【Linux Programming】Day6

Tags	
Date	@May 20, 2022
Summary	Control Structure: while, until, and case

【Ch2】Shell Programming

2.4.6 Control Structures(2)

while

When we need to repeat a sequence of commands, but don't know in advance how many times they should execute, we will normally use a while loop.

```
while condition
do
  statements
done
```

For example, here is a rather poor password-checking program:

```
#!/bin/sh

echo "Enter password"
read input

while [ $input != "secret" ]
do
  echo "Wrong password. Try again."
  read input
done

exit 0
```

until

The `until` statement has the following syntax:

```
until condition
do
  statements
done
```

The `until` loop continues until the condition becomes true, not while the condition is true.

Note: In general, if a loop should always execute at least once, use a while loop; if it may not need to execute at all, use an until loop.

As an example, we can set up an alarm that is initiated when another user, whose login name we pass on the command line, logs on.

```
#!/bin/sh

until who | grep "$1" > /dev/null
do
  sleep 60
done

echo "User $1 logged in"

exit 0
```

case

The case construct is a little more complex than it is in other languages. Its syntax is as follows:

```
case variable in
  pattern [ | pattern] ...) statements;;
  pattern [ | pattern] ...) statements;;
esac
```

Note: Each pattern line is terminated with double semicolons(;;). We can put multiple statements between each pattern and the next, so a double semicolon is needed to mark where one statement ends and the next pattern begins.

Be careful with the case construct if we are using wildcards such as ‘*’ in the pattern. The problem is that the first matching pattern will be taken, even if a later pattern matches more exactly.

We can write a more selective version of the input-testing script

```
#!/bin/sh

echo "Is it morning or afternoon?"
read input

case $input in
  yes) echo "Good morning";;
  no) echo "Good afternoon";;
  y) echo "Good morning";;
  n) echo "Good afternoon";;
  *) echo "Wrong input";;
esac

exit 0
```

To make it shorter: