Shell Scripting 3

for Command

Description:

The for command is used to execute a set of commands a specified number of times, iterating over a list of items.

Example:

```
for i in 1 2 3
do
echo $i
done
```

Output:

```
1 2 3
```

Note: you can use the for loop without putting the args like the past example , you can pass them when you execute the file.

Example:

```
echo Number of arguments passed is $#
for arg
do
echo $arg
done
```

Make the script executable and run it as follows: ./fname 1 2

while Command

Description:

The while command executes a set of commands repeatedly as long as a specified condition is true.

Example:

```
i=1
while [ "$i" -le 5 ]
do
echo $i
i=$((i + 1))
done
```

Output:

1 2 3 4 5

until Command

Description:

The until command continues execution as long as a specified condition is false. Once the condition becomes true, the loop terminates.

Example:

```
if [ "$#" -ne 1 ]
then
echo "Usage: mon user"
exit 1
```

```
fi
user="$1"
until who | grep "^$user " > /dev/null
do
sleep 60
done
echo "$user has logged on"
```

Output:

[user] has logged on

break Command

Description:

The break command is used to exit a loop immediately, terminating the loop execution prematurely.

Example:

```
while true
do
cmd="$1"
if [ "$cmd" = quit ]
then
break
else
echo "$cmd"
sleep 1
```

```
fi
done
```

Output:

```
 [command1]
[command2]
...
```

continue Command

Description:

The continue command skips the remaining commands in a loop iteration and proceeds with the next iteration of the loop.

Example:

```
for file
do
if [ ! -e "$file" ]
then
echo "$file not found!"
continue
fi
echo $file
done
```

Output:

[file1]

[file2]

getopts Command

Description:

The getopts command is used to parse command-line options in shell scripts. It is executed inside a loop and examines each command line argument, determining whether it is a valid option according to the specified options.

Example 1:

```
#!/bin/sh

while getopts "abc:" flag

do

echo "$flag" $OPTIND $OPTARG

done
```

Output:

```
./getopts1 -abc "foo"

a 1

b 1

c 3 foo
```

Example 2:

```
#!/bin/sh
while getopts "abc:def:ghi" flag
do
echo "$flag" $OPTIND $OPTARG
done
```

Output:

```
./getopts2 -a -bc foo -f "foo bar" -h -gde
a 2
b 2
c 4 foo
f 6 foo bar
h 7
g 7
d 7
e 8
```

read Command

Description:

The read command is used to read input from standard input. It allows shell scripts to be interactive with users, reading user input and assigning it to specified variables.

Example:

```
#!/bin/sh
echo "$tofile already exists; overwrite (yes/no)?"
read answer
```

Output:

```
[tofile] already exists; overwrite (yes/no)?
```

printf Command

Description:

The printf command is used to print formatted output in shell scripts. It allows for more control over the output format compared to echo.

Example:

```
printf "This is a number: %d\n" 10
```

Output:

```
This is a number: 10
```

Character	Use for Printing
d	Integers
u	Unsigned integers
o	Octal integers
x	Hexadecimal integers, using a-f
X	Hexadecimal integers, using A-F
c	Single characters
s	Literal strings
b	Strings containing backslash escape characters
%	Percent signs

سبحان الله والحمد لله ولا إله إلا الله، والله أكبر.