



HAL
open science

Shell Scripts Documentation . Release 24/09/2021

Matthieu Cabos

► **To cite this version:**

| Matthieu Cabos. Shell Scripts Documentation . Release 24/09/2021. 2021. hal-03412512

HAL Id: hal-03412512

<https://hal.science/hal-03412512v1>

Preprint submitted on 3 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Shell Scripts Documentation

Release 24/09/2021

CABOS Matthieu

Sep 24, 2021

CONTENTS:

- 1 Compile** **1**
 - 1.1 Algorithm Compile 2
 - 1.2 Script usage Compile 3
 - 1.3 Options Compile 4
 - 1.4 Source Compile 5

- 2 Converter** **21**
 - 2.1 Algorithm Converter 22
 - 2.2 Script usage Converter 23
 - 2.3 Options Converter 24
 - 2.4 Source Converter 25

- 3 Get_lib_list** **27**
 - 3.1 Algorithm get_lib_list 28
 - 3.2 Script Get_lib_list 29
 - 3.3 Source Get_lib_list 30

- 4 Transfert_ssh.sh** **33**
 - 4.1 Algorithm Transfert 34
 - 4.2 Script Usage Transfert 35
 - 4.3 Options Transfert 36
 - 4.4 Source Transfert 37

- 5 Sudo-upgrade-all** **41**
 - 5.1 Algorithm Sudo-upgrade-all 42
 - 5.2 Script Usage Sudo-upgrade-all 43
 - 5.3 Source Sudo-upgrade-all 44

- 6 Indices and tables** **45**

CHAPTER

ONE

COMPILE

1.1 Algorithm Compile

This script has been wrote to automate the compilation process. It gather gcc, g++ and fortran for each version of them.

It has been thought as a multiplexer defining the differents branches of the switch.

Each branch correspond to a specific way to act (with options, wit a given name, etc).

To understand it, I will explain the differents steps of the algorithm.

The first switch is ruled by the mode parameter defining the compilation directives as chain, modular, mpi, openmp and lib associations compilation.

Each af these branchs will be splitted into differents sections ruled by a main loop coursing parameters list :

- We **read extension** and define differents flags used to get the correct path
- For each extension we **split again the algorithm** using a switch for new falgs as :
 - **exe_flag** : It determine if the exe filename is specified or not
 - **rep_flag** : It determines if the repertory name is specified or not
- Once the path determined by the differents flags, we **automate the compilation process** calling the right method, for example :
 - `gcc $parameters $lib -o $name $lib_option || gcc $lib -L $parameters -o $name || error` will give a compilation via variables substitution. The `||` allow to test the first command, goes to the second if failed and finally call the error method to get a print. The differents used variables in this example are :
 - * **parameters** : is the source(s) file(s) to compile
 - * **lib** : is the specified Librairie to link
 - * **name** : is the specified Filename of the executable
 - * **lib_option** : is the differents added librairies option

Once the correct execute-path have been founded, the correct compilation call is applied.

1.2 Script usage Compile

This script has been developed to automate the compilation process.

It treat c, c++ and fortran source files. Compilation can be ruled with four modes :

- The **chain mode** realize a chain compilation mode : Each source file is compiled independantly from each other
- The **modular mode** realize a modular compilation using one main source file and the dependency modules and functions as source files.
- The **Mpi compilation** mode allow parallel compilation using Open Mpi
- The **Openmp compilation mode** allow parallel compilation using Open MP
- The **Librairies Linking Mode** allow modular compilation using Unix Librairies

This mode must be specified as argument.

The script take 2 types of arguments : the first one determine the mode between

- **1** (chain)
- **2** (modular)
- **3** (mpi compilation)
- **4** (openmp compilation)
- **5** (Librairies linking mode)

The others parameters are the source files to compile.

The source file must be **.c**, **.cpp** or **fortran** files.

Others extensions files **WILL NOT BE TREATED**.

You have to use the correct syntaxe specifying the mode for each execution :

```
./compile.sh mode source file 1 source file 2 ... source file n
```

In case of modular compilation, please to keep this parameter structure :

```
./compile.sh mode Main source file Module source file 1 Module source file 2 ...
```


1.3 Options Compile

- **-O** : In case of additional features like Libraries using an option like `math.h` or compiler directive options as `-pthread`, etc. . . . It will act as enlarged compiler options directive. Option(s) as following arguments (**MUST be specified as the last parameters**) : `./compile.sh mode source file 1 source file 2 ... source file n -O -lm ...`
- **-o** : If specified you should give the executable the name you want as following argument : `./compile.sh mode src_file -o executable name`
- **-d** : If the source file(s) are not in the current directory, the `-d` option should specified the directory to treat (`-d /my_project_to_compile_directory/` as example) `./compile.sh mode src_file -d src_file_repertory_relative_path`
- **-I** : In case of additional libs, you may define the path of access header files using the syntax `-I./path_to_include/` : `./compile.sh mode src_file -I /path_to_include/`
- **-L** : In case of additional libs, you may define the path of access lib files using the syntax `-L./path_to_lib` : `./compile.sh mode src_file -L/path_to_lib/`

1.4 Source Compile

```
#!/bin/bash

# Author : CABOS Matthieu
# Date : 28/09/2020

function help(){
    printf "Please to refer Documentation."
}

function error(){
    printf"
        An error ocured, please to check the help file using --help option or -
↪h option.
        "
    echo $USER #| mail -s "error" matthieu.cabos@tse-fr.eu
}

rep=`echo $1 | grep [0-9]`
if [ "$rep" = "" ] || [ $# -eq 0 ] || [ "$1" = "--help" -o "$1" = "-h" ] || [ $# -lt 2 ] ||
↪|| [ $1 -gt 5 ] || [ `echo $1 | grep [0-9]` = "" ] || [ $1 -le 0 ]
then
    help
    exit
fi

rep_flag=0
repertory=""
lib=""
ind=0
exe_name=""
exe_flag=0
param_list=""
lib_option=""
lib_opt_flag=0
mode=$1
arguments=""
exe=0
for i in $@
do
    if [ "$i" != "1" ] && [ "$i" != "2" ] && [ "$i" != "3" ] && [ "$i" != "4" ] && [
↪"$i" != "5" ]
    then
        arguments=$arguments" "$i
    fi
done

for i in $arguments
# Treating options flags
do
# Getting lib parameters
```

(continues on next page)

(continued from previous page)

```

if [ ` echo $i | grep "\-d.*" != "" ] 2> /dev/null
then
(( rep_flag+=1 ))
repertory=`echo $i | sed -e "s|-d||g"`
test=`echo $repertory | grep "/"`
if [ "$test" = "" ] 2> /dev/null
then
    repertory=$repertory"/"
fi
elif [ "$i" = "-l" ] && [ $ind -eq 0 ]
then
    (( ind+=1 ))
elif [ $ind -ne 0 ]
then
    lib="$lib" "$i"
elif [ "$i" = "-o" ] && [ $exe_flag -eq 0 ]
then
    exe_flag=1
elif [ $exe_flag -ne 0 ]
then
    exe_name=$i
    ((exe_flag=0))
    ((exe=1))
elif [ "$i" = "-0" ]
then
    lib_opt_flag=1
elif [ $lib_opt_flag -ne 0 ]
then
    lib_option=$lib_option" "$i
elif [ `echo $i | grep "\-L.*" != "" ] 2> /dev/null
then
    lib_option=$lib_option" "$i
elif [ `echo $i | grep "\-I.*" != "" ] 2> /dev/null
then
    lib_option=$lib_option" "$i
else
    param_list=$param_list" "$i
fi
done

if [ $exe -eq 1 ]
then
    ((exe_flag=1))
fi

relative_way=$repertory
parameters=""
name=" "
if [ $mode -eq 1 ]
# Executing script profile in Chain Compilation mode
then
    for i in $param_list

```

(continues on next page)

(continued from previous page)

```

do
    if [ "$i" != "" ]
    # Rebuilding the file name parameters list
    then
        parameters=$parameters" "$i
    fi
done
for i in $parameters
# Executing the compilation for each file as parameter
do
    e=${i#*.}
    if [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e" !
↪ = "f95" ]&& [ "$e" != "F90" ]&& [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03
↪ " ] 2> /dev/null
    then
        e=${i#*.*}
    fi
    while [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e
↪ " != "f95" ]&& [ "$e" != "F90" ]&& [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" !=
↪ "F03" ] 2> /dev/null
    # then
    do
        e=${i#*.*.*}
    done
    if [ $e = "c" ]
    # Getting the file extension
    then
        name=`basename $i '.c'`
        # Getting the .exe filename
        if [[ ! $lib = "" ]]
        then
            then
                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $i $lib -o $name $lib_option || gcc $lib -L $i -o
↪ $name || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $i $lib -o $exe_name $lib_option || gcc $lib -L $i -
↪ o $exe_name || error
                elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                then
                    gcc $relative_way$i $lib -o $relative_way$name $lib_
↪ option || gcc $lib -L $relative_way$i -o $relative_way$name || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                then
                    gcc $relative_way$i $lib -o $relative_way$exe_name $lib_
↪ option || gcc $lib -L $relative_way$i -o $relative_way$exe_name || error
                fi
            else
                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $i -o $name $lib_option || error

```

(continues on next page)

(continued from previous page)

```

        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
        then
            gcc $i -o $exe_name $lib_option || error
        elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
        then
            gcc $relative_way$i -o $relative_way$name $lib_option ||_
↪error

        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
        then
            gcc $relative_way$i -o $relative_way$exe_name $lib_
↪option || error
        fi
    fi
    # Compiling the code file as parameter
    elif [ "$e" = "cpp" ]
    then
        name=`basename $i '.cpp'`
        # Getting the .exe filename
        if [[ ! $lib = "" ]]
        then
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                g++ $i $lib -o $name $lib_option || g++ $lib -L $i -o $name ||_
↪error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                g++ $i $lib -o $exe_name $lib_option || g++ $lib -L $i -o $exe_
↪name || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                g++ $relative_way$i $lib -o $relative_way$name $lib_option ||_
↪g++ $lib -L $relative_way$i -o $relative_way$name || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
            then
                g++ $relative_way$i $lib -o $relative_way$exe_name $lib_option_
↪|| g++ $lib -L $relative_way$i -o $relative_way$exe_name || error
        fi
    else
        if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
        then
            g++ $i -o $name $lib_option || error
        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
        then
            g++ $i -o $exe_name $lib_option || error
        elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
        then
            g++ $relative_way$i -o $relative_way$name $lib_option || error
        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
        then
            g++ $relative_way$i -o $relative_way$exe_name $lib_option ||_
↪error
    fi
fi

```

(continues on next page)

(continued from previous page)

```

        fi
        # Compiling the code file as parameter
        elif [ "$e" = "f90" -o "$e" = "f95" -o "$e" = "F90" -o "$e" = "F" -o "$e"
↪" = "f03" -o "$e" = "F03" ]
        then
            e=".""$e"
            name=`basename $i $e`
            # Getting the .exe filename
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                gfortran -o $name $i $lib_option || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                gfortran -o $exe_name $i $lib_option || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                gfortran -o $relative_way$name $relative_way$i $lib_option ||_
↪error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
            then
                gfortran -o $relative_way$exe_name $relative_way$i $lib_option_
↪|| error
            fi
            # fi
        fi
    done
elif [ $mode -eq 2 ]
# Executing script profile in Modular Compilation mode
then
    for i in $param_list
    do
        if [ "$i" != "1" ]
        # Rebuilding the file name parameters list
        then
            if [ $rep_flag -eq 0 ]
            then
                parameters=$parameters" "$i
            elif [ $rep_flag -eq 1 ]
            then
                parameters=$parameters" "$relative_way$i
            fi
        fi
    done
    for i in $parameters
    # Brownsing parameters list
    do
        e=${i#*.}
        # Getting the file extension
        testeur_beg="{i#*."
        testeur_end="*." "
        ((counter=1))
        if [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e" !
↪" = "f95" ] && [ "$e" != "F90" ] && [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03" ] && [ "$e" != "f02" ] && [ "$e" != "F02" ]
↪ ] 2> /dev/null

```

(continued from previous page)

```

        then
            e=${i#*.*.}
        fi
        while [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e"
↪ != "f95" ] && [ "$e" != "F90" ] && [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" !=
↪ "F03" ] 2> /dev/null
            # then
            do
                e=${i#*.*.}
            done
            if [ "$e" = "c" ]
            then
                name=`basename $i '.c'`
                # Getting the .exe filename
                break
            elif [ "$e" = "cpp" ]
            then
                name=`basename $i '.cpp'`
                # Getting the .exe filename
                break
            fi
        done
        if [ "$e" = "c" ]
        then
            if [[ ! $lib = "" ]]
            then
                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters $lib -o $name $lib_option || gcc $lib -L
↪ $parameters -o $name || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters $lib -o $exe_name $lib_option || gcc
↪ $lib -L $parameters -o $exe_name || error
                elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                then
                    gcc $parameters $lib -o $relative_way$name $lib_option_
↪ || gcc $lib -L $parameters -o $name || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                then
                    gcc $parameters $lib -o $relative_way$exe_name
↪ $lib_option || gcc $lib -L $parameters -o $relative_way$exe_name || error
                fi
            else
                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters -o $name $lib_option || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters -o $exe_name $lib_option || error
                elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                then

```

(continues on next page)

(continued from previous page)

```

gcc $parameters -o $relative_way$name $lib_option || _
↪error
    elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
    then
        gcc $parameters -o $relative_way$exe_name $lib_option || _
↪error
    fi
    fi
    # Compiling the Modular file as parameters
    elif [ "$e" = "cpp" ]
    then
        if [[ ! $lib = "" ]]
        then
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                g++ $parameters $lib -o $name $lib_option || g++ $lib -L
↪$parameters -o $name || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                then
                    g++ $parameters $lib -o $exe_name $lib_option || g++
↪$lib -L $parameters -o $exe_name || error
                elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                then
                    g++ $parameters $lib -o $relative_way$name $lib_option _
↪|| g++ $lib -L $parameters -o $name || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                then
                    g++ $parameters $lib -o $relative_way$exe_name
↪$lib_option || g++ $lib -L $parameters -o $relative_way$exe_name || error
            fi
        else
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                echo "commande écrite : "
                echo "g++ $parameters -o $name $lib_option" || error
                echo "resultats obtenus : "
                g++ $parameters -o $name $lib_option
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                echo "commande écrite : "
                echo "g++ $parameters -o $exe_name $lib_option" || error
                echo "resultats obtenus : "
                g++ $parameters -o $exe_name $lib_option
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                echo "commande écrite : "
                echo "g++ $parameters -o $relative_way$name $lib_option
↪" || error
                echo "resultats obtenus : "
                g++ $parameters -o $relative_way$name $lib_option
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
            then

```

(continues on next page)

(continued from previous page)

```

                                echo "commande écrite : "
                                echo "g++ $parameters -o $relative_way$exe_name $lib_
↪option" || error

                                echo "resultats obtenus : "
                                g++ $parameters -o $relative_way$exe_name $lib_option

                                fi
                                fi
                                # Compiling the Modular file as parameters
                                elif [ "$e" = "f90" -o "$e" = "f95" -o "$e" = "F90" -o "$e" = "F" -o "$e" = "f03"
↪" -o "$e" = "F03" ]
                                then
                                    rename_flag=0
                                    for i in $parameters
                                    do
                                        e=${i#*.}
                                        if [ $e != "c" ] && [ $e != "cpp" ] && [ "$e" != "f90" ]&& [ "$e" !=
↪"f95" ]&& [ "$e" != "F90" ] && [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03" ]
↪]
                                        then
                                            e=${i#*.*}
                                            fi
                                            e="."$e"
                                            name=`basename $i $e`
                                                files=$files" "$i
                                        done
                                        if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                                        then
                                            gfortran -o $name $files $lib_option || error
                                        elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                                        then
                                            gfortran -o $relative_way$name $files $lib_option || error
                                        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                                        then
                                            gfortran -o $exe_name $files $lib_option || error
                                        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                                        then
                                            gfortran -o $relative_way$exe_name $files $lib_option || error
                                        fi
                                        if [ $rename_flag -eq 1 ]
                                        then
                                            rm -r tmp
                                        fi
                                    fi
                                elif [ $mode -eq 3 ]
                                # Executing script profile in MPI parallel Compilation mode
                                then
                                    for i in $param_list
                                    do
                                        if [ "$i" != "2" ]
                                        then
                                            parameters=$parameters" "$i
                                        fi

```

(continues on next page)

(continued from previous page)

```

done
    for i in $parameters
    do
        e=${i#*.}
        if [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e" != "f95" ] && [ "$e" != "F90" ] && [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03" ] 2> /dev/null
        then
            e=${i#*.*}
        fi
        # Getting the file extension
        while [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e" != "f95" ] && [ "$e" != "F90" ] && [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03" ] 2> /dev/null
        # then
        do
            e=${i#*.*.*}
        done
        if [ $e = "c" ]
        then
            name=`basename $i '.c'`
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                # Getting the .exe filename
                mpicc -o $name $i $lib_option || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                mpicc -o $exe_name $i $lib_option || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                mpicc -o $relative_way$name $relative_way$i $lib_option_
            || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
            then
                mpicc -o $relative_way$exe_name $relative_way$i $lib_
            option || error

        fi
        # Compiling the code file as parameter
        elif [ "$e" = "cpp" ]
        then
            name=`basename $i '.cpp'`
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                # Getting the .exe filename
                mpicxx -o $name $i $lib_option || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                mpicxx -o $exe_name $i $lib_option || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                mpicxx -o $relative_way$name $relative_way$i $lib_option_
            || error
    
```

(continues on next page)

(continued from previous page)

```

        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
        then
            mpicxx -o $relative_way$exe_name $relative_way$i $lib_
↪option || error
        fi
        # Compiling the code file as parameter
        elif [ "$e" = "f90" -o "$e" = "f95" -o "$e" = "F90" -o "$e" = "F
↪" -o "$e" = "f03" -o "$e" = "F03" ]
        then
            e=".$e"
            name=`basename $i $e`
            # Getting the .exe filename
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                mpifort -o $name $i $lib_option || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                mpifort -o $exe_name $i $lib_option || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                mpifort -o $relative_way$name $relative_way$i
↪$lib_option || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
            then
                mpifort -o $relative_way$exe_name $relative_way
↪$i $lib_option || error
        fi
        # fi
    fi
done
elif [ $mode -eq 4 ]
# Executing script profile in OpenMP parallel Compilation mode
then
    for i in $param_list
    do
        if [ "$i" != "3" ]
        # Rebuilding the file name parameters list
        then
            if [ $rep_flag -eq 0 ]
            then
                parameters=$parameters" "$i
            elif [ $rep_flag -eq 1 ]
            then
                parameters=$parameters" "$relative_way$i
            fi
        fi
    done
    for i in $parameters
    do
        e=${i#*.}
        if [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e" !
↪= "f95" ]&& [ "$e" != "F90" ]&& [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03
↪" ] 2> /dev/null

```

(continues on next page)

(continued from previous page)

```

        then
            e=${i#*.*.}
        fi
        # Getting the file extension
        while [ "$e" != "c" ] && [ "$e" != "cpp" ] && [ "$e" != "f90" ] && [ "$e"
↪ != "f95" ] && [ "$e" != "F90" ] && [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" !=
↪ "F03" ] 2> /dev/null
            # then
            do
                e=${i#*.*.}
            done
            if [ $e = "c" ]
            then
                name=`basename $i '.c'`
                # Getting the .exe filename
                break
            elif [ "$e" = "cpp" ]
            then
                name=`basename $i '.cpp'`
                # Getting the .exe filename
                break
            fi
        done
        if [ $e = "c" ]
        then
            if [[ ! $lib = "" ]]
            then
                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters $lib -o $name -fopenmp $lib_option ||
↪ gcc $lib -L $parameters -o $name -fopenmp || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters $lib -o $exe_name -fopenmp $lib_option
↪ || gcc $lib -L $parameters -o $exe_name -fopenmp || error
                elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                then
                    gcc $parameters $lib -o $relative_way$name -fopenmp $lib_
↪ option || gcc $lib -L $parameters -o $relative_way$name -fopenmp || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                then
                    gcc $parameters $lib -o $relative_way$exe_name -fopenmp
↪ $lib_option || gcc $lib -L $parameters -o $relative_way$exe_name -fopenmp || error
                fi
            else
                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters -o $name -fopenmp $lib_option || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                then
                    gcc $parameters -o $exe_name -fopenmp $lib_option ||
↪ error

```

(continues on next page)

(continued from previous page)

```

        elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
        then
            gcc $parameters -o $relative_way$name -fopenmp $lib_
↪option || error
        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
        then
            gcc $parameters -o $relative_way$exe_name -fopenmp $lib_
↪option || error
        fi
    fi
    # Compiling the Modular file as parameters
    elif [ $e = "cpp" ]
    then
        if [[ ! $lib = "" ]]
        then
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                g++ $parameters $lib -o $name -fopenmp $lib_option ||
↪g++ $lib -L $parameters -o $name -fopenmp || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                g++ $parameters $lib -o $exe_name -fopenmp $lib_option
↪|| g++ $lib -L $parameters -o $exe_name -fopenmp || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                g++ $parameters $lib -o $relative_way$name -fopenmp $lib_
↪option || g++ $lib -L $parameters -o $relative_way$name -fopenmp || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
            then
                g++ $parameters $lib -o $relative_way$exe_name -fopenmp
↪$lib_option || g++ $lib -L $parameters -o $relative_way$exe_name -fopenmp || error
            fi
        else
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                g++ $parameters -o $name -fopenmp $lib_option
↪|| error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                g++ $parameters -o $exe_name -fopenmp $lib_
↪option || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then
                g++ $parameters -o $relative_way$name -fopenmp
↪$lib_option || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
            then
                g++ $parameters -o $relative_way$exe_name -
↪fopenmp $lib_option || error
            fi
        fi
    fi
    # Compiling the Modular file as parameters

```

(continues on next page)

(continued from previous page)

```

        elif [ "$e" = "f90" -o "$e" = "f95" -o "$e" = "F90" -o "$e" = "F" -o "$e" = "f03"
↪" -o "$e" = "F03" ]
        then
            for i in $parameters
            do
                e=${i#*.*}
                if [ "$e" != "f90" ] && [ "$e" != "f95" ] && [ "$e" != "F90" ] &&
↪ [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03" ]
                then
                    e=${i#*.*.*}
                fi
                e="."$e
                name=`basename $i $e`
                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gfortran -o $name $i $lib_option -fopenmp || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                then
                    gfortran -o $exe_name $i $lib_option -fopenmp || error
                elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                then
                    gfortran -o $relative_way$name $i $lib_option -fopenmp_
↪ || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                then
                    gfortran -o $relative_way$exe_name $i $lib_option -
↪fopenmp || error
                fi
            done
        fi
    elif [ $mode -eq 5 ]
    #Executing script with librairies Linking mode
    then
        libs=""
        libflag="t"
        cflag="t"
        for i in $param_list
        do
            if [ "$i" != "4" ]
            # Rebuilding the file name parameters list
            then
                parameters=$parameters" "$i
            fi
        done
        for i in $parameters
        do
            e=${i#*.*}
            # Getting the file extension
            if [ $e != "c" ] && [ $e != "cpp" ] && [ "$e" != "f90" ] && [ "$e" !=
↪"f95" ] && [ "$e" != "F90" ] && [ "$e" != "F" ] && [ "$e" != "f03" ] && [ "$e" != "F03"
↪" ]
            then

```

(continues on next page)

(continued from previous page)

```

        e=${i#*.*.}
    fi
    if [ $e = "c" ]
    then
        name=`basename $i '.c'`
        # Getting the .exe filename
        cflag=$e
        elif [ "$e" = "cpp" ]
        then
            name=`basename $i '.cpp'`
            # Getting the .exe filename
            cflag=$e
        elif [ "$e" = "o" -o "$e" = "a" -o "$e" = "so" ]
        then
            libs="$libs"" "$i"
            libflag=$e
        fi
done
if [ $cflag = "c" ]
then
    cflag="."$cflag
    tocompile=$name$cflag
    if [ "$libflag" = "o" ]
    # Script profile in case of Object Librairie
    then
        if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
        then
            gcc -o $tocompile $libs $lib_option || error
        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
        then
            gcc $tocompile $libs -o $exe_name $lib_option || error
        elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
        then
            gcc $relative_way$tocompile $libs $lib_option ↵
↵ || error
        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
        then
            gcc $relative_way$tocompile $libs -o $relative_way$exe_
↵name $lib_option || error
        fi
        # Compiling the Modular Libs as parameters
        elif [ "$libflag" = "a" ]
        # Script profile in case of Static Librairie
        then
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
            then
                gcc $tocompile $libs $lib_option || error
            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
            then
                gcc $tocompile $libs -o $exe_name $lib_option || error
            elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
            then

```

(continues on next page)

(continued from previous page)

```

gcc $relative_way$tocompile $libs $lib_
↪option || error
    elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
        then
            gcc $relative_way$tocompile $libs -o $relative_
↪way$exe_name $lib_option || error
        fi
    elif [ "$libflag" = "so" ]
        # Script profile in case of Dynamic Librairie
        then
            if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                then
                    gcc -$libs -L $tocompile $lib_option || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                    then
                        gcc -$libs -L $tocompile -o $exe_name $lib_option || ↪
↪error
                elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                    then
                        gcc -$libs -L $relative_way$tocompile
↪$lib_option || error
                elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                    then
                        gcc -$libs -L $relative_way$tocompile -o
↪$relative_way$exe_name $lib_option || error
                    fi
                elif [ $cflag = "cpp" ]
                    then
                        cflag="."$cflag
                        tocompile=$name$cflag
                        if [ "$libflag" = "o" ]
                            # Script profile in case of Object Librairie
                            then
                                if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
                                    then
                                        g++ $tocompile $libs $lib_option || error
                                    elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
                                        then
                                            g++ $tocompile $libs -o $exe_name $lib_option || error
                                        elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
                                            then
                                                g++ $relative_way$tocompile $libs $lib_
↪option || error
                                            elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
                                                then
                                                    g++ $relative_way$tocompile $libs -o $relative_
↪way$exe_name $lib_option || error
                                                fi
                                                # Compiling the Modular Libs as parameters
                                                elif [ "$libflag" = "a" ]
                                                    # Script profile in case of Static Librairie

```

(continues on next page)


```

then
    if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
    then
        g++ $stocompile $libs $lib_option || error
    elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
    then
        g++ $stocompile $libs -o $exe_name $lib_option || error
    elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
    then
        g++ $relative_way$stocompile $libs $lib_
↳option || error
    elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
    then
        g++ $relative_way$stocompile $libs -o $relative_
↳way$exe_name $lib_option || error
    fi
    elif [ "$libflag" = "so" ]
    #Script profile in case of Dynamic Librairie
    then
        if [ $exe_flag -eq 0 ] && [ $rep_flag -eq 0 ]
        then
            libs="-"$libs"
            g++ $stocompile $libs $lib_option || error
        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 0 ]
        then
            g++ $stocompile $libs -o $exe_name $lib_option || error
        elif [ $exe_flag -eq 0 ] && [ $rep_flag -eq 1 ]
        then
            g++ $relative_way$stocompile $libs $lib_
↳option || error
        elif [ $exe_flag -eq 1 ] && [ $rep_flag -eq 1 ]
        then
            g++ $relative_way$stocompile $libs -o $relative_
↳way$exe_name $lib_option || error
        fi
    fi
fi

```

CONVERTER

2.1 Algorithm Converter

This script has been written to treat large amount of picture data. It must be used as a resizer.
It is ruled by parameters :

- **Resolution** : Define the DPI resolution of the picture
- **Folder** : Determine the root folder to treat
- **size** : Determine the Picture size, must have form $a \times b$ where a & b are integers (ex : 800x600)

In a first step we get essential information since command parameters such as Resolution, size, etc.

The mode is determined by the keyword '*clean*', if not specified the algorithm resizes all the pictures found in the folder and associated sub-tree, else it will remove all the resized pictures.

To treat them, we get in the `liste` variable the list of sub-folders paths.

We browse each folder in a loop and convert each picture file to the specified size and resolution.

2.2 Script usage Converter

Please to use this script with the correct parameters :

./ResConverter.sh **Resolution_DPI Folder size**

to convert all pictures with the specified DPI and size from given folder a root.

or

./ResConverter.sh **clean**

to clean the workspace

Where :

- **Resolution** is a value between 72 and 500 DPI
- **Folder** is the name of the folder containing pictures
- **size** is the picture length in pixel. Must have form “800x600”.

This script act recursively and will resize all the pictures contained into sub-tree

2.3 Options Converter

- **clean** : Will remove all your generated resized pictures.

2.4 Source Converter

```
#!/bin/bash

# Author : CABOS Matthieu
# Date : 23/09/2021

function usage(){
    echo "
    Please to refer Documentation.
"
}

if [ $# -eq 0 ]
then
    usage
    exit
fi

ind=1
param=$*
mode=0
size=0

for i in $param
do
    if [ $ind -eq 1 ]
    then
        if [ "$(echo $i | grep "^[[:digit:]]*$")" ]
        then
            Resolution=$i
        elif [ $i == "clean" ]
        then
            mode=1
            break
        fi
    elif [ $ind -eq 3 ]
    then
        size=$i
    fi
    ind=$((ind+1))
done

path="./"
Folders=`find $2 -type d`

for f in $Folders
do
    liste=`ls $f`
    if [ $mode -eq 0 ]
    then
        for item in $liste
```

(continues on next page)

(continued from previous page)

```
        do
            echo $f/"$item
            if [ -f $f/"$item ]
            then
                ↪"/resized"$item          convert $f/"$item -resize $size -density $Resolution $f
            fi
        done
    else
        find . -type f -name 'resized*.jpg' -delete
    fi
    liste=""
    path="./"
done
```

GET_LIB_LIST

3.1 Algorithm get_lib_list

This algorithm has been wrote to manage Librairies missing notifications when compiling with gcc.

In a first step, I'm coursing arguments list (wich is the results of a failes compilation).

In case of Librairie name founded, I update the lib variable and the Lib Flag to True.

In a second step, for each founded librairies, I get the exact name of the librairy via the **basename** command.

On the final loop, I try, for each librairy (from the exact name) to install it via the **vcpkg** command, if founded on server, It will install it, else it will give you the missing Librairies name.

3.2 Script Get_lib_list

Script Usage.

Please to use this script with the correct arguments. This script analyse the results of a failed compilation and determine the name(s) of the missing librairie(s).

Please to launch using the following syntax :

```
./get_lib_list.sh`gcc my_source_file.c`
```

3.3 Source Get_lib_list

```
#!/bin/bash

# Author : CABOS Matthieu
# Date : 31/08/2020

lib_flag=0
lib=""
for i in $@
# Coursing arguments list
do
    if [ $lib_flag -eq 1 ]
    # Getting lib name if founded
    then
        lib="$lib" "$i"
        ((lib_flag=0))
    fi
    if [ "$i" = "#include" ]
    then
        ((lib_flag=1))
    else
        ((lib_flag=0))
    fi
done
for libp in $lib
do
    ind=0
    for i in $(seq 1 ${#libp})
    do
        lettre=$(echo $libp | cut -c$i)
        if [ ! $ind -eq 0 ]
        then
            name_lib="$name_lib"$lettre"
        fi
        ((ind=$ind+1))
    done
    name_lib=`basename $name_lib '.h>'`
    # name_lib="$name_lib" "
    name_lib_final="$name_lib_final" "$name_lib"
    name_lib=""
done
for i in $name_lib_final
do
    test=""
    test=`vcpkg search $i | grep "[0-9]"`
    if [ "$test" = "" ]
    then
        echo "#####"
        ↪#####"
        echo "The librairy ""$i"" is not disponible on the Unix Server, please ↪
        ↪to install it manually."
```

(continues on next page)

(continued from previous page)

```
                echo "#####"
↪#####"
        else
            install_name="$install_name" "$i"
        fi
done
for i in $install_name
do
    vcpkg install $i
done
```


TRANSFERT_SSH.SH

4.1 Algorithm Transfert

This script has been thought to manage ssh Files transfert from the existing ip & User id. In fact, the command **scp** will be used from differents switches branches :

- **File upload** : Permit to upload a file to the specified ssh platform
- **File download** : Permit to download a file to the specified ssh platform
- **Folder upload** : Permit to upload a folder to the specified ssh platform
- **Folder download** : Permit to download a folder to the specified ssh platform

These branches are ruled by the parameter mode.

In the first step, we get variables informations since parameters values. Once done, we update from infos, the differents flags and the defaults paths. These updated fields will be used in the last part of the algorithm :

- **Updating the path**
- **Using the scp command** with the correct arguments to automate tranfert.

4.2 Script Usage Transfert

Please to use the script with the correct number of arguments :

```
./transfert.sh mode user source_folder destination_folder filename/foldername IP
```

Where :

- **mode** is the way to transfert between :
 - **1** mean **upload** file to the ssh specified destination folder
 - **2** mean **download** file since the ssh specified source folder
 - **3** mean **upload** folder to the ssh specified destination folder
 - **4** mean **download** folder since the ssh specified source folder
- **user** is your standard user name on the ssh platform
- **Source_folder** is the name of the source repertory
- **Destination_folder** is the name of the destination repertory
- **filename** is the exact files name to transfert or the folder name to transfert
- **IP_adress**”

4.3 Options Transfert

-help : Get the linux-like help from the command prompt.

4.4 Source Transfert

```
#!/bin/bash
# Author : CABOS Matthieu
# Date : 23/07/2020

function usage(){
    echo "Please to refer Documentation."
}

param=$*
ind=1
files=""
index=0
home_flag_src=0
home_flag_dst=0
IP=0
if [ $# -eq 0 ] || [ $# -ne 6 ] || [ $1 -gt 4 ] || [ $1 -le 0 ]
then
    usage
    exit
fi

if [ "$1" = "--help" ]
then
    usage
    exit
fi

for i in $param
do
    if [ $ind -eq 1 ]
    then
        mode=$i
    elif [ $ind -eq 2 ]
    then
        user=$i
    elif [ $ind -eq 3 ]
    then
        if [ $i = ~ ]
        then
            home_flag_src=1
        fi
        source=$i
    elif [ $ind -eq 4 ]
    then
        if [ $i = ~ ]
        then
            home_flag_dst=1
        fi
        dest=$i
    elif [ $ind -eq 5 ]
```

(continues on next page)

```

    then
        files="$i"
    else
        IP="$i"
    fi
    ind=$((ind+1))
done

source_way=$source
dest_way=$dest

# Xchange source and dest when Mode
if [ $mode -eq 1 -o $mode -eq 3 ]
then
    if [ $home_flag_src -eq 1 ]
    then
        source_way="/home/$USER"
    fi
    if [ $home_flag_dst -eq 1 ]
    then
        dest_way="/users/$user"
    fi
else
    if [ $home_flag_src -eq 1 ]
    then
        source_way="/users/$user"
    fi
    if [ $home_flag_dst -eq 1 ]
    then
        dest_way="/home/$USER"
    fi
fi

source_way=$source_way"/"
dest_way=$dest_way"/"

if [ $mode -eq 1 ]
then
    if [ `echo $dest_way | grep "home/" ` != "" ] 2> /dev/null
    then
        dest_way=`echo $dest_way | sed -e "s|/home/$USER|/users/$user|g"`
    fi
    scp $source_way$files $user@$IP:$dest_way
elif [ $mode -eq 2 ]
then
    if [ `echo $source_way | grep "home/" ` != "" ] 2> /dev/null
    then
        source_way=`echo $source_way | sed -e "s|/home/$USER|/users/$user|g"`
    fi
    scp $user@$IP:$source_way$files $dest_way

```

(continues on next page)

(continued from previous page)

```
elif [ $mode -eq 3 ]
then
    if [ `echo $dest_way | grep "home/"` != "" ] 2> /dev/null
    then
        dest_way=`echo $dest_way | sed -e "s|/home/$USER|/users/$user|g"`
    fi
    scp -r $source_way$files $user@$IP:$dest_way
elif [ $mode -eq 4 ]
then
    if [ `echo $source_way | grep "home/"` != "" ] 2> /dev/null
    then
        source_way=`echo $source_way | sed -e "s|/home/$USER|/users/$user|g"`
    fi
    scp -r $user@$IP:$source_way$files $dest_way
fi
```


SUDO-UPGRADE-ALL

5.1 Algorithm Sudo-upgrade-all

I get the application list of the system from the `rez='apt list -upgradable'` command. Once done, I browse the returned list to get the exact extensions and libraries name as list also. In the final loop, I upgrade each extension from thier name and the `apt upgrade $i` command.

5.2 Script Usage Sudo-upgrade-all

Called without arguments like that : `./sudo-upgrade-all.sh`

This script is used to upgrade all the present binaries libraries on a Unix system.

Please to use **if and only if** the Unnix system use the apt command (see also **sudo apt** command in Linux Manual).

5.3 Source Sudo-upgrade-all

```
#!/bin/bash

# Author : CABOS Matthieu
# Date : 31/08/2020

rez=`apt list --upgradable`
for i in $rez
do
    libname=`echo $i | grep "/" `
    if [ ! "$libname" = "" ]
    then
        lib=`echo $i | cut -d "/" -f1`
        lib_final="$lib_final"" "$lib"
        libname=""
    fi
done
for i in $lib_final
do
    apt upgrade $i
done
```

INDICES AND TABLES

- [genindex](#)
- [modindex](#)
- [search](#)