Advanced Shell

Zilogic Systems

1. Shell Variables

- Used to store information in memory for later use by the user or the shell.
- To set the value of variable

```
$ myvar=value
```

- Note that there is no space around the `=` sign.
- To retreive the value of the variable prefix the variable name with a `$` sign.

```
$ myvar2=$myvar1
```

- The `echo` command is used to print the string passed as arguments.
- To view the value of a variable, the `echo` command can be used.

```
$ echo $myvar
```

- TAB completion also works with shell variables, in `bash`.
- The values of the variables are stored internally as a string of characters.
- There are no integer variables.
- The variables are not preserved across re-boots.

Try Out

- Store the string `/usr/share/iceweasel/icons` in a variable `icons`
- List the contents of the directory using the variable.
- Copy the contents of the directory to `/tmp` using the variable.

2. Quoting

- Certain characters have special meaning to the shell, like `*`, `?`, `>`, `&`, `$`, etc.
- If a command contains these characters they will be specially interpreted by the shell.
- Quoting can be used to prevent the special interpretation of these characters.

Quoting a Single Character

- To remove the special meaning of single character, prefix the character by a `\`.
- Example to copy a file called `m&n`, the following command can be used.

```
$ cp m\&n.txt /tmp
```

Quoting a String of Characters

- To prevent the shell from interpreting special characters in a string of characters, surround the string by a single quote.

- The above example can be repeated with a single quote.

```
$ cp 'm&n.txt' /tmp
```

- The double quotes is similar to single quote, except that `$` retains its special meaning.
- To copy a file called `m&n-1.txt`, the following command can be used.

```
$ i=1
$ cp "m&n-$i.txt" /tmp
```

Try Out

- The directory `quoted` in the home directory contains files with special characters.
- Try copying each file to the directory `slash`, using the `\` quote.
- Try copying each file to the directory `single`, using the `'` quote.
- Try copying each file to the directory `double`, using the `"` quote.

3. Special Shell Variables

`PATH`. Specifies a list of directories in which the shell should look for commands. The directories are separated by a `:`.

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin
```

When the user types `ls` in the command line, the shell looks for `ls` in `/usr/local/bin`, `/bin` and `/usr/bin`. The first match is executed.

To add a directory to existing list of directories, the following command can be used.

```
$ PATH=$PATH:/path/to/new/directory ❶
$ PATH=/path/to/new/directory:$PATH ❷
```

❶ The directory is added to the end of the list.
❷ The directory is added to the start of the list.

Note that the directory added to `PATH` should be an absolute path.

Try Out

- Take a backup of your original `PATH` using

```
BPATH=$PATH
```

- There are two binaries `ls` and `hello` in `~/mybin`
- From some other directory invoke `hello`, the command will fail.
- Add the directory `mybin` to the end of `PATH`
- Invoke `ls` and `hello`.
- Add the directory `mybin` to the beginning of `PATH`.
- Invoke `ls` and `hello`.
- Restore your original `PATH`.

```
PATH=$BPATH
```

`PS1`. Specifies the format of the bash prompt. The prompt can be as simple as a dolar sign. To change the prompt to a dollar sign.

```
PS1="$ "
```

To change the prompt to a constant string, the string can be assigned to `PS1`.

```
PS1="hello> "
```

The following information can be included, using escape sequences.

- absolute path of current working directory, `\w`
- current working directory, `\W`
- hostname, `\h`
- username, `\u`
- date, `\d`
- time in 12-hour format, `\@`

Try Out

- Modify `PS1` so that the date, time and current working directory is displayed in the prompt

4. Regular Expressions

- Commonly called Regex (= **Reg** ular **Ex** pressions).
- Regex - string of characters and metacharacters that specifies the rules for the string to be matched
- Metacharacters - interpretations beyond their literal meaning
- Regex are used in many text processing tools like `grep`, `sed`, `emacs` and `vi`.
- It is also widely used in programming languages like `perl`, `python` and `ruby`.

4.1. Matching

- most characters match themselves - alphabets and numbers
- `.` - match any character, equivalent to `?` character in shell wild cards.

Try Out

- Find out all words in the dictionary that contain 21 characters.

  1. Character Class

- `[]` - character class
  - characters can be listed
  - or a range can be specified
  - metacharacters lose their meaning within the character class
  - complementing with `^`
- Example: to find out all words that contain an `x` or `y` followed by `ing`

```
$ cat /usr/share/dict/words | grep [xy]ing
```

- The `--color` option to grep can be used to highlight the matched portion of the line.

Try Out

- Find out all words in the dictionary that contains 4 consequent vowels.

More on Character Classes

- common sets of characters are given a special name
- named classes - `[:alpha:]`, `[:space:]`, `[:digit:]`, `[:alnum:]`, `[:lower:]`, `[:upper:]`

4.2. Repeating

- `*` - previous character can be matched zero or more times
- Ex: `ca*t` matches `ct`, `caat`, `caaat`, `caaaat`
- `\` - convert metacharacter to ordinary character
- `\` - convert ordinary character to metacharacter
- `\+` - previous character can be matched one or more times
- Ex: `ca\+t` matches `cat`, `caat`, `caaat` but not `ct`
- `\?` - previous character can be matched zero or 1 time, optional
- Ex: `ca\?t` matches `ct`, `cat`
- `\{m,n\}` - previous character must be matched atleast m times and at most n times
- Ex: `ca\{2,3\}t` - `caat`, `caaat` but not `cat` or `caaaat`
- if `m` is omitted, `0` is assumed
- if `n` is omitted, infinite is assumed

4.3. Anchors

- `^` - matches beginning of line
- `$` - matches end of line

4.4. Examples

- Find the occurence of the word "are" in the GPL license: /usr/share/common-licenses/GPL
- Find the occurence of years in the GPL license.
- Find all words in `dict` that start with X.
- Find all words in `dict` that start with X and x.
- Find all words in `dict` that contains more than one consecutive o.
- Find all words in `dict` that contains more than one consecutive vowel.
- Find all words in `dict` that does not contain vowels.
- Find all words in `dict` that contains all vowels in the same order.

5. Scripting

- Hello World

  1. Hello world script

```
#!/bin/sh ❶
```

```
echo "Hello World" ❷
```

❶ Sha bang sequence
❷ Shell command

- Making it executable

```
chmod +x hello.sh
```

- Executing a shell script

```
$ ./hello.sh
```

Try Out

- Create a Hello World shell script.

Beyond Hello World

1. A script to create a backup of a directory

Backup script.

```
#!/bin/sh
                        ❶
zipfile="/tmp/backup-$(date +%d%m%y).zip" ❷
backdir="/home/vijaykumar/projects/smash"

zip -r $zipfile $backdir ❸
```

❶ The `$()` performs command substitution. The command is replaced by the commands output.
❷ The `date` command displays the current date in the specified format.
❸ The `zip` command is used create a ZIP file. The `-r` option specifies that directories are to be zipped recursively.

Looping Constructs

- The `for` statement executes the loop body once for each item in the list specified after `in`.
- The body for the `for` loop is enclosed between `do` and `done`.

`for` loop sample.

```
for x in 1 2 3
do
echo Hello $x
done
```

Sample code output.

```
Hello 1
Hello 2
Hello 3
```

A script to convert all .bmp files to .jpg files

Convert BMP files to JPEG files.

```
#!/bin/sh

for file in *.bmp
do
  file=$(basename $file .bmp) # ❶
  echo "Converting $file.bmp to $file.jpg ..."
  convert $file.bmp $file.jpg # ❷
done
```

❶ `basename` command removes displays filename without extension. The filename is the first argument and extension is the second argument.
❷ `convert` command can be used to convert between file types.

6. Further Reading

- Bash Programming - Introduction HOWTO http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html
- An Introduction to the Unix Shell by Steve Bourne. The author of the original bourne shell. http://partmaps.org/era/unix/shell.html