【Linux Programming】Day8(2)

Tags	
Date	@May 28, 2022
Summary	The . command, eval, exec, and echo

【Ch2】Shell Programming

2.5 Commands(2)

2.5.4 The . Command

The dot command executes the command in the current shell:

```
. ./shell_script
```

Normally, when a script executes an external command or script, a new enviornment(a subshell) is created, the command is executed in the new enviornment. The new enviornment is then discarded apart from the exit code.

However, the external source and the dot command run the commands listed in a script in the same shell that called the script.

If we have two separate scripts:

```
#!/bin/sh

# File named classic_set.sh
version=classic
```

```
#!/bin/sh

# File named latest_set.sh
version=latest
```

Then, if we execute

```
$ . ./classic_set.sh
$ echo $version
classic
$ . ./latest_set.sh
$ echo $version
latest
```

2.5.5 echo

To suppress the newline character, we co do

```
echo -n "string to output"
```

But we can also do

```
echo -e "strign to output\c"
```

The second option, `echo -e`, ensures that the interpretation of backslashed excape characters, such as `\c` for suppressing a newline, \t for outputting a tab and `\n` for outputting carriage return, is enabled.

2.5.6 eval

The `eval` command enables us to evaluate arguments.

```
foo=10
x=foo
y=’$’$x
echo $y
```

This gives us `$foo`. However

```
foo=10
x=foo
```

```
eval y=’$’$x
echo $y
```

This gives us 10.

2.5.7 exec

The `exec` command has two different uses. Its typical use is to replace the current shell with a different program. For example,

```
exec wall "Thanks for all the fish"
```

in a script will replace the current shell with the `wall` command. No lines inthe script after the exec will be processed, because the shell executing the script no longer exists

The second use is to modify the current file descriptor

```
exsec 3< afile
```

This causes file descriptor three to be opened for reading from file afile.