ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 12 LESSON

IF – ELIF – ELSE STATEMENTS
FOR LOOPS (CONTINUED) / WHILE LOOPS
COMMAND SUBSTITUTION / START-UP FILES

LESSON TOPICS

Additional Control Flow Statements

- **if-else** / **if-elif-else**
- **for** loop (continued) / Using **Command Substitution**
- **while** loop

Additional Features / Start-up Files

- Purpose
- **/etc/profile , ~/.bash_profile , ~/.bashrc , ~/.bash_logout**

Perform Week 12 Tutorial

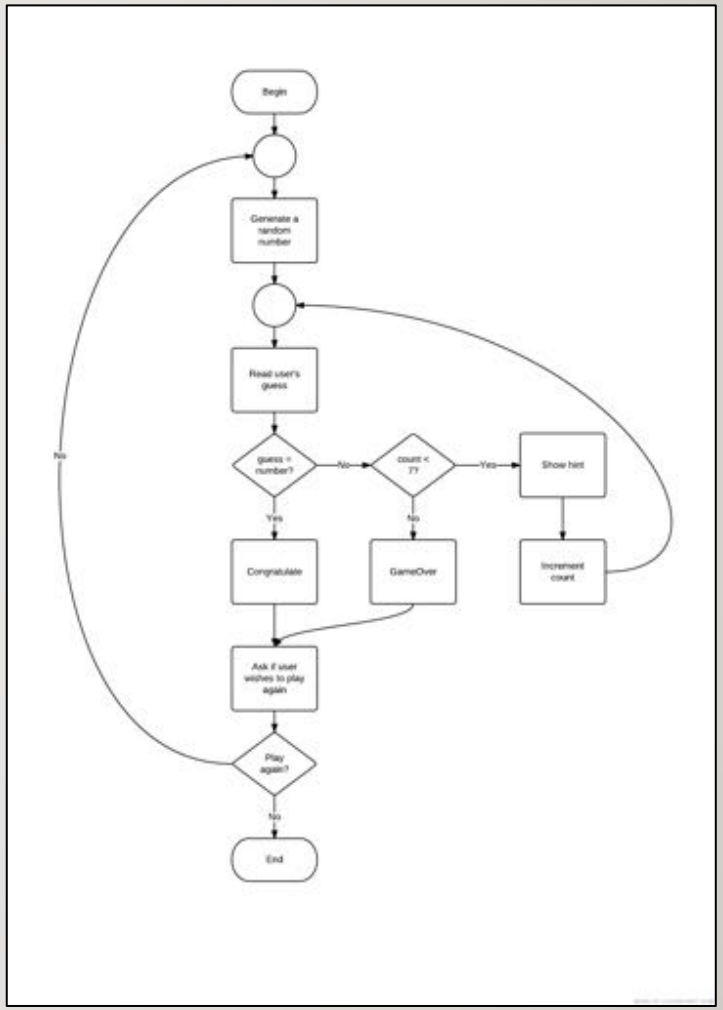
- Investigations 1,2 & 3
- Review Questions (Questions **1 - 8**)

ADDITIONAL CONTROL FLOW STATEMENTS

As discusses in a previous lesson, we can use **control flow statements** that will control the sequence of a running script based on various situations or conditions.

Control Flow Statement are used to make your shell scripts more flexible and can adapt to changing situations.

We are going to learn more types of control flow statements (both logical and loops) to give more flexibility and power to your shell scripts.



ADDITIONAL CONTROL FLOW STATEMENTS

How an if statement works

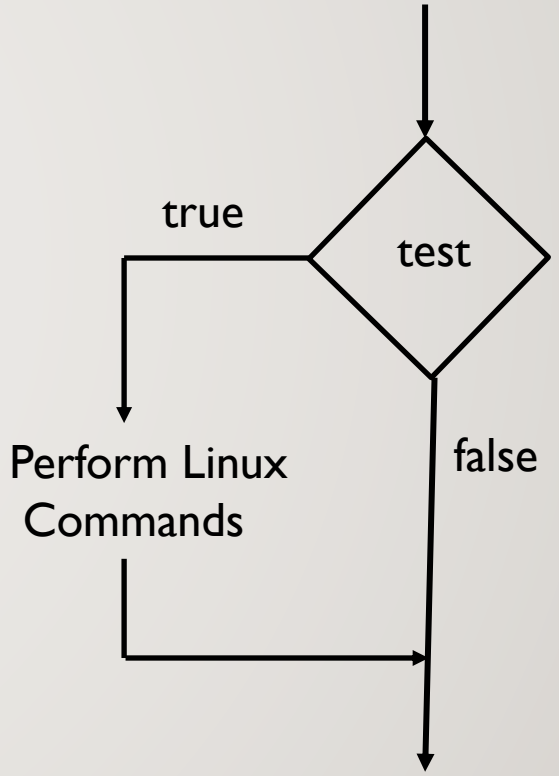
As you recall from a previous lesson, if the **test** condition returns a **TRUE** value, the Linux Commands between **then** and **fi** statements are executed.

If the test returns a **FALSE** value, then the if statement is bypassed.

Although useful, there are no Linux commands issued if the test condition returns a **FALSE** value.

Example:

```
num1=5
num2=10
if [ $num1 -lt $num2 ]
 then
    echo "Less Than"
fi
```



ADDITIONAL CONTROL FLOW STATEMENTS

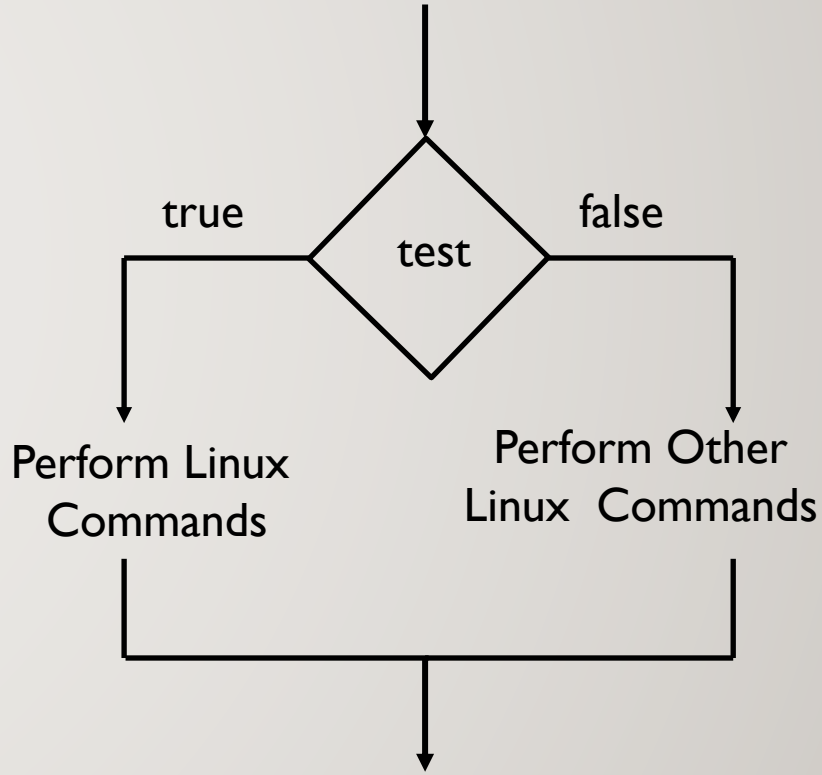
if-else Statements

If the test condition returns a **TRUE** value, then the Linux Commands between the **then** and **else** statements are executed.

If the test returns a **FALSE** value, then the the Linux Commands between the **else** and **fi** statements are executed.

Example:

```
num1=5
num2=10
if test $num1 -lt $num2
 then
    echo "Less Than"
else
    echo "Greater Than or Equal to"
fi
```



ADDITIONAL CONTROL FLOW STATEMENTS

Instructor Demonstration

Your instructor will demonstrate examples of using **if-else** statements.

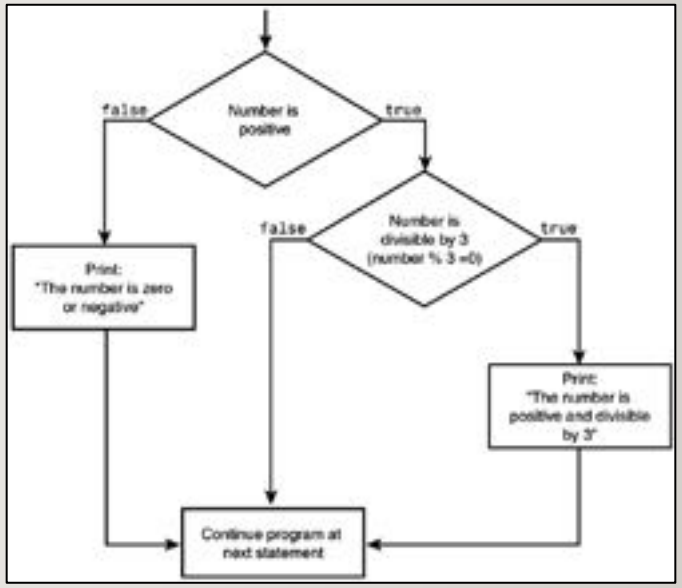
ADDITIONAL CONTROL FLOW STATEMENTS

if-elif-else Statements

If the test condition returns a **TRUE** value, then the Linux Commands between **then** and **else** statements are executed.

If the test returns a **FALSE** value, then a **new condition is tested**, and action is taken if the result is **TRUE**

Otherwise, an action will be taken if the new test condition is **FALSE**



Example:

```
num1=5;num2=10
if test $num1 -lt $num2
 then
    echo "Less Than"
elif test $num1 -gt $num2
 then
    echo "Greater Than"
else
    echo "Equal to"
fi
```

ADDITIONAL CONTROL FLOW STATEMENTS

Instructor Demonstration

Your instructor will demonstrate examples of using **if-elif-else** statements.

ADDITIONAL CONTROL FLOW STATEMENTS

Loop Statements

As discussed in a previous lesson, a **loop** statement is a *series of steps or sequence of statements* ***executed repeatedly*** *zero or more times satisfying the given condition*

Reference:
https://www.chegg.com/homework-help/definitions/loop-statement-3

ADDITIONAL CONTROL FLOW STATEMENTS

for Loop

In a previous lesson, you learned how to use the for loop using a **list**.

A list consists of **arguments** that are used for each iteration of the loop.

Example:

```
for item in list
do
    command(s)
Done
```

NOTE: There are other ways we can use the for loop (including **command substitution**) to allow our shell scripts to be more effective.

ADDITIONAL CONTROL FLOW STATEMENTS

Command Substitution

Command substitution *is a facility that allows a command to be run and its output to be pasted back on the command line as arguments to another command.*

Reference: https://en.wikipedia.org/wiki/Command_substitution

Usage:

`command1 $(command2)` or `command1 ` `command2` `

Examples:

```
file $(ls)
```

```
Mail –s “message” $(cat email-list.txt) < message.txt
```

ADDITIONAL CONTROL FLOW STATEMENTS

Instructor Demonstration

Your instructor will demonstrate examples of using **Command Substitution**.

ADDITIONAL CONTROL FLOW STATEMENTS

for Loop using Command Substitution

In the example below, we will use **command substitution** to issue the **ls** command and have that output (filenames) become **arguments** in the **for** loop list.

Example:

```
for var in $(ls)
do
    echo "Filename is: $var"
done
```

```
ls
file1  file2  file3  for-command-substitution.bash

cat for-command-substitution.bash
#!/bin/bash

for var in $(ls)
do
  echo "Filename is: $var"
done

./for-command-substitution.bash
Filename is: file1
Filename is: file2
Filename is: file3
Filename is: for-command-substitution.bash
```

ADDITIONAL CONTROL FLOW STATEMENTS

Instructor Demonstration

Your instructor will demonstrate examples of for loops using **Command Substitution**.

ADDITIONAL CONTROL FLOW STATEMENTS

Using the while Loop Statement

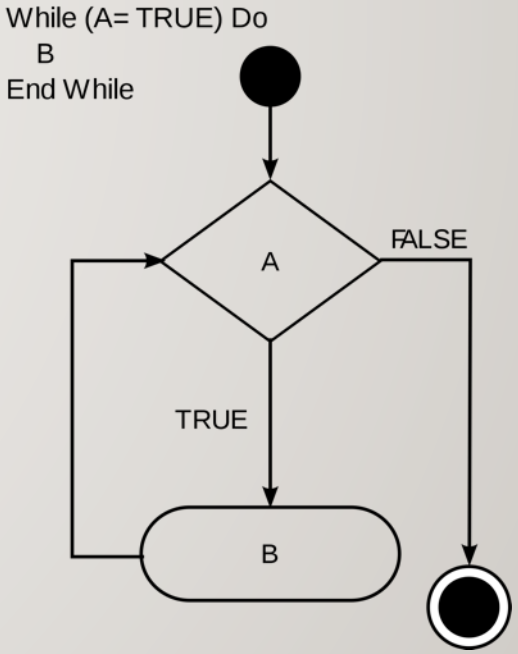
The condition/expression is evaluated, and if the condition/expression is ***TRUE***, the code within … the block is executed.

This repeats until the condition/expression becomes **FALSE**.

Reference: https://en.wikipedia.org/wiki/While_loop

Example:

```
answer=10
read –p “pick a number between 1 and 10: “ guess
while test $guess –eq 10
do
    read –p “Try again: “ guess
done
echo “You are correct”
```



ADDITIONAL CONTROL FLOW STATEMENTS

Instructor Demonstration

Your instructor will demonstrate examples of using the **while** loop.

ADDITIONAL FEATURES

Start-up Files

Shell configuration (start-up) files are **scripts** that are run when you **log in, log out**, or **start a new shell**. Start-up files can be used, for example, to set the prompt and screen display, create local variables, or create temporary Linux commands (aliases).

The file pathname **/etc/profile** belongs to the **root** user and is the first start-up file that executes when you log in, regardless of shell.

User-specific config start-up files are in the user's home directory:
~/.bash_profile runs when you log in **~/.bashrc** runs when you start an interactive subshell.

ADDITIONAL FEATURES

Logout Files

There are files that reset or restore the environment or properly shut-down running programs when the user logs out of their shell.

User-specific logout start-up files are in the user's home directory:

~/.bash_logout

ADDITIONAL CONTROL FLOW STATEMENTS

Instructor Demonstration

Your instructor will demonstrate examples of using **start-up** and **logout** files.

ADDITIONAL CONTROL FLOW STATEMENTS / FEATURES

Getting Practice

To get practice to help perform **assignment #3**, perform **Week 12 Tutorial:**

- INVESTIGATION 1: ADDITIONAL LOGIC STATEMENTS
- INVESTIGATION 2: ADDITIONAL LOOPING STATEMENTS
- INVESTIGATION 3: USING STARTUP FILES
- LINUX PRACTICE QUESTIONS (Questions **1 - 8**)