then
    if [[ "$i" == *~  || "$i" == *.log ||\
        "$i" == *.aux || "$i" == *.bak ||\
        "$i" == *.dvi || "$i" == *.zip ||\
        "$i" == *.toc || "$i" == *.gz* ||\
        "$i" == *.bbl || "$i" == *.blg ||\
        "$i" == *.mt* || "$i" == *.maf ||\
        "$i" == *.out  ]]
   then
         svn rm --force $i
         echo removing file $i
    fi
  fi
done
```

Quoting variables

```
#!/bin/bash
echo "What␣is␣the␣difference␣between"
words="I␣am␣here"
for word in $words; do
    echo "$word"
done
echo "and"
for word in "$words"; do
    echo "$word"
done
echo 'and'
for word in '$words'; do
    echo "$word";
done
```

For more details, see

http://tldp.org/LDP/abs/html/quotingvar.html