CSCI 274 - Intro to Linux OS

Week 6 - Pattern-centric Control and Loop Structures

Mona Wade (dwade@mines.edu)

Overview

1. Pattern-centric Control
2. Looping Statements
    a. for
    b. while
    c. until
    d. break
    e. continue

Pattern-centric Control

Pattern-centric Control (aka case statement) is similar to switch statement in C. It can be used to test simple values like integers and characters. Case statement is not a loop, it doesn't execute a block of code for any number of times. Instead, it checks the condition and controls the flow of the program.The case statement is good alternative to **multilevel if-then-else-fi** statement. It enable you to match several values against one variable.

```
case expression in
    pattern1 )
        statements ;;
    pattern2 )
        statements ;;
    ...
esac
```

Looping Statements

The purpose of loops is to repeat the same, or similar, code a number of times. This number of times could be specified to a certain number, or the number of times could be dictated by a certain condition being met. There are **4** looping statements that can be used in bash programming:

1. while statement
2. for statement
3. select statement
4. until statement

while

Command is evaluated and based on the result loop will executed, if command raise to false then loop will be terminated.

```
while command
do
   Statement to be executed
done
```

Infinite for while can be created with empty expressions.

for

This loop operates on lists of items. It repeats a set of commands for every item in a list.

```
for var in word1 word2 ...wordn
do
   Statement to be executed
done
```

Latest bash version 3.0+ has inbuilt support for setting up ranges.

Infinite for loop can be created with empty expressions.

until

The **until** loop is almost equal to the while loop, except that the code is executed while the control expression evaluates to false. The condition is evaluated before executing the commands.

```
until [CONDITION]
do
  [COMMANDS]
done
```

The **while** and **until** loops are similar to each other. The main difference is that the **while** loop iterates as long as the condition evaluates to **true** and the **until** loop iterates as long as the condition evaluates to **false**.

select

The **select** mechanism allows you to create a simple menu system. A few points to note:

- No error checking is done.
- If the user hits enter without entering any data then the list of options will be displayed again.

```
select var in <list>
do
    <commands>
done
```

break

You can do early exit with the **break** statement inside the while loop. This is done when you don't know in advance how many times the loop will have to execute.

```
while [ condition ]
do
   statements1      #Executed as long as condition is true and/or, up to a disc
   statements2
  if (disaster-condition)
  then
        break              #Abandon the while lopp.
  fi
  statements3          #While good and, no disaster-condition.
done
```

continue

To resume the next iteration of an enclosed while loop, use the **continue** statement. When used in a for loop, the controlling variable takes on the value of the next element in the list.

```
while [ condition ]
do
  statements1      #Executed as long as condition is true and/or, up to a disa
  statements2
  if (condition)
  then
        continue   #Go to next iteration of I in the loop and skip statements3
  fi
  statements3
done
```