



Instituto Politécnico Nacional



Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Sistemas Operativos en Tiempo Real

Prof. Lamberto Maza Casas

Alumno:

Yáñez Cervantes Luis Felipe

Trabajo:

Guía para programa de alarma en MarteOS

Semestre 2020/1

Octubre 2019

Índice

Índice	2
Antes de la instalación de MarteOS.	2
Instalación de MarteOS.	3
Creación del programa de alarma	16
Creación de una live USB	22

Antes de la instalación de MarteOS.

Para poder hacer una instalación exitosa de este sistema operativo en tiempo real hay que tener en cuenta el año de la versión que vamos a instalar, esto por cualquier problema de compilación que pueda surgir al usar una versión distinta tanto del sistema operativo “base”, el cual usualmente es alguna versión de Linux en cualquiera de sus plataformas, ya sea Debian, Ubuntu, etc., como con la versión del SOTR que queremos instalar.

Para este caso tenemos que tanto MarteOS y GNAT (Que es un conocido compilador de lenguaje Ada, basado en la infraestructura de compilación de GCC) son versiones desarrolladas en el año 2017 y 2016 respectivamente. Por lo que es importante usar un sistema operativo base cuyo año de desarrollo sea 2016 o 2017, en otras palabras, un sistema operativo en el que estemos seguros que el compilador GNAT no tendrá ningún problema a la hora de instalarse. Esto con el fin de hacer una instalación más rápida y sencilla, no quiere decir que no pueda hacerse en un sistema operativo que haya sido desarrollado recientemente, sin embargo, para esto se debe tomar en cuenta lo que el compilador necesita para poder llevar a cabo una instalación exitosa.

Una vez que se tenga instalado el sistema operativo base, ya sea como un sistema operativo nativo o como una máquina virtual, tenemos que contar con los siguientes archivos:

- gnat-gpl-2016-x86_64-linux-bin.tar.gz
- marte_2.0_22Feb2017_src.tar.gz

El primer archivo podemos encontrarlo en el siguiente enlace:

<http://mirrors.cdn.adacore.com/art/5739cefdc7a447658e0b016b>

Mientras que el segundo archivo podemos encontrarlo en el siguiente URL:

https://marte.unican.es/marte/marte_2.0_22Feb2017_src.tar.gz

El primer archivo como es de suponerse es el compilador GNAT, mientras que el segundo es el Sistema operativo en tiempo real.

Instalación de MarteOS.

Una vez que se tiene instalado el sistema operativo base (Ubuntu 16.04), lo primero que tenemos que hacer es abrir nuestro gestor de archivos, en la pestaña “Carpeta personal”, crearemos una carpeta con el nombre “myapps”.

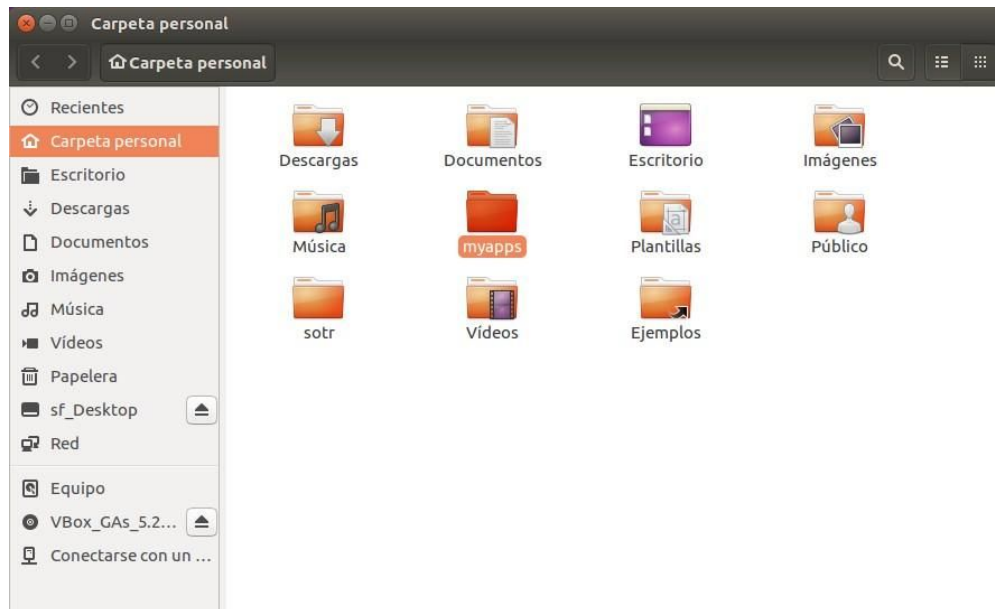


Ilustración 1 Creación de la carpeta myapps en el gestor de archivos.

Una vez creada esta carpeta debemos guardar en ella los archivos “gnat-gpl-2016-x86_64-linux-bin.tar.gz” y “marte_2.0_22Feb2017_src.tar.gz”, a su vez, dentro de la carpeta “myapps” crearemos una carpeta de nombre “gnat”.

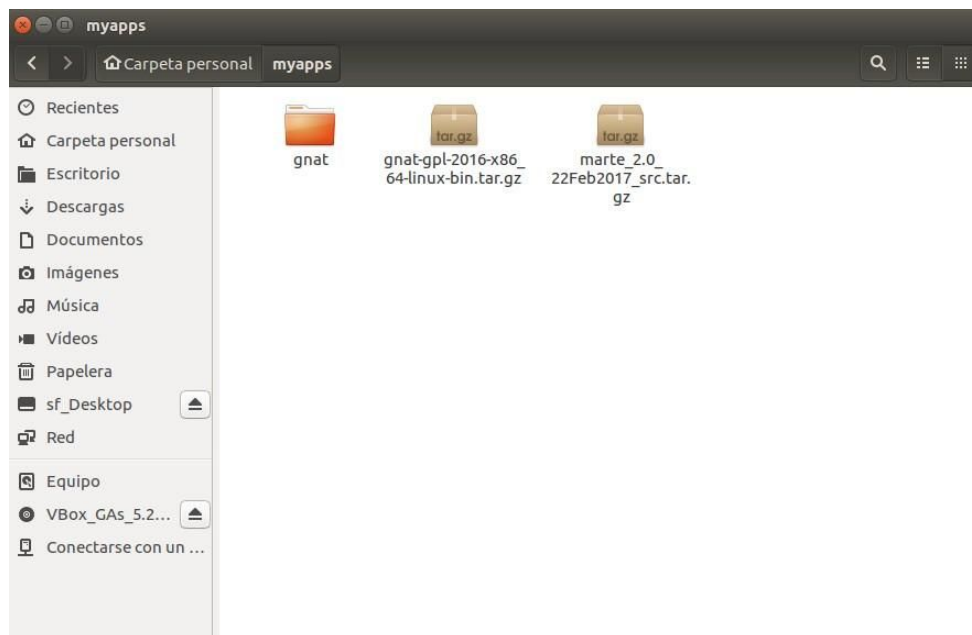


Ilustración 2 Creación de la carpeta gnat y ubicación de los archivos del compilador y sotr.

El siguiente paso es descomprimir los archivos que contienen el compilador GNAT así como el SOTR, comenzamos con el compilador GNAT, dentro de la carpeta myapps damos click derecho y abrimos un nuevo terminal, acto seguido ingresamos la siguiente instrucción “`tar xvf gnat-gpl-2016-x86_64-linux-bin.tar.gz`”, una vez terminado este proceso, procederemos a descomprimir el archivo contenedor del SOTR y eso lo haremos con la siguiente instrucción “`tar xvf marte_2.0_22Feb2017_src.tar.gz`”. Como se muestra a continuación.

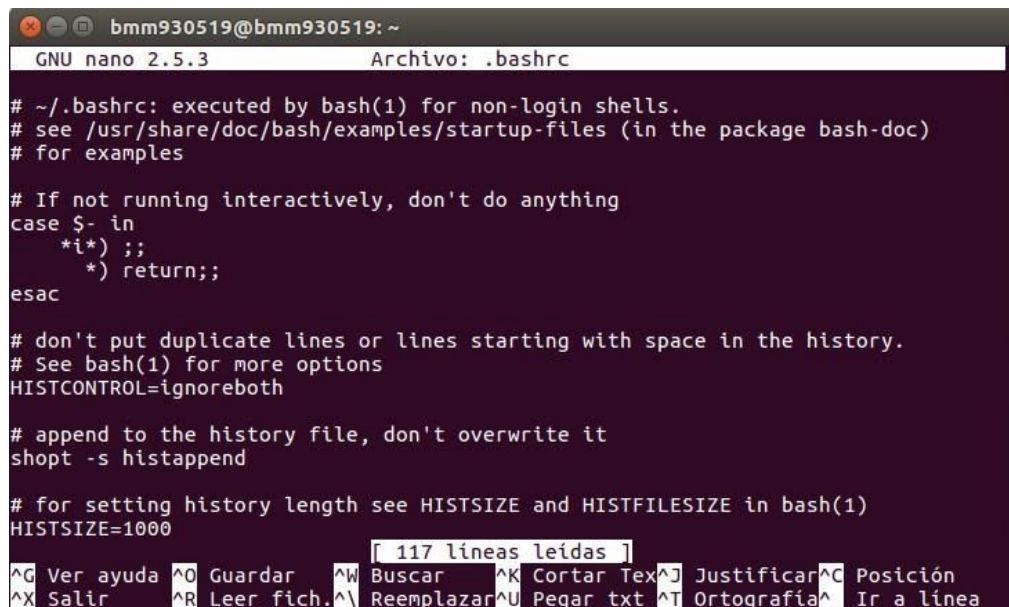
```
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/20-unhint-small-vera.conf
bmm930519@bmm930519:~/myapps$ tar xvf marte_2.0_22Feb2017_src.tar.gz
marTE_2.0_22Feb2017/
marTE_2.0_22Feb2017/objs/
marTE_2.0_22Feb2017/objs/linux_objs/
marTE_2.0_22Feb2017/objs/linux_lib_objs/
marTE_2.0_22Feb2017/objs/xtratum_objs/
marTE_2.0_22Feb2017/objs/x86_objs/
marTE_2.0_22Feb2017/objs/rpi_objs/
marTE_2.0_22Feb2017/tasks_inspector/
marTE_2.0_22Feb2017/tasks_inspector/tasks_inspector
marTE_2.0_22Feb2017/tasks_inspector/README
marTE_2.0_22Feb2017/tasks_inspector/get_sched_info.sh
marTE_2.0_22Feb2017/tasks_inspector/tasks_inspector.pl
marTE_2.0_22Feb2017/tests/
marTE_2.0_22Feb2017/tests/circular_memory_buffer/
marTE_2.0_22Feb2017/tests/circular_memory_buffer/Makefile
marTE_2.0_22Feb2017/tests/circular_memory_buffer/test_circular_memory_buffer.c
marTE_2.0_22Feb2017/tests/posix_extensions/
marTE_2.0_22Feb2017/tests/posix_extensions/test_interrupt_clock.c
marTE_2.0_22Feb2017/tests/posix_extensions/test_timed_handlers_for_group_clocks.c
marTE_2.0_22Feb2017/tests/posix_extensions/Makefile
marTE_2.0_22Feb2017/tests/posix_extensions/test_group_clock_timer.c
marTE_2.0_22Feb2017/tests/posix_extensions/test_interrupt_clock_2.c
marTE_2.0_22Feb2017/tests/posix_extensions/test_group_clock_timer2.c
marTE_2.0_22Feb2017/tests/posix_extensions/test_group_clock_timer3.c
marTE_2.0_22Feb2017/tests/posix_extensions/test_interrupt_clock_and_cpu_timers.c
marTE_2.0_22Feb2017/tests/time/
```

Ilustración 3 Descomprimiendo la carpeta contenedora del SOTR

Una vez que hemos descomprimido los archivos contenedores del compilador y el SOTR, lo siguiente es modificar el archivo `bashrc` para poder instalar el SOTR sin ningún problema, para ello haremos lo siguiente, en la misma terminal, será necesario salir a la raíz del sistema, esto lo lograremos con la instrucción “`cd`”, una vez ahí, ejecutaremos el comando “`nano .bashrc`”, como se muestra a continuación.

```
bmm930519@bmm930519: ~
marTE_2.0_22Feb2017/x86_arch/include/misc/load_loop.h
marTE_2.0_22Feb2017/x86_arch/include/misc/logger.h
marTE_2.0_22Feb2017/x86_arch/include/misc/timespec_operations.h
marTE_2.0_22Feb2017/x86_arch/include/misc/error_checks.h
marTE_2.0_22Feb2017/x86_arch/include/misc/time_measurement_hwtimer.h
marTE_2.0_22Feb2017/x86_arch/include/misc/circular_memory_buffer.h
marTE_2.0_22Feb2017/x86_arch/include/misc/load.h
marTE_2.0_22Feb2017/x86_arch/include/misc/generic_lists_order.h
marTE_2.0_22Feb2017/x86_arch/include/misc/time_measurement_posix.h
marTE_2.0_22Feb2017/x86_arch/include/misc/generic_lists_prio.h
marTE_2.0_22Feb2017/x86_arch/include/misc/freelist.h
marTE_2.0_22Feb2017/x86_arch/include/stddef.h
marTE_2.0_22Feb2017/x86_arch/include/stdio.h
marTE_2.0_22Feb2017/x86_arch/include/semaphore.h
marTE_2.0_22Feb2017/x86_arch/include/intr.h
marTE_2.0_22Feb2017/x86_arch/include/assert.h
marTE_2.0_22Feb2017/x86_arch/include/malloc.h
marTE_2.0_22Feb2017/x86_arch/include/stdbool.h
marTE_2.0_22Feb2017/x86_arch/include/pthread.h
marTE_2.0_22Feb2017/x86_arch/include/dirent.h
marTE_2.0_22Feb2017/x86_arch/include/stdlib.h
bmm930519@bmm930519:~/myapps$ cd
bmm930519@bmm930519:~$ nano .bashrc
```

Esta instrucción nos dará acceso al siguiente archivo.



```
bmm930519@bmm930519: ~
GNU nano 2.5.3 Archivo: .bashrc

# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
  *) ;;
  *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

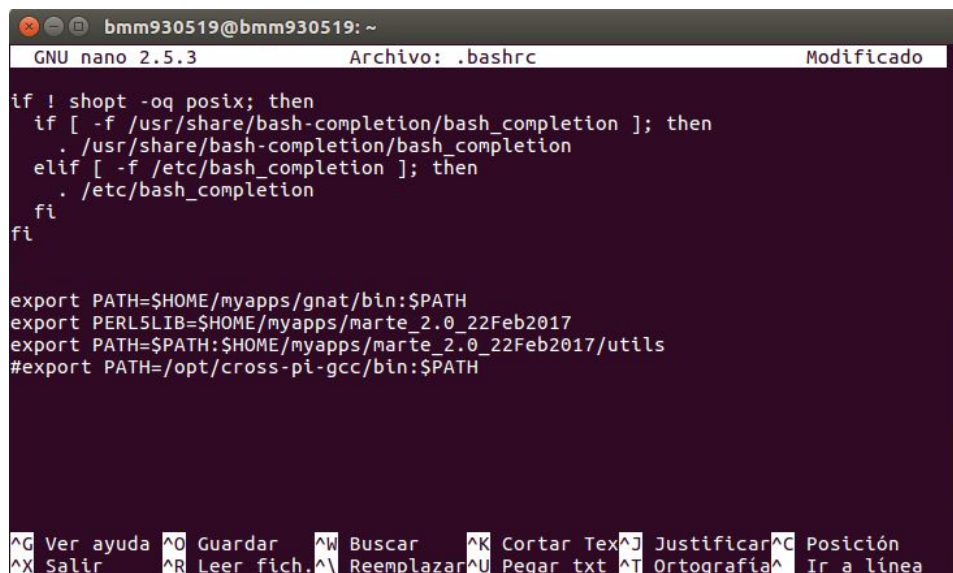
# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000

[ 117 líneas leídas ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Text ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

Dentro de este archivo, iremos al final y agregaremos las siguientes líneas:

- export PATH=\$HOME/myapps/gnat/bin:\$PATH
- export PERL5LIB=\$HOME/myapps/marte_2.0_22Feb2017
- export PATH=\$PATH:\$HOME/myapps/marte_2.0_22Feb2017/utils
- #export PATH=/opt/cross-pi-gcc/bin:\$PATH
- export DISPLAY=:0



```
bmm930519@bmm930519: ~
GNU nano 2.5.3 Archivo: .bashrc Modificado

if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export PATH=$HOME/myapps/gnat/bin:$PATH
export PERL5LIB=$HOME/myapps/marte_2.0_22Feb2017
export PATH=$PATH:$HOME/myapps/marte_2.0_22Feb2017/utils
#export PATH=/opt/cross-pi-gcc/bin:$PATH

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Text ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

Ilustración 4 Edición del archivo bash para la instalación de un SOTR

Finalmente, como lo indica el archivo, hay que guardar los cambios efectuados (CTRL+O) para después salir del archivo (CTRL+X)

Es importante que luego de hacer esto, cerrar la terminal en la que nos encontrábamos, esto para que los cambios que acabamos de realizar puedan efectuarse exitosamente.

Luego en la carpeta myapps nuevamente abrimos un nuevo terminal. Donde entraremos a la carpeta contenedora del compilador GNAT, esto lo lograremos con la instrucción `"cd gnat-gpl-2016- x86_64-linux-bin/"`, una vez situados en esa carpeta ejecutaremos la siguiente instrucción `"./doinstall"` y se nos presentará una ventana donde se nos describe la versión del compilador que vamos a instalar, para seguir con la instalación solo presionamos "ENTER".

A terminal window with a dark background and light-colored text. The title bar shows the user 'bmm930519' and the current directory '~/myapps/gnat-gpl-2016-x86_64-linux-bin'. The text inside the terminal reads: 'This script is provided to simplify the installation of the x86_64-pc-linux-gnu binary version of the GNAT GPL Edition Ada (2012, 2005, 95, and 83) environment maintained by AdaCore. For information on commercial support please contact sales@adacore.com. This script will ask you how you would like to configure GNAT. Confirmation is required before any write action is taken. Please press RETURN to continue.'

Ilustración 5 descripción del compilador que se va a instalar

Posteriormente, se nos preguntará la dirección de la carpeta donde queremos instalar el compilador.

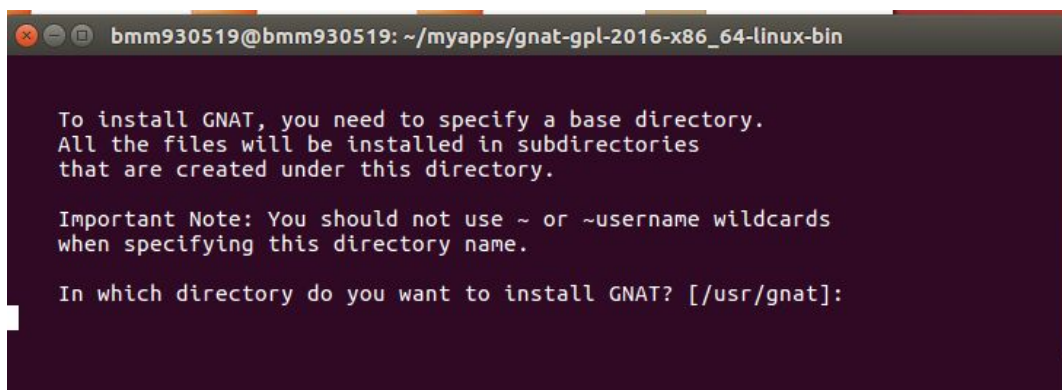
A terminal window with a dark background and light-colored text. The title bar shows the user 'bmm930519' and the current directory '~/myapps/gnat-gpl-2016-x86_64-linux-bin'. The text inside the terminal reads: 'To install GNAT, you need to specify a base directory. All the files will be installed in subdirectories that are created under this directory. Important Note: You should not use ~ or ~username wildcards when specifying this directory name. In which directory do you want to install GNAT? [/usr/gnat:]'. The prompt is followed by a blank line.

Ilustración 6 ¿Dónde se instalará el compilador?

Es aquí donde entra en juego la carpeta "gnat" que creamos en un inicio dentro de la carpeta "myapps". Para poder obtener la dirección de esta carpeta sólo es necesario seleccionar la carpeta "gnat" y dar click derecho sobre ella, seleccionar la opción "propiedades" y tendremos la vista de una ventana que nos dirá donde se encuentra nuestra carpeta con una etiqueta de nombre "Lugar".

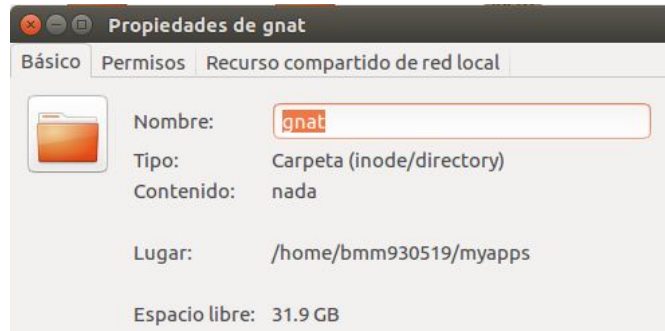


Ilustración 7 Propiedades de la carpeta "gnat"

Para este caso en particular, la carpeta de interés, tiene lugar dentro de nuestra máquina virtual en la dirección “/home/bmm930519/myapps”, para poder ingresar la dirección correctamente a la ventana de la ilustración 6, agregaremos el directorio “gnat”, entonces la cadena que ingresamos, quedará de la siguiente manera “/home/bmm930519/myapps/gnat”, con esto estaremos indicando exactamente donde queremos que se instale el compilador.

```
bmm930519@bmm930519: ~/myapps/gnat-gpl-2016-x86_64-linux-bin

To install GNAT, you need to specify a base directory.
All the files will be installed in subdirectories
that are created under this directory.

Important Note: You should not use ~ or ~username wildcards
when specifying this directory name.

In which directory do you want to install GNAT? [/usr/gnat]:
/home/bmm930519/myapps/gnat
```

Ilustración 8 Indicación de la carpeta en la cual se instalará el compilador

Para proseguir solo oprimimos la tecla “ENTER”. Se nos preguntará si la ruta de instalación es la correcta, a lo que solo debemos oprimir “Y” y luego “ENTER”.

```
bmm930519@bmm930519: ~/myapps/gnat-gpl-2016-x86_64-linux-bin

To install GNAT, you need to specify a base directory.
All the files will be installed in subdirectories
that are created under this directory.

Important Note: You should not use ~ or ~username wildcards
when specifying this directory name.

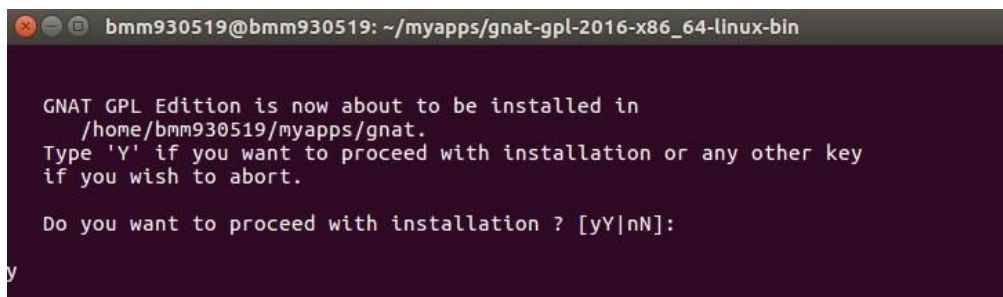
In which directory do you want to install GNAT? [/usr/gnat]:
/home/bmm930519/myapps/gnat

The GNAT GPL Edition installation directory will be:
/home/bmm930519/myapps/gnat
Is this correct ? Type 'Y' if so, otherwise type 'N' and you'll
be prompted for another directory name.

Do you want to continue ? [yY|nN]:
Y
```

Ilustración 9 Confirmación de la ruta de instalación

Acto seguido se nos preguntará si queremos proceder con la instalación, y haremos lo mismo que anteriormente presionamos la tecla “Y” y seguido de eso la tecla “ENTER”.

A terminal window with a dark background and light text. The title bar shows the user 'bmm930519' and the directory '~/myapps/gnat-gpl-2016-x86_64-linux-bin'. The text inside the terminal reads: 'GNAT GPL Edition is now about to be installed in /home/bmm930519/myapps/gnat. Type 'Y' if you want to proceed with installation or any other key if you wish to abort. Do you want to proceed with installation ? [yY|nN]:'. The letter 'y' is visible at the start of a new line, indicating the user's response to the prompt.

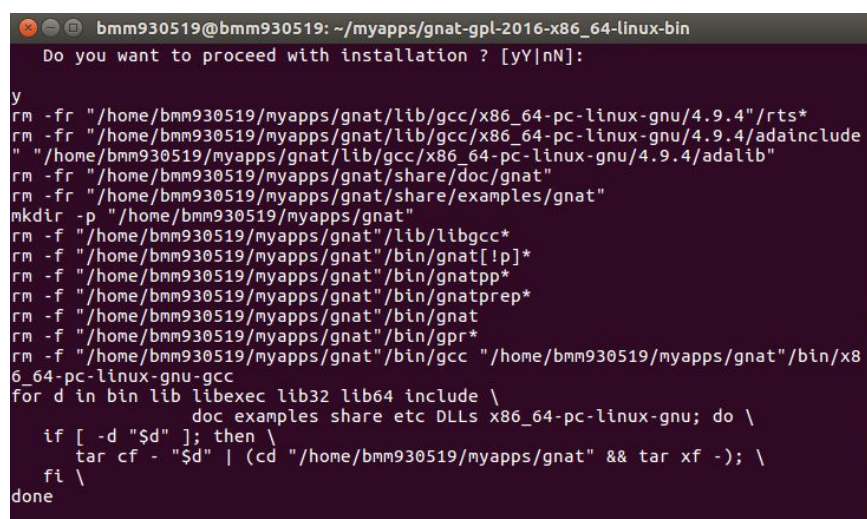
```
bmm930519@bmm930519: ~/myapps/gnat-gpl-2016-x86_64-linux-bin

GNAT GPL Edition is now about to be installed in
/home/bmm930519/myapps/gnat.
Type 'Y' if you want to proceed with installation or any other key
if you wish to abort.

Do you want to proceed with installation ? [yY|nN]:
y
```

Ilustración 10 Confirmación de instalación

Visualizamos una ventana como la siguiente, señal de que el proceso de instalación ha comenzado.

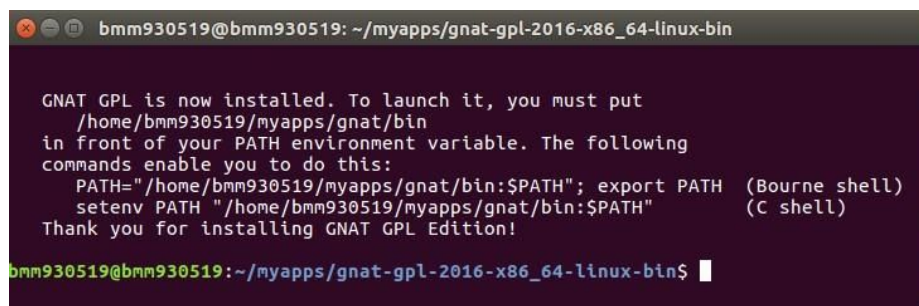
A terminal window showing the execution of the installation script. The title bar is the same as in the previous image. The text shows the prompt 'Do you want to proceed with installation ? [yY|nN]:' followed by the user input 'y'. Below this, a series of commands are executed to create directories and copy files: 'rm -fr' for removing old versions and 'mkdir -p' for creating the new directory structure. The commands are: 'rm -fr "/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4/rts*', 'rm -fr "/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4/adainclude', 'rm -fr "/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4/adalib', 'rm -fr "/home/bmm930519/myapps/gnat/share/doc/gnat', 'rm -fr "/home/bmm930519/myapps/gnat/share/examples/gnat', 'mkdir -p "/home/bmm930519/myapps/gnat', 'rm -f "/home/bmm930519/myapps/gnat/lib/libgcc*', 'rm -f "/home/bmm930519/myapps/gnat/bin/gnat[!p]*', 'rm -f "/home/bmm930519/myapps/gnat/bin/gnatpp*', 'rm -f "/home/bmm930519/myapps/gnat/bin/gnatprep*', 'rm -f "/home/bmm930519/myapps/gnat/bin/gnat', 'rm -f "/home/bmm930519/myapps/gnat/bin/gpr*', 'rm -f "/home/bmm930519/myapps/gnat/bin/gcc "/home/bmm930519/myapps/gnat/bin/x86_64-pc-linux-gnu-gcc', 'for d in bin lib libexec lib32 lib64 include \', 'do \', 'if [-d "\$d"]; then \', 'tar cf - "\$d" | (cd "/home/bmm930519/myapps/gnat" && tar xf -); \', 'fi \', 'done'.

```
bmm930519@bmm930519: ~/myapps/gnat-gpl-2016-x86_64-linux-bin

Do you want to proceed with installation ? [yY|nN]:
y
rm -fr "/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4/rts*
rm -fr "/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4/adainclude
"/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4/adalib"
rm -fr "/home/bmm930519/myapps/gnat/share/doc/gnat"
rm -fr "/home/bmm930519/myapps/gnat/share/examples/gnat"
mkdir -p "/home/bmm930519/myapps/gnat"
rm -f "/home/bmm930519/myapps/gnat/lib/libgcc*
rm -f "/home/bmm930519/myapps/gnat/bin/gnat[!p]*
rm -f "/home/bmm930519/myapps/gnat/bin/gnatpp*
rm -f "/home/bmm930519/myapps/gnat/bin/gnatprep*
rm -f "/home/bmm930519/myapps/gnat/bin/gnat
rm -f "/home/bmm930519/myapps/gnat/bin/gpr*
rm -f "/home/bmm930519/myapps/gnat/bin/gcc "/home/bmm930519/myapps/gnat/bin/x8
6_64-pc-linux-gnu-gcc
for d in bin lib libexec lib32 lib64 include \
do \
    doc examples share etc DLLs x86_64-pc-linux-gnu; do \
        if [ -d "$d" ]; then \
            tar cf - "$d" | (cd "/home/bmm930519/myapps/gnat" && tar xf -); \
        fi \
    done
```

Ilustración 11 Inicio de la instalación del compilador GNA

El proceso anterior puede demorar varios minutos, por lo que es importante no cerrar la ventana hasta que podamos visualizar lo siguiente, señal de que el proceso ha culminado.

A terminal window showing the final steps of the installation. The title bar is the same. The text reads: 'GNAT GPL is now installed. To launch it, you must put /home/bmm930519/myapps/gnat/bin in front of your PATH environment variable. The following commands enable you to do this: PATH="/home/bmm930519/myapps/gnat/bin:\$PATH"; export PATH (Bourne shell) setenv PATH "/home/bmm930519/myapps/gnat/bin:\$PATH" (C shell) Thank you for installing GNAT GPL Edition!'. The prompt 'bmm930519@bmm930519:~/myapps/gnat-gpl-2016-x86_64-linux-bin\$' is visible at the bottom.

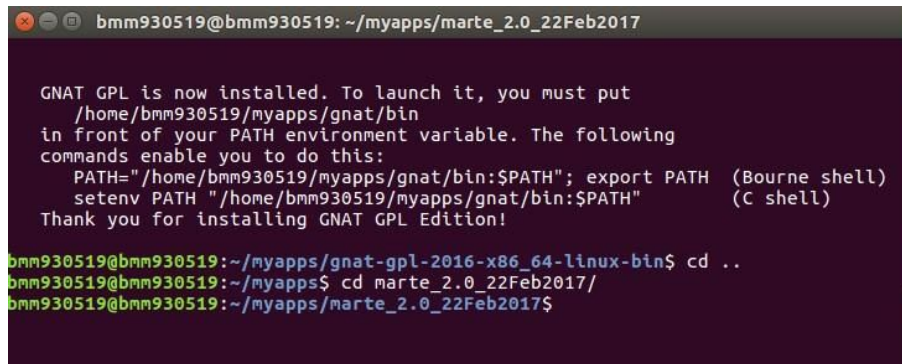
```
bmm930519@bmm930519: ~/myapps/gnat-gpl-2016-x86_64-linux-bin

GNAT GPL is now installed. To launch it, you must put
/home/bmm930519/myapps/gnat/bin
in front of your PATH environment variable. The following
commands enable you to do this:
PATH="/home/bmm930519/myapps/gnat/bin:$PATH"; export PATH (Bourne shell)
setenv PATH "/home/bmm930519/myapps/gnat/bin:$PATH" (C shell)
Thank you for installing GNAT GPL Edition!

bmm930519@bmm930519:~/myapps/gnat-gpl-2016-x86_64-linux-bin$
```

Ilustración 12 Finalización de la instalación del compilador

Lo siguiente es salir a la carpeta “myapps” y acceder a la carpeta “marte_2.0_22Feb2017”, para lograr eso, ejecutaremos la instrucción “cd ..” y luego de ello, ejecutaremos la instrucción “cd marte_2.0_22Feb2017/”.



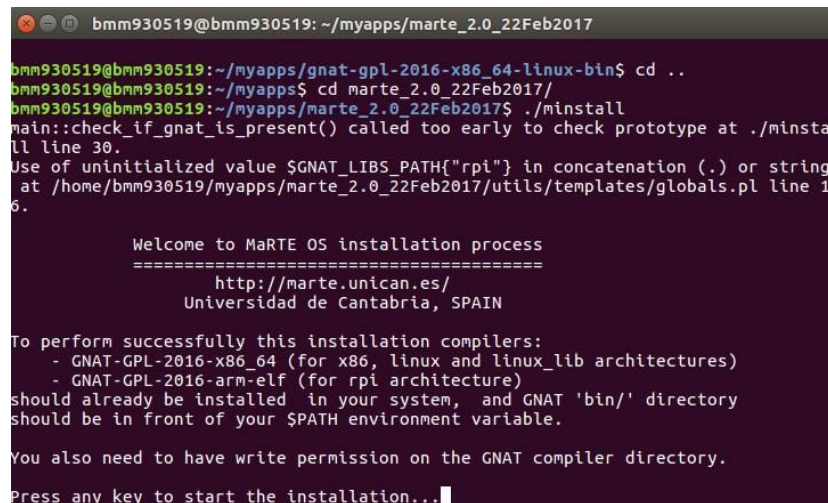
```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017

GNAT GPL is now installed. To launch it, you must put
/home/bmm930519/myapps/gnat/bin
in front of your PATH environment variable. The following
commands enable you to do this:
  PATH="/home/bmm930519/myapps/gnat/bin:$PATH"; export PATH (Bourne shell)
  setenv PATH "/home/bmm930519/myapps/gnat/bin:$PATH"      (C shell)
Thank you for installing GNAT GPL Edition!

bmm930519@bmm930519:~/myapps/gnat-gpl-2016-x86_64-linux-bin$ cd ..
bmm930519@bmm930519:~/myapps$ cd marte_2.0_22Feb2017/
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017$
```

Ilustración 13 Acceso a la carpeta del SOTR

Una vez ahí procederemos a instalar el SOTR, para lo cual ejecutaremos la instrucción “./minstall”, con lo que visualizamos lo siguiente.



```
bmm930519@bmm930519:~/myapps/gnat-gpl-2016-x86_64-linux-bin$ cd ..
bmm930519@bmm930519:~/myapps$ cd marte_2.0_22Feb2017/
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017$ ./minstall
main::check_if_gnat_is_present() called too early to check prototype at ./minsta
ll line 30.
Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string
at /home/bmm930519/myapps/marte_2.0_22Feb2017/Utils/templates/globals.pl line 1
6.

Welcome to MaRTE OS installation process
=====
http://marte.unican.es/
Universidad de Cantabria, SPAIN

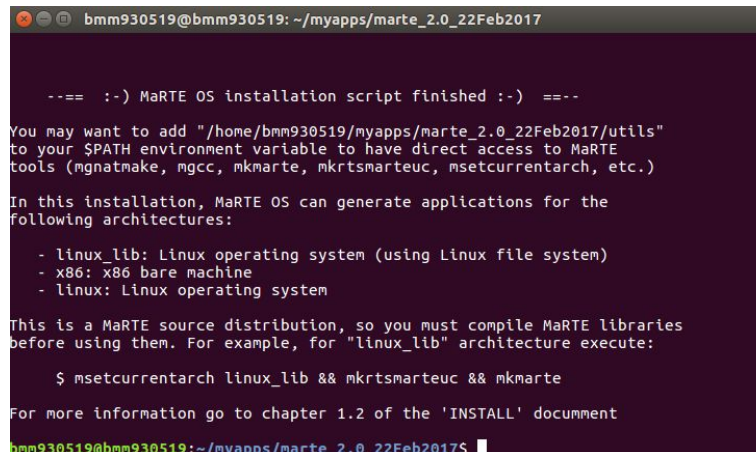
To perform successfully this installation compilers:
- GNAT-GPL-2016-x86_64 (for x86, linux and linux_lib architectures)
- GNAT-GPL-2016-arm-elf (for rpi architecture)
should already be installed in your system, and GNAT 'bin/' directory
should be in front of your $PATH environment variable.

You also need to have write permission on the GNAT compiler directory.

Press any key to start the installation...
```

Ilustración 14 Presentación del SOTR a instalar

Para continuar presionamos la tecla “ENTER”. Y a diferencia del compilador este proceso es bastante rápido, por lo que pronto visualizamos lo siguiente, en señal de que el proceso ha terminado exitosamente.



```
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017

--== :-) MaRTE OS installation script finished :-) ==--

You may want to add "/home/bmm930519/myapps/marte_2.0_22Feb2017/Utils"
to your $PATH environment variable to have direct access to MaRTE
tools (mgatmake, mgcc, mkmmarte, mkrtsmarteuc, msetcurrentarch, etc.)

In this installation, MaRTE OS can generate applications for the
following architectures:

- linux_lib: Linux operating system (using Linux file system)
- x86: x86 bare machine
- linux: Linux operating system

This is a MaRTE source distribution, so you must compile MaRTE libraries
before using them. For example, for "linux_lib" architecture execute:

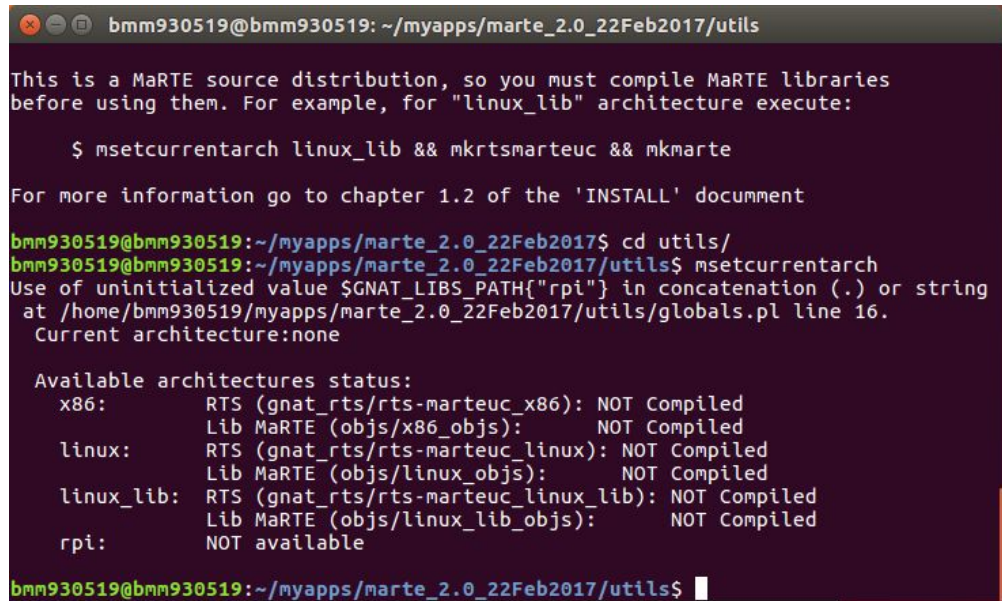
$ msetcurrentarch linux_lib && mkrtsmarteuc && mkmmarte

For more information go to chapter 1.2 of the 'INSTALL' document

bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017$
```

Ilustración 15 Finalización de la instalación del SOTR

Estamos cerca de terminar la instalación, lo siguientes es acceder a la carpeta “utils”, lo que lograremos con la siguiente instrucción “cd utils/” ya que esta carpeta esta dentro de la carpeta “marte_2.0_22Feb2017”, una vez estando dentro de la carpeta utils, procederemos a definir la arquitectura sobre la que trabajará nuestro SOTR, para ello ingresaremos la siguiente instrucción a la terminal “msetcurrentarch”, y podremos ver que tenemos varias arquitecturas para elegir.



```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/utils
This is a MaRTE source distribution, so you must compile MaRTE libraries
before using them. For example, for "linux_lib" architecture execute:

$ msetcurrentarch linux_lib && mkrtsmarteuc && mkmarte

For more information go to chapter 1.2 of the 'INSTALL' document

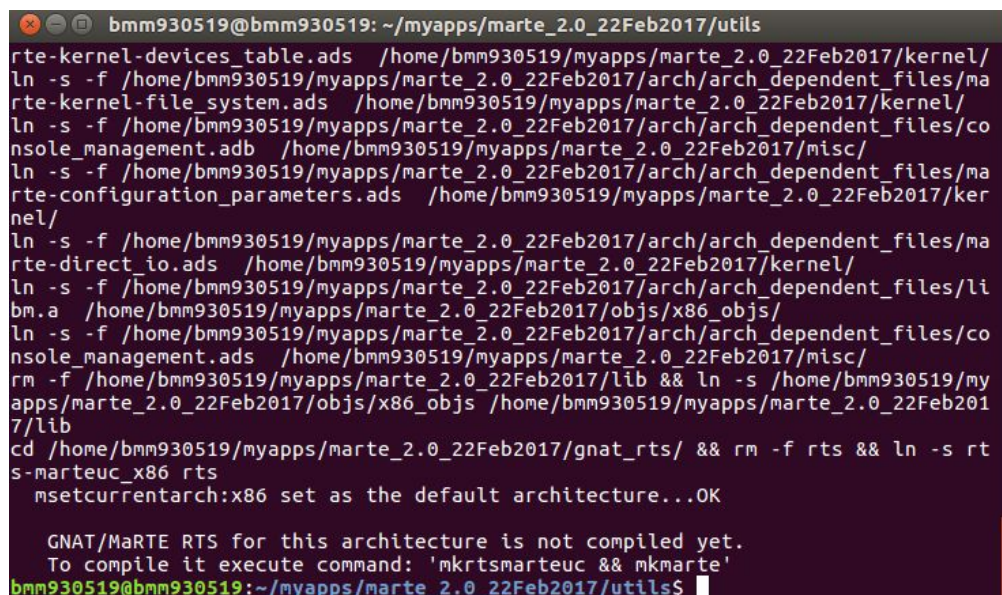
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017$ cd utils/
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/utils$ msetcurrentarch
Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string
at /home/bmm930519/myapps/marte_2.0_22Feb2017/utils/globals.pl line 16.
Current architecture:none

Available architectures status:
x86:      RTS (gnat_rts/rts-marteuc_x86): NOT Compiled
          Lib MaRTE (objs/x86_objs):      NOT Compiled
linux:    RTS (gnat_rts/rts-marteuc_linux): NOT Compiled
          Lib MaRTE (objs/linux_objs):     NOT Compiled
linux_lib: RTS (gnat_rts/rts-marteuc_linux_lib): NOT Compiled
          Lib MaRTE (objs/linux_lib_objs): NOT Compiled
rpi:      NOT available

bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/utils$
```

Ilustración 16 Arquitecturas a elegir para el SOTR

Para esta instalación, elegimos la arquitectura x86, por lo que ingresamos la siguiente instrucción en la terminal “msetcurrentarch x86 i386”, una vez ejecutada, visualizamos lo siguiente.



```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/utils
rte-kernel-devices_table.ads /home/bmm930519/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/bmm930519/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/ma
rte-kernel-file_system.ads /home/bmm930519/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/bmm930519/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/co
nsole_management.adb /home/bmm930519/myapps/marte_2.0_22Feb2017/misc/
ln -s -f /home/bmm930519/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/ma
rte-configuration_parameters.ads /home/bmm930519/myapps/marte_2.0_22Feb2017/ker
nel/
ln -s -f /home/bmm930519/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/ma
rte-direct_io.ads /home/bmm930519/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/bmm930519/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/li
bn.a /home/bmm930519/myapps/marte_2.0_22Feb2017/objs/x86_objs/
ln -s -f /home/bmm930519/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/co
nsole_management.ads /home/bmm930519/myapps/marte_2.0_22Feb2017/misc/
rm -f /home/bmm930519/myapps/marte_2.0_22Feb2017/lib && ln -s /home/bmm930519/my
apps/marte_2.0_22Feb2017/objs/x86_objs /home/bmm930519/myapps/marte_2.0_22Feb201
7/lib
cd /home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/ && rm -f rts && ln -s rt
s-marteuc_x86 rts
msetcurrentarch:x86 set as the default architecture...OK

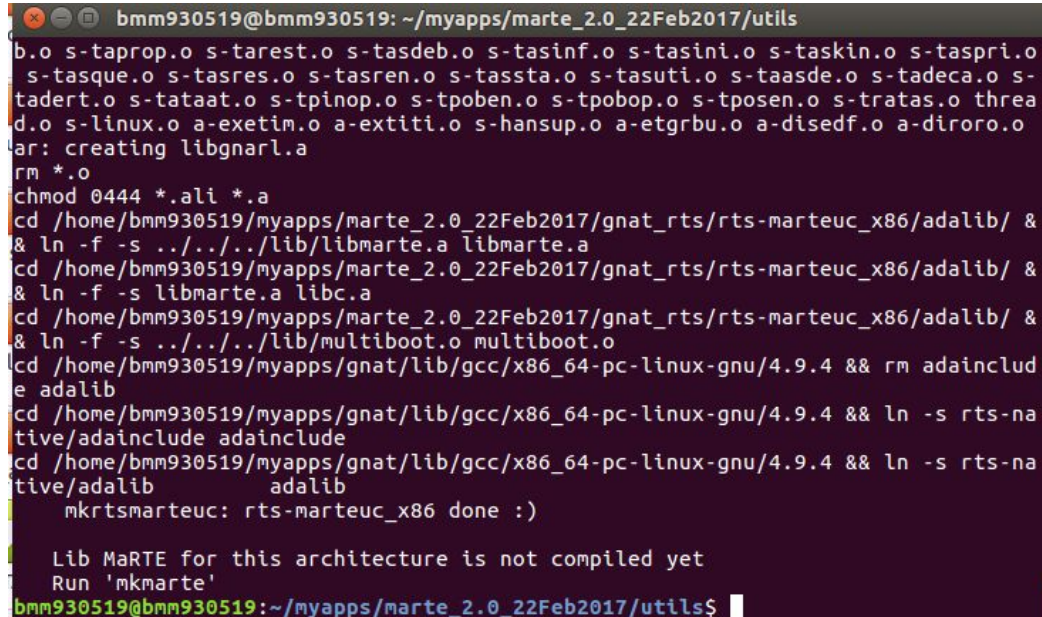
GNAT/MaRTE RTS for this architecture is not compiled yet.
To compile it execute command: 'mkrtsmarteuc && mkmarte'
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/utils$
```

Ilustración 17 Selección de la arquitectura para el SOTR

Como podemos ver, la arquitectura que elegimos ha sido puesta como default para el funcionamiento de nuestro SOTR.

Para continuar, seguiremos las indicaciones que vienen hasta abajo, que es ejecutar las instrucciones “mkrtsmarteuc” y posteriormente “mkmarte”, es importante que sea en ese orden, de lo contrario visualizamos una ventana de error.

La primera instrucción tardará un poco en ejecutarse pero es importante no cerrar la ventana de la terminal. Una vez que visualicemos lo siguiente, podemos continuar a ejecutar la instrucción “mkmarte”.

A terminal window with a dark background and light text. The prompt is 'bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/utils'. The output shows a list of object files being processed, the creation of 'libgnarl.a', and various 'ln' and 'rm' commands for setting up the library path. It ends with 'mkrtsmarteuc: rts-marteuc_x86 done :)' and a message that the library is not yet compiled, suggesting to run 'mkmarte'.

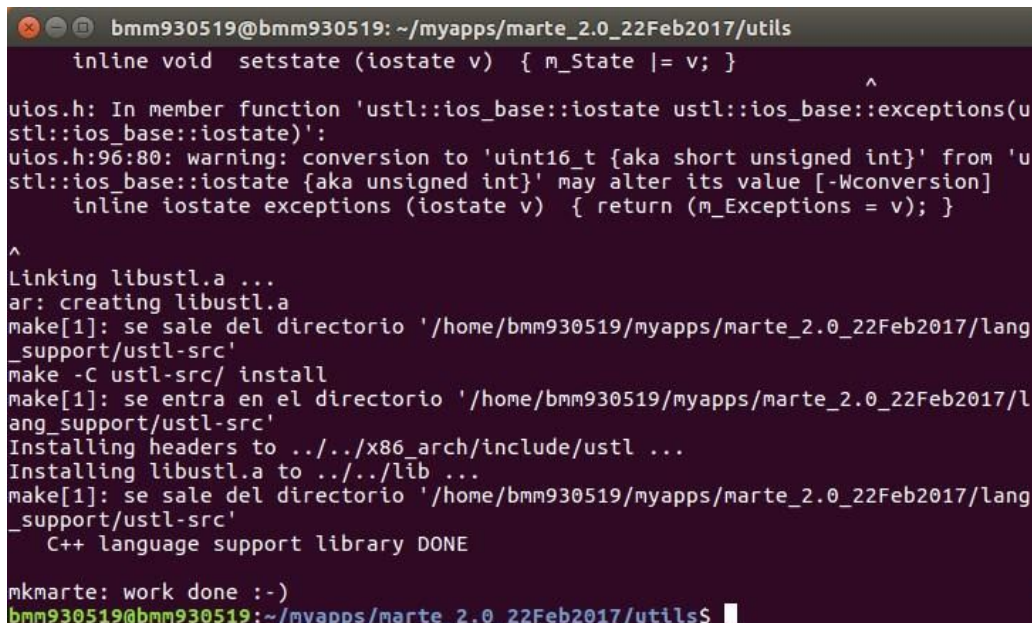
```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/utils
b.o s-taprop.o s-tarest.o s-tasdeb.o s-tasinf.o s-tasini.o s-taskin.o s-taspri.o
s-tasque.o s-tasres.o s-tasren.o s-tassta.o s-tasuti.o s-tasde.o s-tadeca.o s-
tadert.o s-tataat.o s-tpinop.o s-tpoben.o s-tpobop.o s-tposen.o s-tratas.o threa
d.o s-linux.o a-exetim.o a-extiti.o s-hansup.o a-etgrbu.o a-disedf.o a-diroro.o
ar: creating libgnarl.a
rm *.o
chmod 0444 *.ali *.a
cd /home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ &
& ln -f -s ../../../../lib/libmarte.a libmarte.a
cd /home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ &
& ln -f -s libmarte.a libc.a
cd /home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ &
& ln -f -s ../../../../lib/multiboot.o multiboot.o
cd /home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && rm adainclud
e adalib
cd /home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && ln -s rts-na
tive/adainclude adainclude
cd /home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && ln -s rts-na
tive/adalib adalib
mkrtsmarteuc: rts-marteuc_x86 done :)

Lib MaRTE for this architecture is not compiled yet
Run 'mkmarte'
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/utils$
```

Ilustración 18 ejecución de la instrucción mkrtsmarteuc

Y de hecho como podemos ver, al final podemos ver que se nos invita a ejecutar la instrucción “mkmarte” es por ello que es importante respetar el orden de ejecución de estas instrucciones.

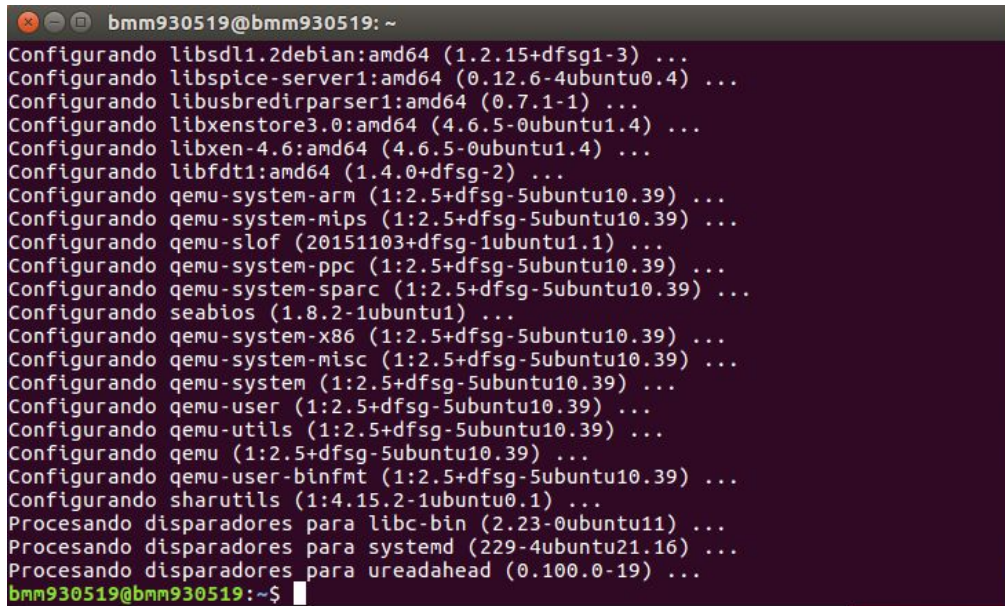
Una vez que se ejecute la instrucción “mkmarte” podremos visualizar lo siguiente.

A terminal window showing the output of the 'mkmarte' command. It displays the linking of 'libustl.a', the creation of 'libustl.a', and the installation of headers and the library to the system paths. The final output is 'mkmarte: work done :-)' and the prompt returns to the shell.

```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/utils
inline void setstate (iostate v) { m_State |= v; }
^
uos.h: In member function 'ustl::ios_base::iostate ustl::ios_base::exceptions(u
stl::ios_base::iostate)':
uos.h:96:80: warning: conversion to 'uint16_t {aka short unsigned int}' from 'u
stl::ios_base::iostate {aka unsigned int}' may alter its value [-Wconversion]
inline iostate exceptions (iostate v) { return (m_Exceptions = v); }
^
Linking libustl.a ...
ar: creating libustl.a
make[1]: se sale del directorio '/home/bmm930519/myapps/marte_2.0_22Feb2017/lang
_support/ustl-src'
make -C ustl-src/ install
make[1]: se entra en el directorio '/home/bmm930519/myapps/marte_2.0_22Feb2017/l
ang_support/ustl-src'
Installing headers to ../../x86_arch/include/ustl ...
Installing libustl.a to ../../lib ...
make[1]: se sale del directorio '/home/bmm930519/myapps/marte_2.0_22Feb2017/lang
_support/ustl-src'
C++ language support library DONE
mkmarte: work done :-)
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/utils$
```

Ilustración 19 ejecución de la instrucción "mkmarte"

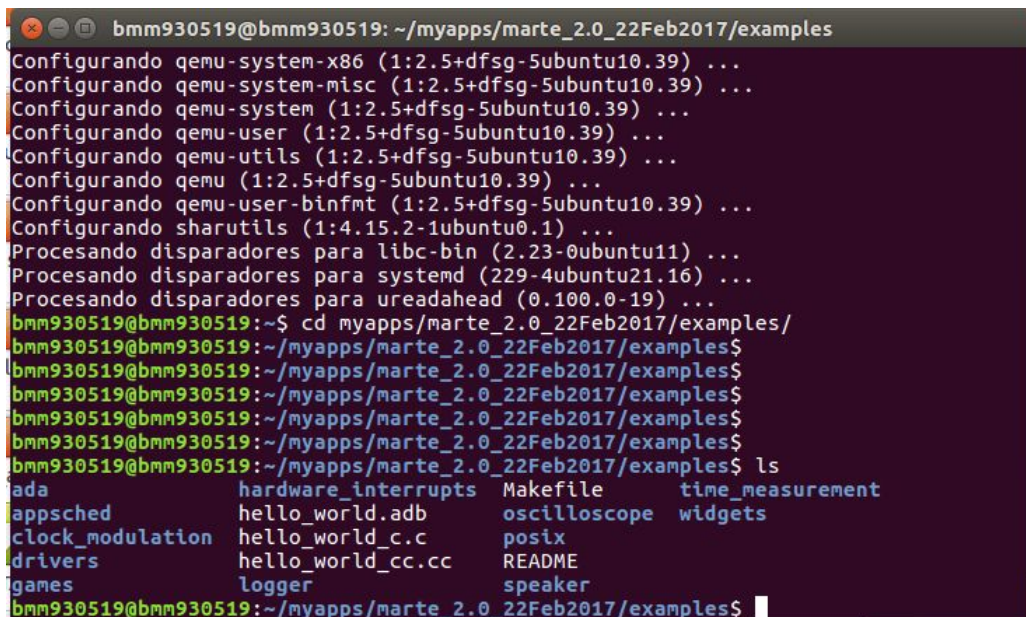
Hasta este punto, terminamos con la instalación del SOTR MaRTE OS, lo siguiente es probar que realmente funcione, para poder hacerlo, requerimos de la aplicación QEMU, si no la tenemos instalada, lo siguiente es ejecutar la instrucción “cd” para salir a la carpeta raíz y una vez ahí, ejecutar la instrucción “sudo apt-get install qemu”, acto seguido la terminal nos pedirá la contraseña de usuario, se ingresa y se procede con la instalación de esta aplicación. Este proceso tardará unos minutos. Para saber que la instalación fue exitosa, deberemos visualizar lo siguiente.



```
bmm930519@bmm930519: ~  
Configurando libsd1.2debian:amd64 (1.2.15+dfsg1-3) ...  
Configurando libspice-server1:amd64 (0.12.6-4ubuntu0.4) ...  
Configurando libusbredirparser1:amd64 (0.7.1-1) ...  
Configurando libxenstore3.0:amd64 (4.6.5-0ubuntu1.4) ...  
Configurando libxen-4.6:amd64 (4.6.5-0ubuntu1.4) ...  
Configurando libfdt1:amd64 (1.4.0+dfsg-2) ...  
Configurando qemu-system-arm (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-system-mips (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-slof (20151103+dfsg-1ubuntu1.1) ...  
Configurando qemu-system-ppc (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-system-sparc (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando seabios (1.8.2-1ubuntu1) ...  
Configurando qemu-system-x86 (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-system-misc (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-system (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-user (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-utils (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-user-binfmt (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando sharutils (1:4.15.2-1ubuntu0.1) ...  
Procesando disparadores para libc-bin (2.23-0ubuntu11) ...  
Procesando disparadores para systemd (229-4ubuntu21.16) ...  
Procesando disparadores para ureadahead (0.100.0-19) ...  
bmm930519@bmm930519:~$
```

Ilustración 20 instalación de la aplicación qemu

Lo siguiente es ir a la carpeta de ejemplos de marte, para lo que ejecutaremos la siguiente instrucción “cd myapps/marte_2.0_22Feb2017/examples”. Una vez ahí ejecutamos la aplicación “ls” para visualizar los archivos que hay dentro de la carpeta de ejemplos.



```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/examples  
Configurando qemu-system-x86 (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-system-misc (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-system (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-user (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-utils (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando qemu-user-binfmt (1:2.5+dfsg-5ubuntu10.39) ...  
Configurando sharutils (1:4.15.2-1ubuntu0.1) ...  
Procesando disparadores para libc-bin (2.23-0ubuntu11) ...  
Procesando disparadores para systemd (229-4ubuntu21.16) ...  
Procesando disparadores para ureadahead (0.100.0-19) ...  
bmm930519@bmm930519:~$ cd myapps/marte_2.0_22Feb2017/examples/  
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$  
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$  
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$  
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$  
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$  
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ ls  
ada hardware_interrupts Makefile time_measurement  
appsched hello_world.adb oscilloscope widgets  
clock_modulation hello_world_c.c posix  
drivers hello_world_cc.cc README  
games logger speaker  
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$
```

Ilustración 21 Archivos ejemplo de MaRTE OS

Como podemos ver, hay archivos para C, C++ y Ada, nosotros trabajaremos con el archivo “hello_world_c.c”, por consiguiente ejecutaremos la siguiente instrucción “mgcc hello_world_c.c” con esto habremos compilado este archivo y una forma de verificar que la compilación fue exitosa es revisar que se haya creado un archivo de nombre “a.out”, volvemos a ejecutar la instrucción “ls” para verificar que en efecto esto pasó.

```

bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/examples
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ ls
ada          hardware_interrupts  Makefile        time_measurement
appsched     hello_world.adb      oscilloscope    widgets
clock_modulation hello_world_c.c      posix
drivers      hello_world_cc.cc    README
games        logger               speaker
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ mgcc hello_world_c.c
Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string
at /home/bmm930519/myapps/marte_2.0_22Feb2017/utils/globals.pl line 16.
gcc -nostdinc -I/home/bmm930519/myapps/marte_2.0_22Feb2017/arch/include hello_w
orld_c.c -m32 -march=i686 /home/bmm930519/myapps/marte_2.0_22Feb2017/arch/
call_main/wrapper_main.c.o -WL,-T,/home/bmm930519/myapps/marte_2.0_22Feb2017/u
tils/linker.lds -static -nostartfiles -L/home/bmm930519/myapps/marte_2.0_22Feb20
17/lib -L/home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/rts/adalib -L/home/
bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnat -lgcc -l
gmarte -lgcc_sjlj
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ ls
ada          games               logger              speaker
a.out        hardware_interrupts Makefile            time_measurement
appsched     hello_world.adb     oscilloscope        widgets
clock_modulation hello_world_c.c      posix
drivers      hello_world_cc.cc   README
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$

```

Ilustración 22 verificación de la creación del archivo a.out

Lo siguiente es ejecutar la siguiente instrucción “mgcc hello_world_c.c -o mprogram” con lo que crearemos un archivo de nombre “mprogram” asociado a la compilación de nuestro archivo “hello_world_c.c”.

Luego de esto crearemos un archivo ejecutable asociado al archivo C, esto lo lograremos con la siguiente instrucción “make hello_world_c.exe”.

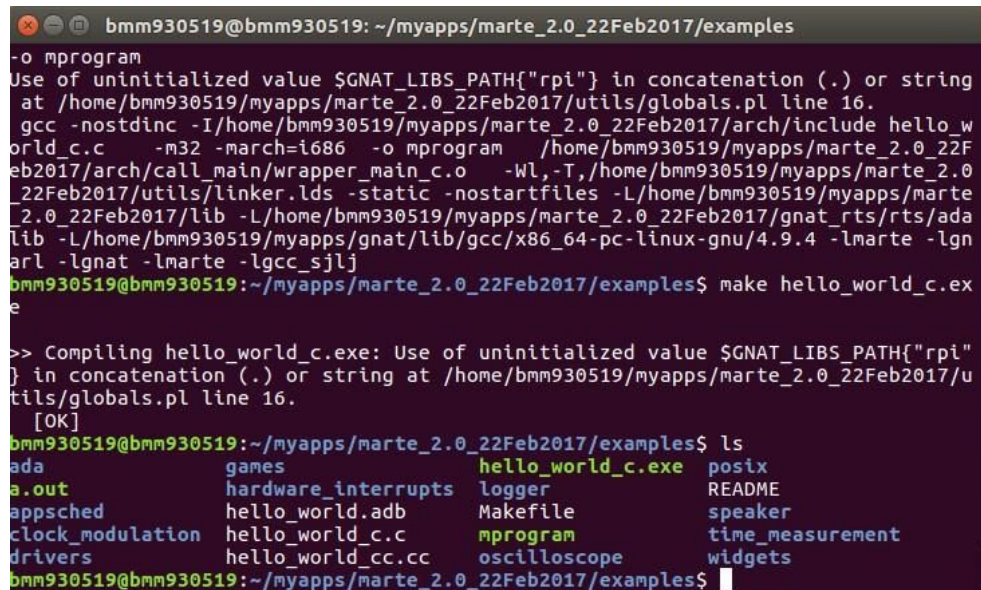
```

bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/examples
ada          games               logger              speaker
a.out        hardware_interrupts Makefile            time_measurement
appsched     hello_world.adb     oscilloscope        widgets
clock_modulation hello_world_c.c      posix
drivers      hello_world_cc.cc   README
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ mgcc hello_world_c.c
-o mprogram
Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string
at /home/bmm930519/myapps/marte_2.0_22Feb2017/utils/globals.pl line 16.
gcc -nostdinc -I/home/bmm930519/myapps/marte_2.0_22Feb2017/arch/include hello_w
orld_c.c -m32 -march=i686 -o mprogram /home/bmm930519/myapps/marte_2.0_22F
eb2017/arch/call_main/wrapper_main.c.o -WL,-T,/home/bmm930519/myapps/marte_2.0
22Feb2017/tils/linker.lds -static -nostartfiles -L/home/bmm930519/myapps/marte
2.0_22Feb2017/lib -L/home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/rts/ada
lib -L/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgn
arl -lgnat -lmarte -lgcc_sjlj
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ make hello_world_c.ex
e
>> Compiling hello_world_c.exe: Use of uninitialized value $GNAT_LIBS_PATH{"rpi"}
in concatenation (.) or string at /home/bmm930519/myapps/marte_2.0_22Feb2017/u
tils/globals.pl line 16.
[OK]
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$

```

Ilustración 23 Creación de un archivo ejecutable

Para verificar la creación de estos dos últimos archivos, volvemos a ejecutar la instrucción “ls”.

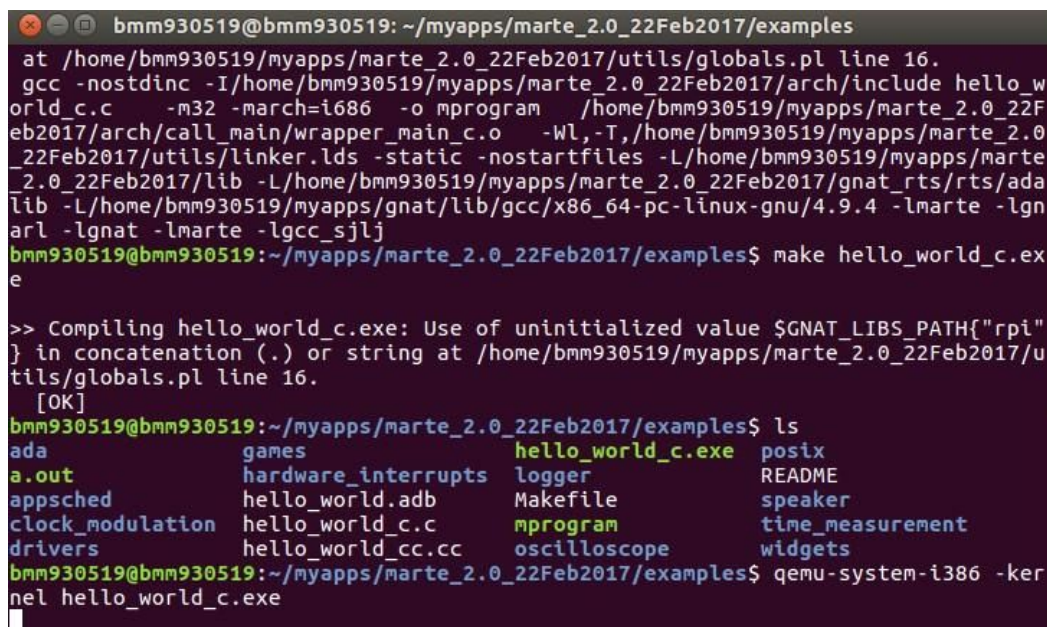


```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/examples
-o mprogram
Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string
at /home/bmm930519/myapps/marte_2.0_22Feb2017/utils/globals.pl line 16.
gcc -nostdinc -I/home/bmm930519/myapps/marte_2.0_22Feb2017/arch/include hello_w
orld_c.c -m32 -march=i686 -o mprogram /home/bmm930519/myapps/marte_2.0_22F
eb2017/arch/call_main/wrapper_main.c.o -WL,-T,/home/bmm930519/myapps/marte_2.0
_22Feb2017/utils/linker.lds -static -nostartfiles -L/home/bmm930519/myapps/marte
_2.0_22Feb2017/lib -L/home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/rts/ada
lib -L/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgn
arl -lgnat -lmarte -lgcc_sjlj
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ make hello_world_c.ex
e
>> Compiling hello_world_c.exe: Use of uninitialized value $GNAT_LIBS_PATH{"rpi"
} in concatenation (.) or string at /home/bmm930519/myapps/marte_2.0_22Feb2017/u
tils/globals.pl line 16.
[OK]
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ ls
ada          games          hello_world_c.exe  posix
a.out        hardware_interru pts          logger            README
appsched     hello_world.adb  Makefile          speaker
clock_modulation hello_world.c.c  mprogram         time_measurement
drivers       hello_world.cc.cc oscilloscope      widgets
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$
```

Ilustración 24 Archivos mprogram y hello_world_c.exe

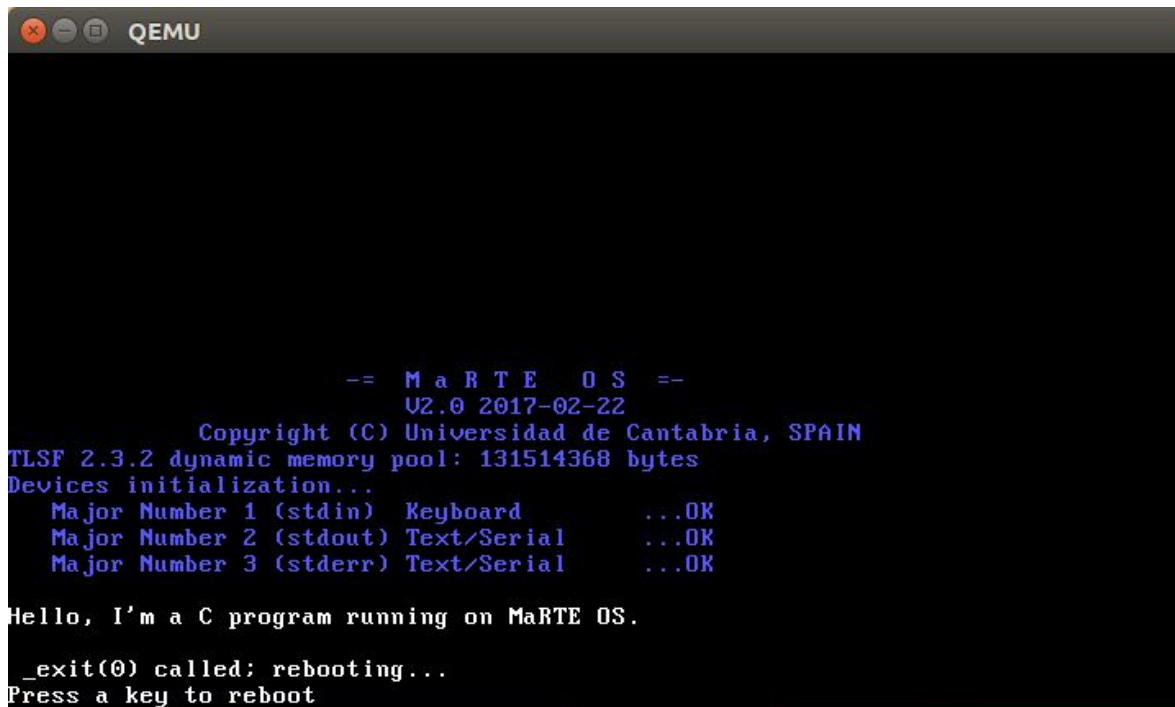
Y como podemos ver, en efecto se han creado.

El último paso es ejecutar la siguiente instrucción “qemu-system-i386 -kernel hello_world_c.exe”, esta instrucción nos ayudará a visualizar en una ventana de qemu, el archivo ejecutable corriendo sobre nuestro SOTR.



```
bmm930519@bmm930519: ~/myapps/marte_2.0_22Feb2017/examples
at /home/bmm930519/myapps/marte_2.0_22Feb2017/utils/globals.pl line 16.
gcc -nostdinc -I/home/bmm930519/myapps/marte_2.0_22Feb2017/arch/include hello_w
orld_c.c -m32 -march=i686 -o mprogram /home/bmm930519/myapps/marte_2.0_22F
eb2017/arch/call_main/wrapper_main.c.o -WL,-T,/home/bmm930519/myapps/marte_2.0
_22Feb2017/utils/linker.lds -static -nostartfiles -L/home/bmm930519/myapps/marte
_2.0_22Feb2017/lib -L/home/bmm930519/myapps/marte_2.0_22Feb2017/gnat_rts/rts/ada
lib -L/home/bmm930519/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgn
arl -lgnat -lmarte -lgcc_sjlj
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ make hello_world_c.ex
e
>> Compiling hello_world_c.exe: Use of uninitialized value $GNAT_LIBS_PATH{"rpi"
} in concatenation (.) or string at /home/bmm930519/myapps/marte_2.0_22Feb2017/u
tils/globals.pl line 16.
[OK]
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ ls
ada          games          hello_world_c.exe  posix
a.out        hardware_interru pts          logger            README
appsched     hello_world.adb  Makefile          speaker
clock_modulation hello_world.c.c  mprogram         time_measurement
drivers       hello_world.cc.cc oscilloscope      widgets
bmm930519@bmm930519:~/myapps/marte_2.0_22Feb2017/examples$ qemu-system-i386 -ker
nel hello_world_c.exe
```

Ilustración 25 Instrucción de visualización del archivo ejecutable.



```
-- M a R T E   O S  ==
U2.0 2017-02-22
Copyright (C) Universidad de Cantabria, SPAIN
TLSF 2.3.2 dynamic memory pool: 131514368 bytes
Devices initialization...
Major Number 1 (stdin) Keyboard      ...OK
Major Number 2 (stdout) Text/Serial   ...OK
Major Number 3 (stderr) Text/Serial   ...OK

Hello, I'm a C program running on MaRTE OS.

_exit(0) called; rebooting...
Press a key to reboot
```

Ilustración 26 Ventana de visualización del archivo ejecutable corriendo en el SOTR

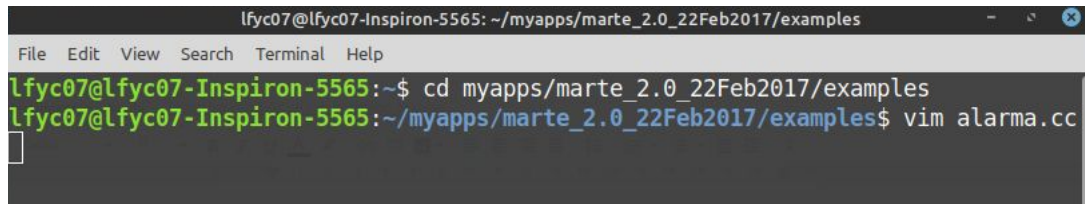
Con esto comprobamos que el SOTR MaRTE OS se instaló correctamente y ahora podemos trabajar sobre él.

Si se ejecuta en Windows es necesario instalar MinGW. Se puede descargar de:

<https://osdn.net/projects/mingw/releases/>

Creación del programa de alarma

Una vez que hemos comprobado el funcionamiento de MarteOS, debemos de comprobar que estemos en el directorio “myapps/marte_2.0_22Feb2017/examples”, usando el comando cd. Una vez en la carpeta usamos el comando “vim alarma.cc” para crear un programa en C++ llamado alarma.



```
lfyc07@lfyc07-Inspiron-5565: ~/myapps/marte_2.0_22Feb2017/examples
File Edit View Search Terminal Help
lfyc07@lfyc07-Inspiron-5565:~$ cd myapps/marte_2.0_22Feb2017/examples
lfyc07@lfyc07-Inspiron-5565:~/myapps/marte_2.0_22Feb2017/examples$ vim alarma.cc
```

Ilustración 27 Creación del programa de alarma mediante vim

Dentro del archivo “alarma.cc” escribimos el siguiente código:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>           //Para poder obtener la fecha del sistema
#include <unistd.h>         //Para poder usar la función sleep

int main ()
{
    time_t tiempo;
    struct tm *hora;
    int h=0, m=0, j=0, n=0, i=0;
    bool stat=0;

    time(&tiempo);
    hora = localtime(&tiempo);

    printf ( "\nEsta es la fecha y hora: %s\n", asctime (hora) );
    printf ( "A que hora quieres que suene la alarma? (h m): ");
    scanf ("%d %d",&h,&m);

    i=time(NULL);

    for(;;)
    {
        sleep(1);
        if(time(NULL)!=i && stat!=1)
        {
            printf ( "%s\n", asctime (hora) );
            i=time(NULL);
        }
    }
}
```

```

time(&tiempo);
hora = localtime( &tiempo);

if((*hora).tm_hour==h && (*hora).tm_min==m && n<3)
{
    printf("DESPIERTA!!\n");
    n++;
    stat=1;
}
else if (n==3)
{
    i=time(NULL);
    while (n==3)
    {
        if (time(NULL)==i+4) n=0;
    }
}
}
//Fin del programa

```

Al ejecutar el programa primero se muestra la fecha del sistema, luego se pregunta al usuario cuando quiere que suene la alarma, para lo cual el usuario ingresa con el teclado la hora (0-23) "espacio" minuto "0-59" (ex. 15 35), luego debe oprimir "Enter".

El programa imprimirá en pantalla el mensaje "Despierta!!" cuando se llegue a la hora y minutos ingresados, el mensaje se imprimirá indefinidamente hasta que se cancele la ejecución del programa. Si se introduce un horario anterior al actual la alarma solo se activaría hasta el día siguiente, ya que solo se revisa si el horario introducido coincide con el actual, no si el horario ya pasó.

```
lfyc07@lfyc07-Inspiron-5565: ~/myapps/marte_2.0_22Feb2017/examples
File Edit View Search Terminal Help

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>

int main ()
{
    time_t tiempo;
    struct tm *hora;
    int h=0, m=0, j=0, n=0, i=0;
    bool stat=0;

    time(&tiempo);
    hora = localtime(&tiempo);

    printf ( "\nEsta es la fecha y hora: %s\n", asctime (hora) );
    printf ( "A que hora quieres que suene la alarma? (h m): ");
    scanf ("%d %d",&h,&m);

    i=time(NULL);
    for(;;)
    {
        sleep(1);
        if(time(NULL)!=i && stat!=1)
        {
            printf ( "%s\n", asctime (hora) );
            i=time(NULL);
        }

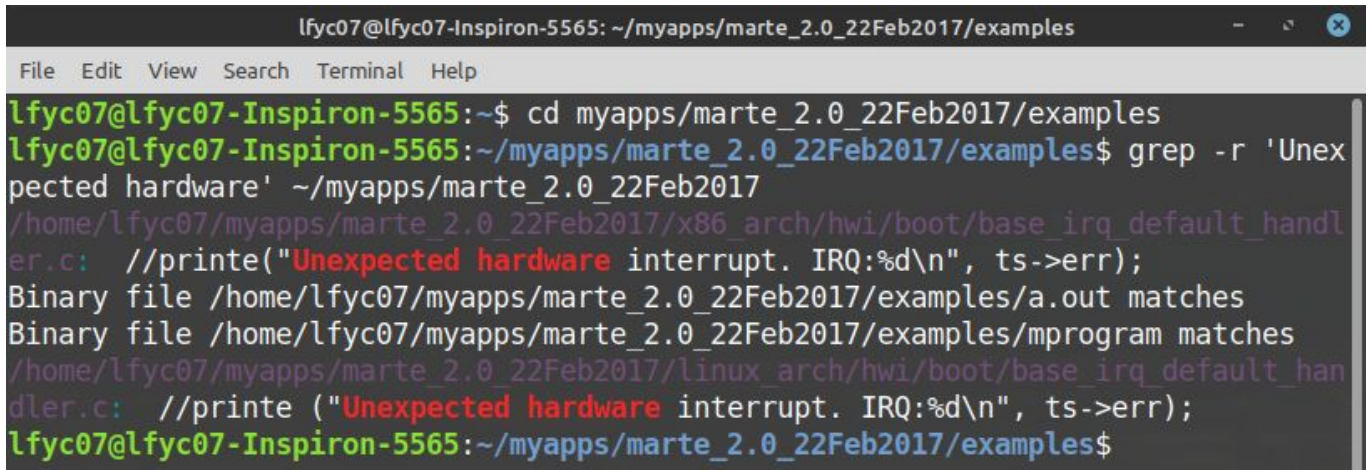
        time(&tiempo);
        hora = localtime( &tiempo);

        if((*hora).tm_hour==h && (*hora).tm_min==m && n<3)
        {
            printf("DESPIERTA!!\n");
            n++;
            stat=1;
        }
        else if (n==3)
        {
            i=time(NULL);
            while (n==3)
            {
                if (time(NULL)==i+4) n=0;
            }
        }
    }
}
-- INSERT --
```

Ilustración 28 Programa de alarma en C++

Una vez creado el programa de alarma presionamos la tecla “Escape”, luego “:wq” o “:x” y finalmente “Enter” para guardar el documento.

Como el archivo no se ejecutaba correctamente al momento de iniciar la live USB en la máquina, dado que mandaba el mensaje de error “Unexpected hardware interrupt: IRQ7”, se comentaron las líneas donde aparece el mensaje de error. Para esto primero se busco donde aparecía “Unexpected hardware” mediante el comando “grep -r 'Unexpected hardware' ~/myapps/marte_2.0_22Feb2017”.



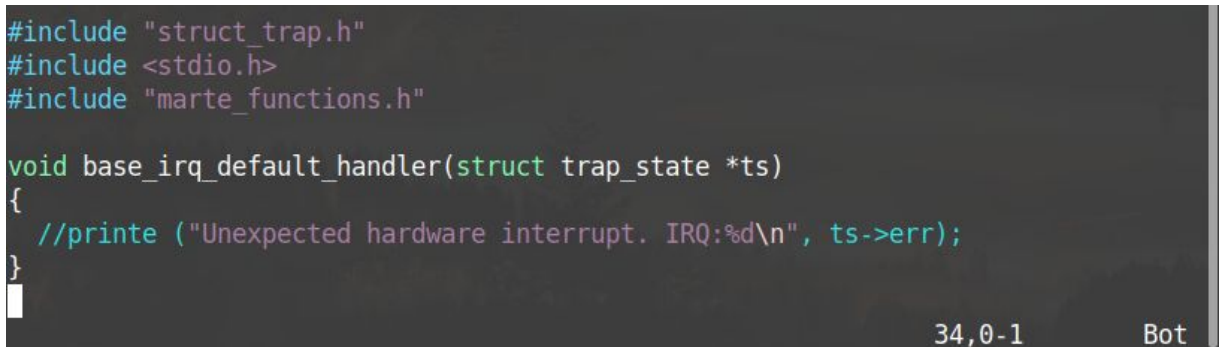
```
lfyc07@lfyc07-Inspiron-5565: ~/myapps/marte_2.0_22Feb2017/examples
File Edit View Search Terminal Help
lfyc07@lfyc07-Inspiron-5565:~$ cd myapps/marte_2.0_22Feb2017/examples
lfyc07@lfyc07-Inspiron-5565:~/myapps/marte_2.0_22Feb2017/examples$ grep -r 'Unexpected hardware' ~/myapps/marte_2.0_22Feb2017
/home/lfyc07/myapps/marte_2.0_22Feb2017/x86_arch/hwi/boot/base_irq_default_handler.c: //printe("Unexpected hardware interrupt. IRQ:%d\n", ts->err);
Binary file /home/lfyc07/myapps/marte_2.0_22Feb2017/examples/a.out matches
Binary file /home/lfyc07/myapps/marte_2.0_22Feb2017/examples/mprogram matches
/home/lfyc07/myapps/marte_2.0_22Feb2017/linux_arch/hwi/boot/base_irq_default_handler.c: //printe ("Unexpected hardware interrupt. IRQ:%d\n", ts->err);
lfyc07@lfyc07-Inspiron-5565:~/myapps/marte_2.0_22Feb2017/examples$
```

Ilustración 29 Búsqueda del error con el comando grep

El comando nos da dos ubicaciones donde aparece el mensaje de error

- ~/marte_2.0_22Feb2017/x86_arch/hwi/boot/base_irq_default_handler.c
- ~/marte_2.0_22Feb2017/linux_arch/hwi/boot/base_irq_default_handler.c

Usamos cd para cambiar de ubicación a esos directorios y usamos vim para comentar la línea “printe (“Unexpected hardware interrupt. IRQ:%d\n”, ts->err);”



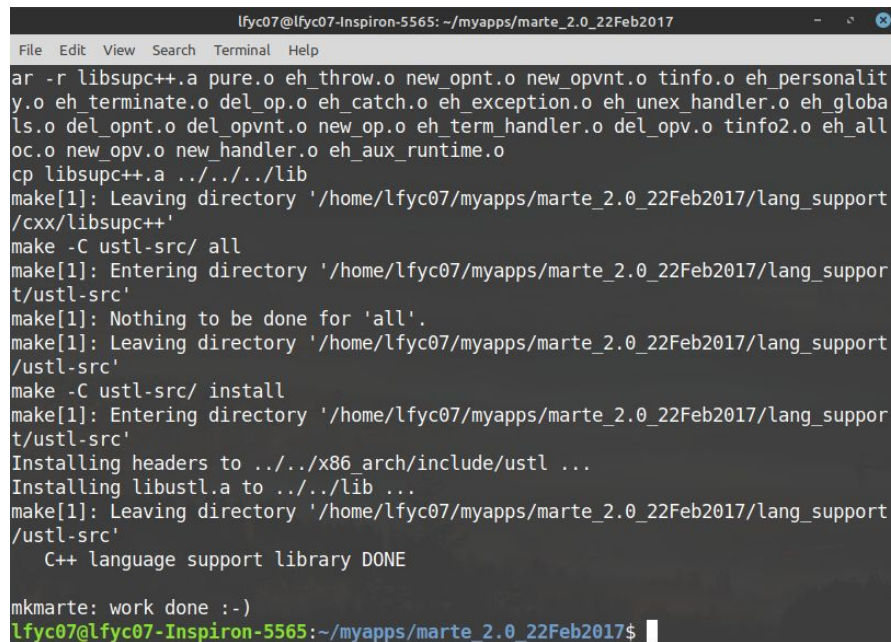
```
#include "struct_trap.h"
#include <stdio.h>
#include "marte_functions.h"

void base_irq_default_handler(struct trap_state *ts)
{
    //printe ("Unexpected hardware interrupt. IRQ:%d\n", ts->err);
}

34,0-1 Bot
```

Ilustración 30 Edición del archivo de error en Vim

Regresamos el directorio “martes_2.0_22Feb2017” y ejecutamos el comando “mkmarte”, es importante revisar todos lo que apareció en la terminal, porque pueden aparecer errores a la mitad de la compilación y aun así se termine de compilar MarteOS.

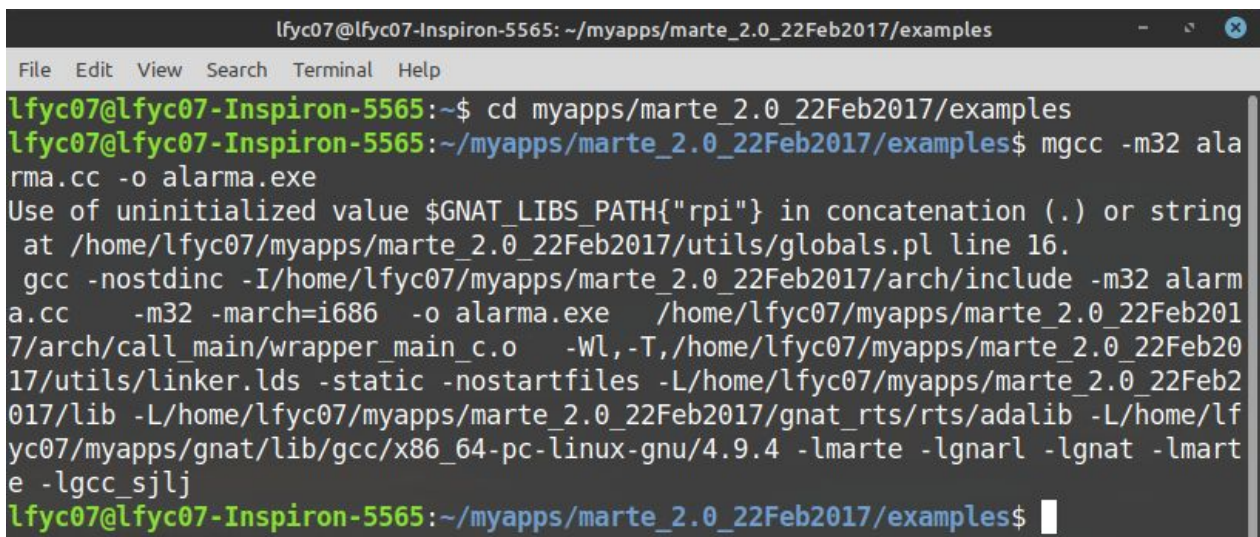


```
lfyc07@lfyc07-Inspiron-5565: ~/myapps/marte_2.0_22Feb2017
File Edit View Search Terminal Help
ar -r libsupc++.a pure.o eh_throw.o new_opnt.o new_opvnt.o tinfo.o eh_personalit
y.o eh_terminate.o del_op.o eh_catch.o eh_exception.o eh_unex_handler.o eh_globa
ls.o del_opnt.o del_opvnt.o new_op.o eh_term_handler.o del_opv.o tinfo2.o eh_all
oc.o new_opv.o new_handler.o eh_aux_runtime.o
cp libsupc++.a ../../lib
make[1]: Leaving directory '/home/lfyc07/myapps/marte_2.0_22Feb2017/lang_support
/cxx/libsupc++'
make -C ustl-src/ all
make[1]: Entering directory '/home/lfyc07/myapps/marte_2.0_22Feb2017/lang_support
t/ustl-src'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/lfyc07/myapps/marte_2.0_22Feb2017/lang_support
t/ustl-src'
make -C ustl-src/ install
make[1]: Entering directory '/home/lfyc07/myapps/marte_2.0_22Feb2017/lang_support
t/ustl-src'
Installing headers to ../../x86_arch/include/ustl ...
Installing libustl.a to ../../lib ...
make[1]: Leaving directory '/home/lfyc07/myapps/marte_2.0_22Feb2017/lang_support
t/ustl-src'
C++ language support library DONE

mkmarte: work done :-)
lfyc07@lfyc07-Inspiron-5565:~/myapps/marte_2.0_22Feb2017$
```

Ilustración 31 Ejecución del comando “mkmarte”

Regresamos a la carpeta “examples” y utilizamos el comando “gcc -m32 alarma.cc -o alarma.exe” para crear un archivo tipo.exe con el programa de alarma



```
lfyc07@lfyc07-Inspiron-5565: ~/myapps/marte_2.0_22Feb2017/examples
File Edit View Search Terminal Help
lfyc07@lfyc07-Inspiron-5565:~$ cd myapps/marte_2.0_22Feb2017/examples
lfyc07@lfyc07-Inspiron-5565:~/myapps/marte_2.0_22Feb2017/examples$ gcc -m32 ala
rma.cc -o alarma.exe
Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string
at /home/lfyc07/myapps/marte_2.0_22Feb2017/utls/globals.pl line 16.
gcc -nostdinc -I/home/lfyc07/myapps/marte_2.0_22Feb2017/arch/include -m32 alarm
a.cc -m32 -march=i686 -o alarma.exe /home/lfyc07/myapps/marte_2.0_22Feb201
7/arch/call_main/wrapper_main.c.o -Wl,-T,/home/lfyc07/myapps/marte_2.0_22Feb20
17/utls/linker.lds -static -nostartfiles -L/home/lfyc07/myapps/marte_2.0_22Feb2
017/lib -L/home/lfyc07/myapps/marte_2.0_22Feb2017/gnat_rts/rts/adalib -L/home/lf
yc07/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnarl -lgnat -lmart
e -lgcc_sjlj
lfyc07@lfyc07-Inspiron-5565:~/myapps/marte_2.0_22Feb2017/examples$
```

Ilustración 32 Compilación del programa de alarma

Con la instrucción “qemu-system-i386 -kernel alarma.exe”, podemos visualizar en una ventana de qemu el archivo ejecutable corriendo sobre MarteOS, así comprobamos que se ejecuta correctamente. Pueden aparecer algunos errores al correr el programa en QEMU, pero estos desaparecen cuando se ejecuta el programa en una live USB.



```
== M a R T E O S ==
V2.0 2017-02-22
Copyright (C) Universidad de Cantabria, SPAIN
TLSF 2.3.2 dynamic memory pool: 131485696 bytes
Devices initialization...
Major Number 1 (stdin) Keyboard      ...OK
Major Number 2 (stdout) Text/Serial   ...OK
Major Number 3 (stderr) Text/Serial   ...OK
Major Number 5 (lpt0)  Printer Port   ...OK

Esta es la fecha y hora: Tue Oct 29 21:48:43 2019
A que hora quieres que suene la alarma? (h m): _
```

Ilustración 33 Programa de alarma en QEMU

Si se ejecuta en Windows es necesario instalar MinGW. Se puede descargar de:
<https://osdn.net/projects/mingw/releases/>

Creación de una live USB

Una vez creado y probado el archivo se copia el archivo a la carpeta “MAKEFILE_GRUB_Legacy_Kernel_elf”

descargada del repositorio de Github “sotrteacher/sotr_201808_201812”, el cual se encuentra en:

https://github.com/sotrteacher/sotr_201808_201812

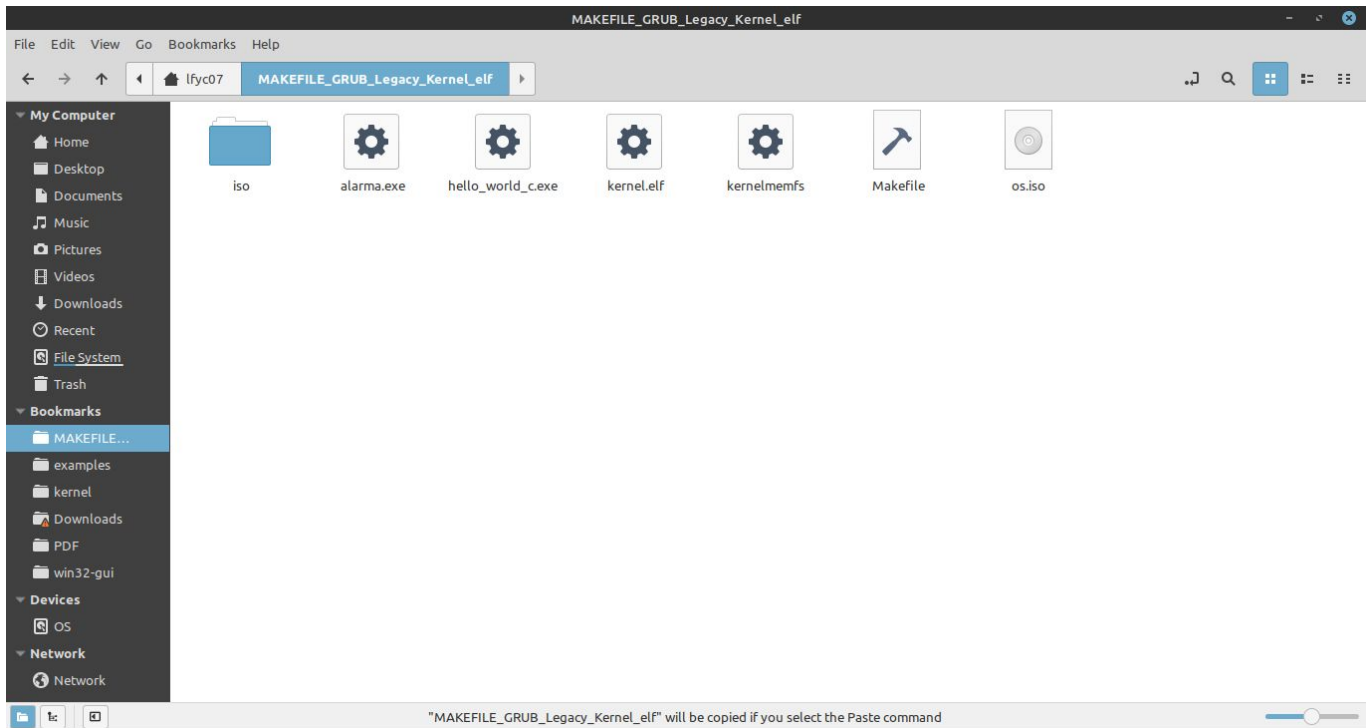


Ilustración 34 Carpeta MAKEFILE_GRUB_Legacy_Kernel_elf

Se utiliza el comando “cp -v alarma.exe kernel.elf” en la terminal de Linux para renombrar el archivo “alarma.exe” como “kernel.elf”



Ilustración 35 Comando cp

Usamos el comando “make” en la terminal para crear el archivo tipo iso con el que crearemos nuestra live USB

```
lfyc07@lfyc07-Inspiron-5565: ~/MAKEFILE_GRUB_Legacy_Kernel_elf
File Edit View Search Terminal Help
lfyc07@lfyc07-Inspiron-5565:~$ cd MAKEFILE_GRUB_Legacy_Kernel_elf
lfyc07@lfyc07-Inspiron-5565:~/MAKEFILE_GRUB_Legacy_Kernel_elf$ cp -v alarma.exe
kernel.elf
'alarma.exe' -> 'kernel.elf'
lfyc07@lfyc07-Inspiron-5565:~/MAKEFILE_GRUB_Legacy_Kernel_elf$ make
cp -v kernel.elf iso/boot/kernel.elf
'kernel.elf' -> 'iso/boot/kernel.elf'
genisoimage -R
\
-b boot/grub/stage2_eltorito\
-no-emul-boot \
-boot-load-size 4 \
-A os \
-input-charset utf8 \
-quiet \
-boot-info-table \
-o os.iso \
iso
lfyc07@lfyc07-Inspiron-5565:~/MAKEFILE_GRUB_Legacy_Kernel_elf$
```

Ilustración 36 Comando make

En el software de Linux live USB creator, el cual podemos descargar de: <https://www.linuxliveusb.com/>, primero seleccionamos la USB donde vamos a crear nuestra live USB, luego el segundo paso nos pide que seleccionemos el archivo iso que creamos en el paso anterior, finalmente presionamos la imagen del rayo para crear nuestra live USB. El proceso puede tardar varios minutos.



Ilustración 37 Creación de la live USB

Finalmente ejecutamos la live USB en una computadora presionando la tecla adecuada (normalmente F12) al encender la computadora y seleccionando nuestra USB para iniciar el sistema.

```

== M a R T E   O S ==
U2.0 2017-02-22
Copyright (C) Universidad de Cantabria, SPAIN
TLSF 2.3.2 dynamic memory pool: 131485696 bytes
Devices initialization...
Major Number 1 (stdin) Keyboard      ...OK
Major Number 2 (stdout) Text/Serial   ...OK
Major Number 3 (stderr) Text/Serial   ...OK
Major Number 5 (lpt0) Printer Port    ...OK

Esta es la fecha y hora: Tue Oct 29 21:48:43 2019
A que hora quieres que suene la alarma? (h m): _
```

Ilustración 38 Programa de alarma ejecutado desde la live USB