【Linux Programming】Day9

☰ Tags	
🗓 Date	@May 29, 2022
☰ Summary	exit, export, shift, and expr

【Ch2】Shell Programming

2.5 Commands(3)

2.5.8 exit n

The `exit` command causes the script to exit with exit code n. If we allow our script to exit without specifying an exit status, then the status of the last command is used as the return value.

In shell programming, exit code 0 is success, and codes 1 through 125 are error codes.

Exit Code	Description
126	The file was not executable
127	A command was not found
128 and above	A signal occurred

Here is a simple example that returns success if a file called .profile exists:

```
#!/bin/sh

if [ -f .profile ]; then
  exit 0
fi

exit 1
```

2.5.9 export

The `export` command makes the variable named as its parameter available in subshells.

First, list `export2` :

```
#!/bin/sh

echo "$foo"
echo "$bar"
```

Now for `export1` , invoke `export2` :

```
#!/bin/sh

foo="The first variable"
bar="The second variable"

export2
```

If we run these, we get the following:

```
$ ./export1

The second variable
```

The `foo` variable was lost because it wasn't exported.

2.5.10 expr

The `expr` command evaluates its arguments as an expression. It's most commonly used for simple arithmetic in the following form:

```
x=`expr $x + 1`
```

The ` `(backtick) characters make x take the result of executing the command ~~expr $x + 1~~.

2.5.11 set

The `set` command sets the parameter variables for the shell.
Suppose we want to use the name of the current month in a shell script

```
#!/bin/sh

echo the date is $(date)
set $(date)
echo The month is $2

exit 0
```

The program sets the parameter list to the date command's output and then uses the positional parameter `$2` to get at the month.

2.5.12 shift

The `shift` command moves all the parameter variables down by one, so that $2 becomes $1, $3 becomes $2, and so on.

shift is often useful to scan through all the parameters.

```
#!/bin/sh

while [ "$1" != "" ]; do
  echo $1
  shift
done

exit 0
```