【Linux Programming】Day5(2)

Tags	
Date	@May 19, 2022
Summary	If, elif, for

【Ch2】Shell Programming

2.4.6 Control Structures

If

The `if` statement is very simple: It tests the result of a command and then conditionally executes a group of statements:

```
if condition
then
  statements
else
  statements
fi
```

A common use for if is to ask a question then make a decision based on the asnwer:

```
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeOfDay

if [$timeOfDay -eq "yes"]
then
  echo "Good morning"
else
  echo "Good afternoon"
fi

exit 0
```

elif

We can use elif to add another condition

```
elif [ $timeOfDay = "no" ]
then
  echo "Good afternoon
else
  echo "Sorry, $timeOfDay not recognized"
  exit 1
fi

exit 0
```

A Problem with Variables

Try this new script, but just press Enter without answering the question. We'll see the following error:

```
[: =: unary operator expected
```

The problem is in the first if clause. When the variable `timeOfDay` was tested, it consisted of a blank string. Therefore, the if clause looks like:

```
if [ = "yes" ]
```

which isn't a valid condition. To avoid this, we must use quotes around the variable:

```
if [ "$timeOfDay" = "yes" ]
```

An empty variable then gives the valid test:

```
if [ "" = "yes" ]
```

If we want to suppress the suppress the new line, we can use

```
echo -n "Is it morning? Please answer yes or no"
```

for

Use the for construct to loop through a range of values, which can be any set of strings.

The syntax is simple:

```
for variable in values
do
  statements
done
```

Example:

```
#!/bin/sh

for foo in bar fud 43
do
  echo $foo
done
exit 0
```

Suppose we want to print all the scripts files that start with an `f` and end with `.sh`