OPERATING SYSTEMS

- Laboratory 3 -

SHELL PROGRAMMING

1. INTRODUCTION

* shell = a special program/language that provides an interface between user and the operating system kernel
* shell types: *sh (Bourne shell)*, *csh (C shell)*, *ksh (Korn shell)*, *bash (GNU Bourne-again shell)*
* script = a text file that contains commands (internal or external)
* bash script example:

|  |
| --- |
| #!/bin/bash  pwd  ls |

Important:

* + the characters #! On the first line of the script is NOT a simple comment (it’s called *shebang*) and after it must follow the absolute path to the program that must be run for the next lines of the script, in this case /bin/bash
* to run a bash script we must add execution permissions first, then run with ./scriptname:

chmod +x script\_1.sh

./script\_1.sh

* comments start with #(hash)
* variables:
  + variable names can contain letters, digits and „ \_” *(underscore*), first character must be a letter, reserved words can not be used as var names
  + all is case sensitive (distinction between uppercase and lowercase letters
  + examples:

n=45

name=Ana

msg="Enter a number:"

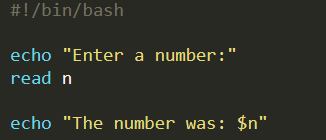
* reserved words *(keywords)*:

if then else elif fi

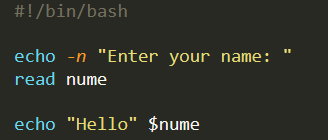
for while until do done

case in esac

* internal commands *(built-in commands)*:
* print the list of internal commands: help
* print information about a command: help ***command***
* examples: echo read printf test
* bash scripts:
* read and print a number: script\_2.sh



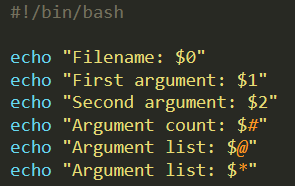
* read and print a string: script\_3.sh



* special variables:

|  |  |
| --- | --- |
| $0 | Name of the script file |
| $1,..., $9 | Command line arguments given for execution |
| $# | Number of command line arguments given |
| $\* | Array of command line arguments |
| $@ | List of individual command line arguments |
| $? | Exit code (exit status) of the last executed command |
| $$ | PID-ul of the current process |
| $! | PID-ul of the last command launched in background |

* example: script\_4.sh

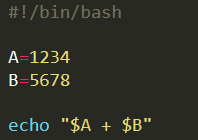


Use chmod +x script\_4.sh to add permissions, then run with various arguments:

./script\_4.sh 1 2 3 string word “hello everyone“

2. ARITHMETICAL EXPRESSIONS WITH INTEGER NUMBERS

* a shell variable is implicitly considered a string
* example: script\_5.sh



2.a. Command expr

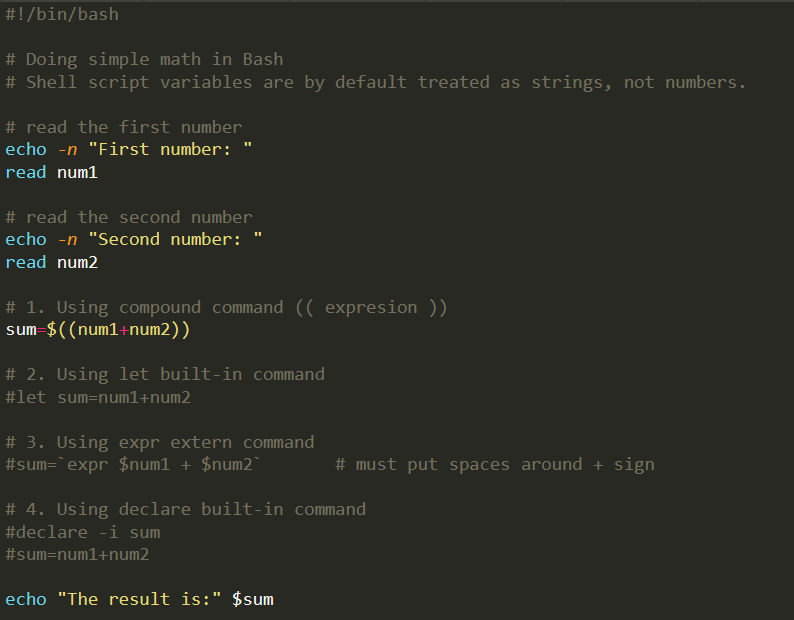
|  |
| --- |
| **expr** *expression* |

* evaluates and prints at standard output the value of an arithmetical integer expression
* operators:

|  |  |
| --- | --- |
| + - \\* / // % | Sum, diff, mult, div , mod |
| = != | Numerical comparisons: |
| \> \>= | - return 1 if the relation between s and d is true |
| \< \<= | - else returns 0 |
| \( \) | For subexpressions (parenthesis) |
| S \| D | return S if S is not NULL and not 0, return D otherwise |
| S \& D | return S if both S and D are not NULL and not 0, 0 otherwise |
| length S | Length of S |
| index S CHARS | Position of the first occurrence in S or 0 (index starts at 1) |
| substr S P L | Substring starting with S on position P and length L |

2.b. Double parenthesis

* example: script\_6.sh



3. COMMAND test

* syntax:

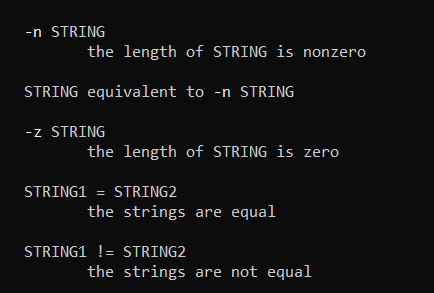
|  |
| --- |
| **test** *condition* **or** **[** *condition* **]** |

* evaluate *condition* and return 0 if true, otherwise a nonzero value
* allow string/integer comparisons, and file options checking

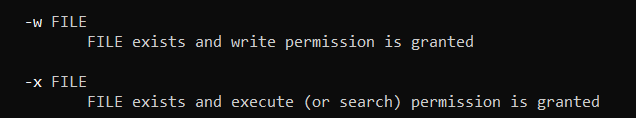
3.a. Compare integers

* relational operators: -lt -le -eq -ne -ge -gt
* AND /OR / NOT: -a, -o , ! ,
* **see** man test

3.b. String comparison



3.c. File checking options

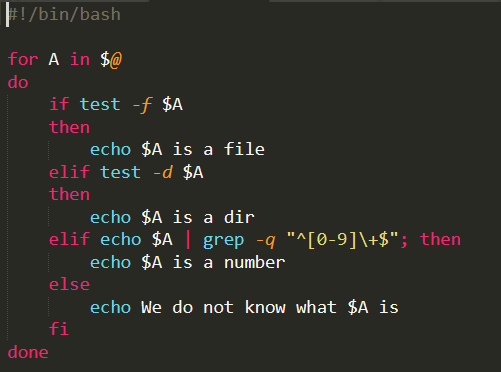


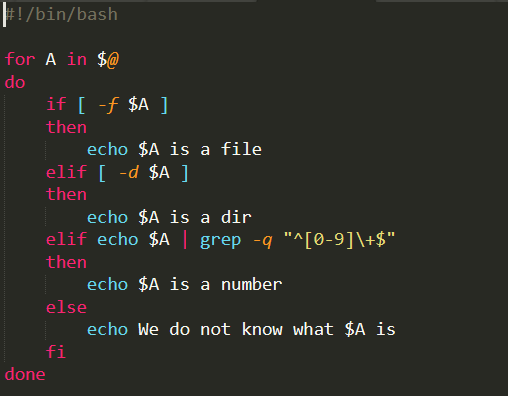
4. IF/THEN/ELIF/ELSE/FI

* syntax:

|  |
| --- |
| **if** *condition*  **then**  statement(s) to be executed  **elif** *condition*  **then**  statement(s) to be executed  **elif** *condition*; **then**  statement(s) to be executed  **else**  statement(s) to be executed  **fi** |

* examples: if\_1.sh, if\_2.sh



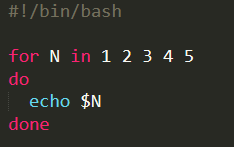
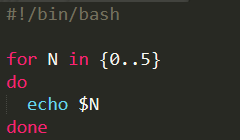
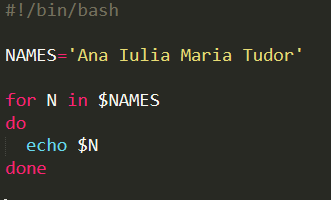


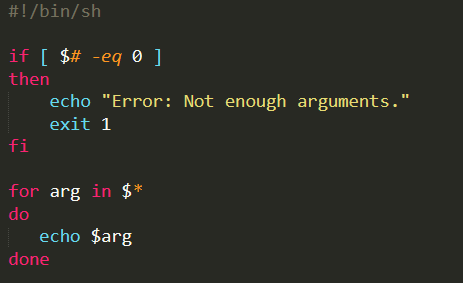
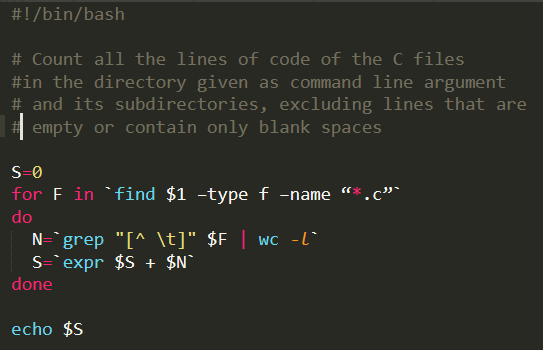
5. FOR/DO/DONE

* syntax:

|  |
| --- |
| **for** *var* **in** *list*  **do**  statement(s) to be executed  **done** |

* examples: for\_1.sh, for\_2.sh, for\_3.sh, for\_4.sh, for\_5.sh

* specify a pattern for a file name (*filename wildcards, diferite de regular expressions*)

|  |  |
| --- | --- |
| \* | Any sequence of characters, even empty (except first dot - beginning of file) |
| ? | One character (except first dot at the beginning) |
| [abc] | Any character in the list between [ ] |
| [!abc] | Any character not in the list between [! ] |

* example:
* print files with a name that starts with a letter and have extension of two letters:

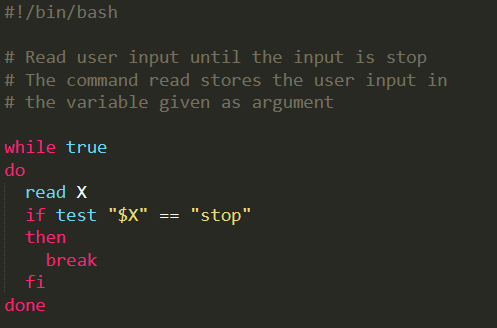
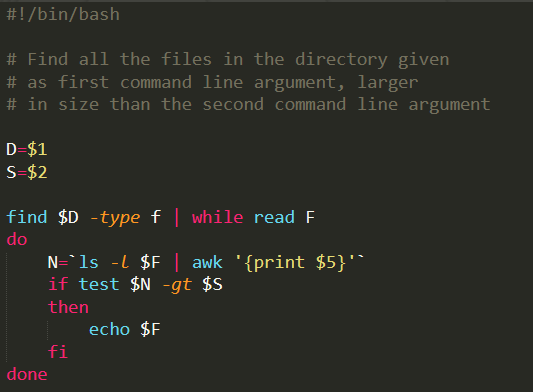
ls [a-zA-Z]\*.??

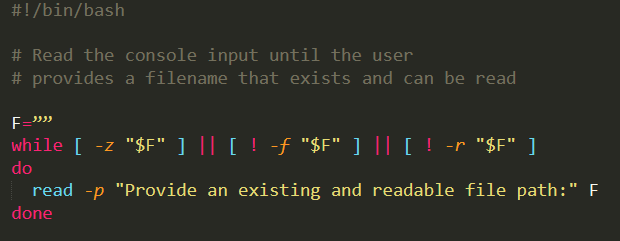
6. WHILE/DO/DONE, UNTIL/DO/DONE

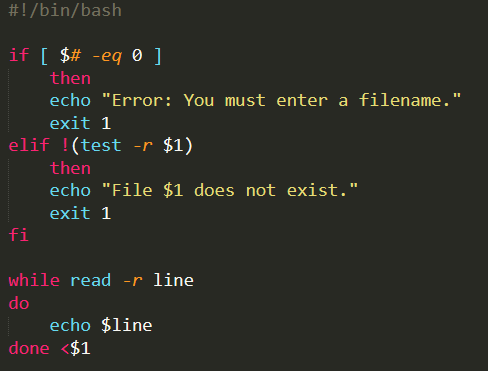
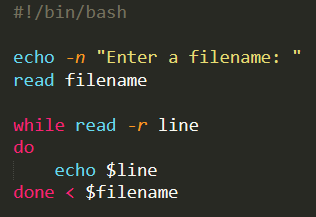
* syntax:

|  |  |
| --- | --- |
| **while** *condition*  **do**  statement(s) to be executed  **done** | **until** *condition*  **do**  statement(s) to be executed  **done** |

* examples: while\_1.sh, while\_2.sh, while\_3.sh, while\_4.sh, while\_5.sh



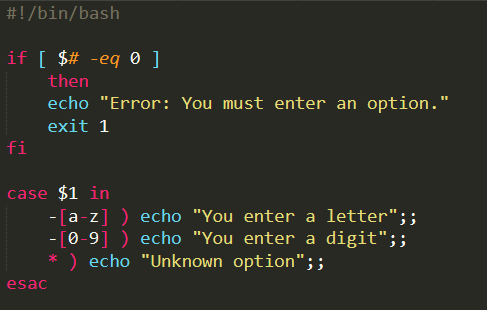
 

7. CASE/ESAC

* syntax:

|  |
| --- |
| **case** *var* **in**  ***pattern\_1*)**  statement(s) to be executed if***patern\_1*** is matched**;;**  ***pattern\_2*)**  statement(s) to be executed if***patern\_2*** is matched**;;**  **...**  **\*)**  default condition to be executed**;;**  **esac** |

* examples: case\_1.sh



You can use C like syntax expressions with (( )), example: *if (( $num <= 5 && $a > 9 ))* … or

*for (( i=2; %i<=$N; i++ )) ; do …*  or *f=$(( f\*$i )) …*

8. OTHER USEFUL COMMANDS

8.a. Command cut

cut -d: -f 1 /etc/passwd

cut -d ":" -f 5 /etc/passwd

who | cut -d " " -f 1,11

8.b. Command find

find . -type f -name "\*.sh"

find /tmp -type d -empty

8.c. Command shift

shift [n] shift left with n positions the arguments given in command line

shift 2 (first two command line arguments are deleted)

8.d. Command sleep

sleep [n] suspend execution for n seconds

8.e. Command exit

exit [n] terminate execution and return to the process that launched it

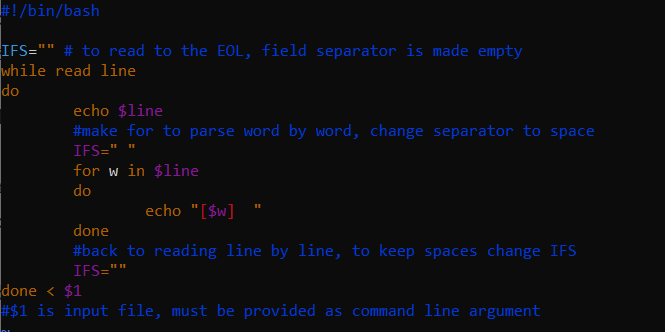
exit 0 – SUCCESS ; exit 1 – ERROR CODE 1

8.f. Command read

read var

read -p “Give a number: ” n # read with prompt

- to read from a file line by line we need to change IFS internal field separator variable. See more in man read; Example:



REFFERENCES:

* Shell programming: https://ryanstutorials.net/bash-scripting-tutorial/