



Lab #5

Loops, Conditionals, and Parameters

BINF 2111, Fall 2024





Terminology



Conditional

A programming element that tells the computer to execute certain actions, provided certain conditions are met



Loop

A programming element that repeats a portion of code a set number of times until the desired process is complete



Parameter

A special kind of variable used in a function or program to refer to one of the pieces of data provided as input (sometimes called **input variables**)



Iterate

The repetition of a section of code within a computer program for a number of instances or until status is encountered





Commands To Know

Commands are
case
sensitive!!

- Command
- Options
- Input (like a file or folder)
- RegEx

Command	Meaning	Usage
diff	Find the differences between two files	<code>diff [file1] [file2]</code>
sed	Stream editor. Diverse command that can manipulate text/files.	<code>sed [options] '[regex]'</code> <code>[file]</code>
if	Conditional statement to compare two data points	<code>if [[condition]]; then</code> <code>commands</code> <code>fi</code>
for	Loop statement to look through a group of elements and run a command on each of those elements, one at a time	<code>for i in group; do</code> <code>commands</code> <code>done</code>



Command Breakdown - diff

- **diff:** finds the differences between two files
 - Useful Options
 - -i Case insensitive, ignore capitalizations
 - Usage
 - `diff file1.txt file2.txt`
 - Understanding Output
 - Line numbers corresponding to the first file
 - A special symbol
 - a: add
 - c: change
 - d: delete
 - Line numbers corresponding to the second file



Command Breakdown - diff

- Understanding Output
 - Lines preceded by a `<` are lines from the `first file`.
 - Lines preceded by a `>` are lines from the `second file`.
 - The three dashes separate the lines of file1 and file2

- `2d1`: line 2 in file1 needs to be `deleted` to match line 1 in file2

- `6c5`: line 6 in file1 needs to be `changed` to match line 5 in file2

```
madelinebellanger@Madelines-iMac:~/Desktop/BINF2111/F24/Lab5$ cat file1.txt
hello
how are you

testing
this is a line
-
this is lab 5%
madelinebellanger@Madelines-iMac:~/Desktop/BINF2111/F24/Lab5$ cat file2.txt
hello

testing
this is a line

this is lab 5%
madelinebellanger@Madelines-iMac:~/Desktop/BINF2111/F24/Lab5$ diff file1.txt file2.txt
2d1
< how are you
6c5
<
---
>
```

Command Breakdown - sed

- **sed**: stream editor. Diverse command that can manipulate text/files
 - Mac users need to use gsed
 - Printing specific lines
 - Print line three only
 - `sed -n '3p' file.txt`
 - Print lines four and six
 - `sed -n '4p;6p' file.txt`
 - Print lines 2 through 5
 - `sed -n '2,5p' file.txt`



Conditionals - If Statements

- Used to evaluate if a statement/condition is true or false

```
if [[ condition ]]; then
    commands
fi
```

- Useful Conditions

Meaning

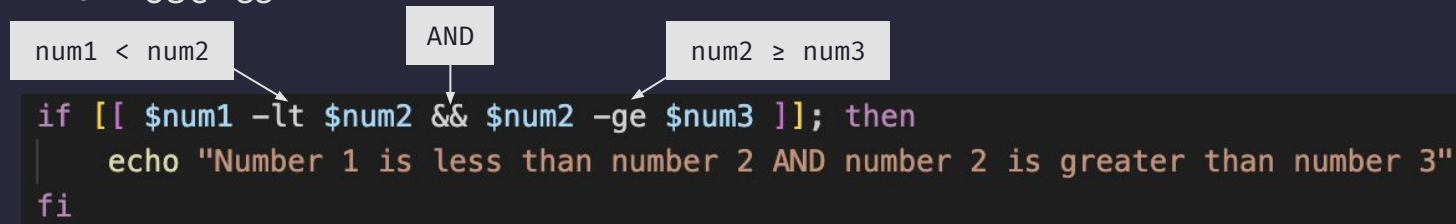
<code>[[number1 -gt number2]]</code>	Is <code>number1 > number2</code> ?
<code>[[number1 -lt number2]]</code>	Is <code>number1 < number2</code> ?
<code>[[number1 -ge number2]]</code>	Is <code>number1 ≥ number2</code> ?
<code>[[number1 -le number2]]</code>	Is <code>number1 ≤ number2</code> ?
<code>[[-z string1]]</code>	Is <code>string1</code> empty?
<code>[[-n string1]]</code>	Is <code>string1</code> not empty?
<code>!</code>	Opposite (usually not)



If Statements - Multiple Evaluations

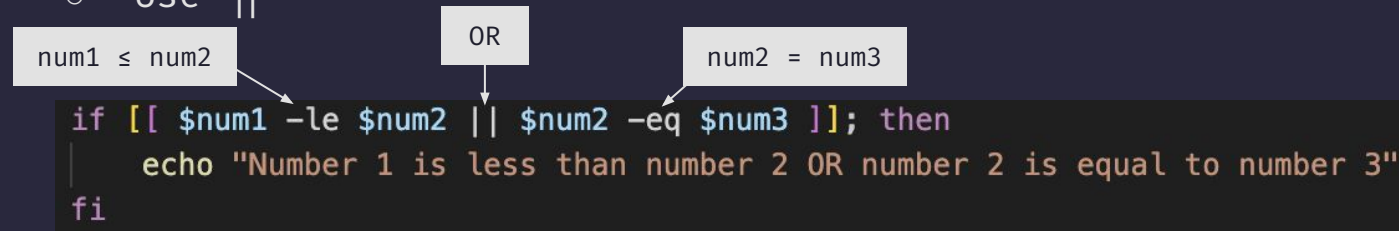
- Test if statement 1 and statement 2 are **both true**

- Use `&&`



- Test if statement 1 or statement 2 is **true**

- Use `||`



If Statements - Multiple Evaluations

- Do something else if the statement is **false**

- Use else

num1 < num2

```
if [[ $num1 -lt $num2 ]]; then
    echo "Number 1 is less than number 2"
else
    echo "Number 1 is NOT less than number 2"
fi
```

Only run if
num1 < num2

Only run if
num1 ≥ num2

- Test if **statement 2** is **true** when **statement 1** is **false**

- Use elif

num1 ≥ num2

Else if

num1 ≤ num2

```
if [[ $num1 -ge $num2 ]]; then
    echo "Number 1 is greater than or equal to number 2"
elif [[ $num1 -le $num2 ]]; then
    echo "Number 1 is less than or equal to number 2"
fi
```

Only run if
num1 ≥ num2

Only run if
num1 ≤ num2

For Loops

- Loop statement to look through a group of elements and run a command on each of those elements, one at a time

```
for i in group; do
    Command ran on every i in group
done
```

- Common Uses

- Looping through files that end in .txt

File variable

```
for file in *.txt; do
    wc -l $file
done
```

Any file that ends in .txt

- Looping through a range of numbers

Number variable

```
for i in {1..10}; do
    ((sum+=i))
    echo "The sum of all the numbers thus far: $sum"
done
```

List numbers 1-10

Parameters Within Scripts

- Each parameter is specified by a dollar sign and the parameter number
 - \$1, \$2, \$3, etc.
- Can be set equal to variables or used as a variable itself

Parameters set
equal to
variables

```
param1=$1      # the first parameter used when running the program
param2=$2      # the second parameter used when running the program
```

```
echo "These parameters are set to variables within the code: "
echo "This is the first parameter: $param1"
echo "This is the second parameter: $param2"
```

```
echo "These parameters are not set to variables within the code: "
echo "These are a couple more parameters: $3, $4"
```

Parameters used
as variables

```
if [[ ${#param1} -ge ${#3} ]]; then
    echo "Parameter 1 is longer than parameter 3"
fi
```

Running Scripts with Parameters

- When using parameters, they are set when the script is called:
 - `bash script.sh parameter1 parameter2 parameter3 parameter4`
- If the following command was ran:
`bash script.sh hello how are you`
 - Parameter 1 (`$1/$param1`) would be set to `hello`
 - Parameter 2 (`$2/$param2`) would be set to `how`
 - Parameter 3 (`$3`) would be set to `are`
 - Parameter 4 (`$4`) would be set to `you`

```
madelinebellanger@Madelines-iMac:~/Desktop/BINF2111/F24/Lab5$ bash parameters.sh hello how are you
These parameters are set to variables within the code:
This is the first parameter: hello
This is the second parameter: how

These parameters are not set to variables within the code:
These are a couple more parameters: are, you
Parameter 1 is longer than parameter 3
```

Pseudocode

- Outline of code written in plain language, so that anyone can understand it
- Usually written as comments, deleted after code is written

```
# Prompt: Print out numbers 1 - 10, distinguishing between odd and even numbers.
```

```
# The actual code
```

```
for num in {1..10}; do
    if [[ $(( $num%2 )) -eq 0 ]]; then
        echo "$num is even"
    else
        echo "$num is odd"
    fi
done
```

```
The pseduocode
```

```
#for loop through 1..10
#   if num is even
#       print "$num is even"
#   else
#       print "$num is odd"
#   end if
#end for
```