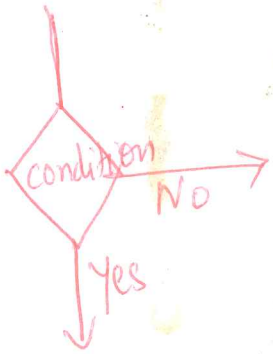
# STRUCTURED COMMANDS

CWI
Centrum Wiskunde & Informatica

## if then statement




```
if command
then
        commands
fi
```

- In other programming languages, the object after if statement is an equation that is Evaluated for a TRUE or FALSE value.
- That's not how bash shell if statement works.
- The bash shell if statement runs the command defined on the if line.
  - Exit status of command = 0, commands under then executed
  - Exit status = anything else, the then part is not executed.

# if - then - else Statement

```
if command
then
    commands ]← if exit status=0
else
    commands ] ← if exit status ≠0
fi
```

Single line: `if command; then commands; else commands; fi`

## Nesting ifs

- Sometimes you must check for several situations in your script.
- Instead of having to write separate if-then statements, you can use an alternative version of the else section called elif.

```
if command1
then
    commands
elif command2
then
    more commands
fi
```

(3)

# The test command



- You have seen in the if Statement line are normal shell commands.
- So, does the bash if-then Statements has the ability to evaluate any condition other than the exit status code of a command?

  Answer: NO, however, there is a utility available called test in the bash shell that helps you evaluate other things using if-then statement.

- The test command provides a way to test different conditions in an if-then Statement.
  - If condition evaluates to TRUE, test command exits with 0 code
  - If condition evaluates to FALSE, test command exits with 1 code.

```
if test condition
then
    commands
fi
```

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

(space)

(4) test command can evaluate three classes of conditions:

- Numeric comparisons
- String comparisons
- File comparisons

## Numeric comparisons (Text codes for numeric comparison)

n1 -eq n2 → n1 = n2
n1 -ge n2 → n1 >= n2
n1 -gt n2 → n1 > n2
n1 -le n2 → n1 <= n2
n1 -lt n2 → n1 < n2
n1 -ne n2 → n1 != n2

## String comparisons (Standard mathematical comparison symbols)

str1 = str2
str1 != str2
str1 < str2
str1 > str2
-n str1 ← checks if length of str1 > 0
-z str1 ← checks if length of str1 = 0

(5)

## String order



When trying to use the > or < thant features of test command:

- > and < must be escaped
- Greater than and less than order is not same as that used with sort.

Example: Capitalized letters are treated less than lowercase letters in test command. But, sort puts lowercase first.

test uses ASCII ordering, using each character's ASCII numeric value to determine the sorting order.

sort command uses the sorting order defined for the system locale language settings. For English language, lowercase appear before uppercase.

⑥ File Comparisons

-d file ← check if file exists and is a directory
-e file ← check if file exists
-f file ← check if file exists and is a file.
-r file ← check if file exists and readable
-s file ← file exists and not empty
-w file ← file exists and writable
-x file ← file exists and executable
-O file ← file exists and owned by current user
-G file ← file exists and default group is same as current user
file1 -nt file2 ← file 1 newer than file2
file1 -ot file2 ← file 1 older than file2

## Compound Condition Testing

- Two boolean operators you can use:

[condition1] && [condition2]

[condition1] || [condition2]

(7)

# Advanced if-then Features

CWI
Centrum Wiskunde & Informatica

— New additions to the bash shell that provide advanced features that you can use in if-then statements

- Double parentheses (( )) for mathematical expressions.
- Double square brackets for advanced string handling functions
  [[ ]]

(( )) ← no need to escape ~~espace~~ > or <
example if (( $val1 ** 2 > 90 ))

[[ ]] ← allows pattern matching
example if [[ $USER == r* ]]

escape

(8) Case command

```
case variable in
pattern1 | pattern2) commands1 ;;
pattern3) command2 ;;
*) default commands ;;
esac
```

- The case command compares the variable specified against the different patterns. If the variable matches the pattern, the shell executes the commands specified for the pattern.
- You can list more than one pattern on a line, using the bar operator to separate each pattern.
- The * is the catch all for values that don't match any of the listed patterns.