



Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria
en Ingenierías y Tecnologías
Avanzadas

INSTALACIÓN DE UN SOTR

“Instalación de MaRTEOS en la terminal Debian desde
Windows 10”

BARRERA ANGELES DIEGO IVÁN

Sistemas Operativos en Tiempo Real

3MV11



Profesor:
Lamberto Maza Casas

Ciudad de México, México

Octubre 2019

Instalación de MaRTEOS

Introducción

MarteOS es un sistema operativo de tiempo real para aplicaciones embebidas que sigue a la mínima en tiempo real POSIX.13. El proyecto se desarrolla por el Grupo de computación y Tiempo Real en la Universidad de Cantabria, aunque también existen colaboradores en distintos lugares.

El entorno de desarrollo se basa en la GNU compiladores GNAT, GCC y GCJ. La mayor parte de su código está escrito en Ada con algunas partes C y ensamblador. [1]

Cabe mencionar que para la instalación del mencionado sistema operativo se utilizó un ordenador con Windows 10 y la aplicación Debian que se puede encontrar fácilmente en la Microsoft Store.

Antes de la instalación de MaRTEOS

Si aún no se ha instalado la aplicación Debian para Windows, nos dirigimos a la Microsoft Store la cual podemos hallar a través del buscador. Una vez ahí, escribimos “Debian” en la barra de búsqueda de Microsoft Store el cual nos guiará a la siguiente pantalla en la que solo basta seguir las instrucciones para la correcta instalación.

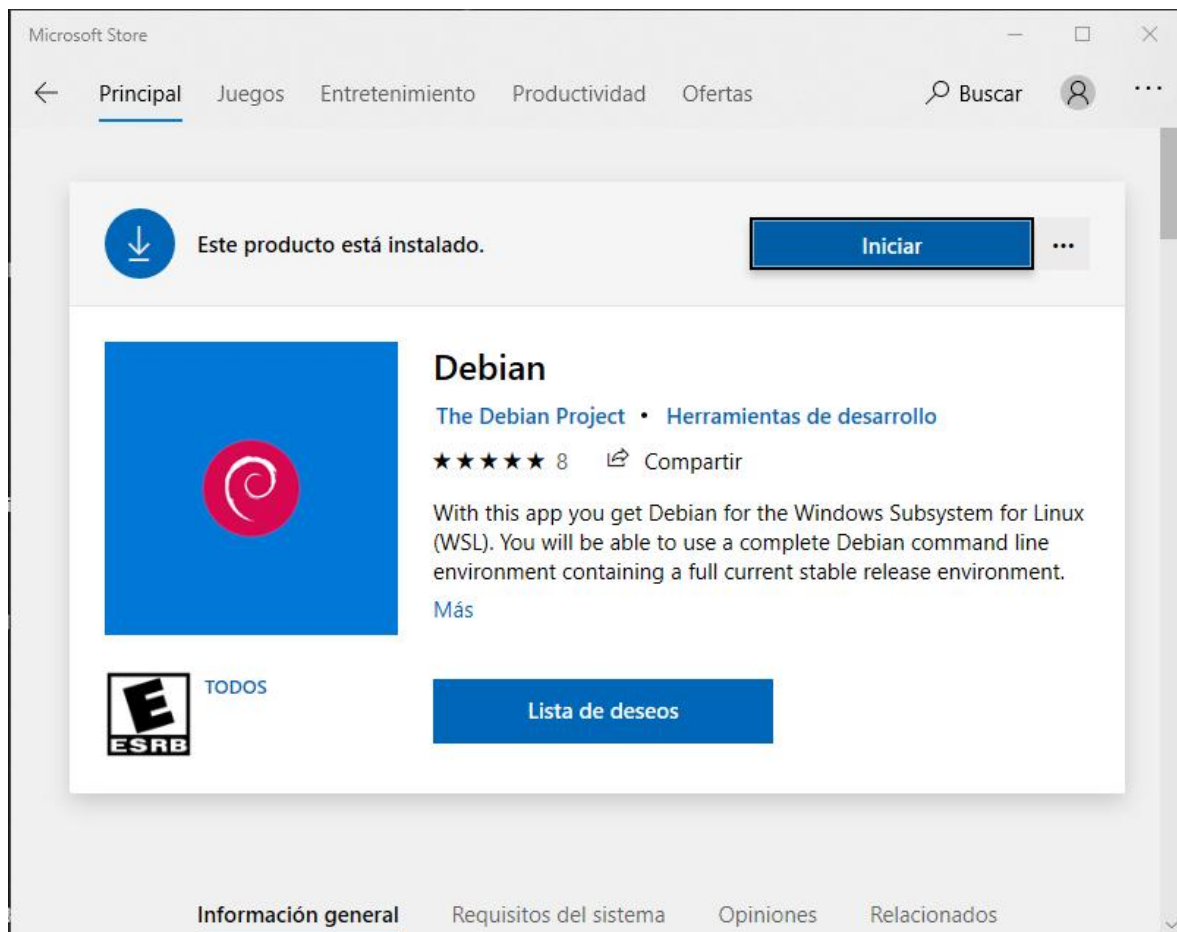


Figura 1. Instalación de Debian en Windows.

Una vez instalado dicha ventana de comandos, se verá de la siguiente manera:

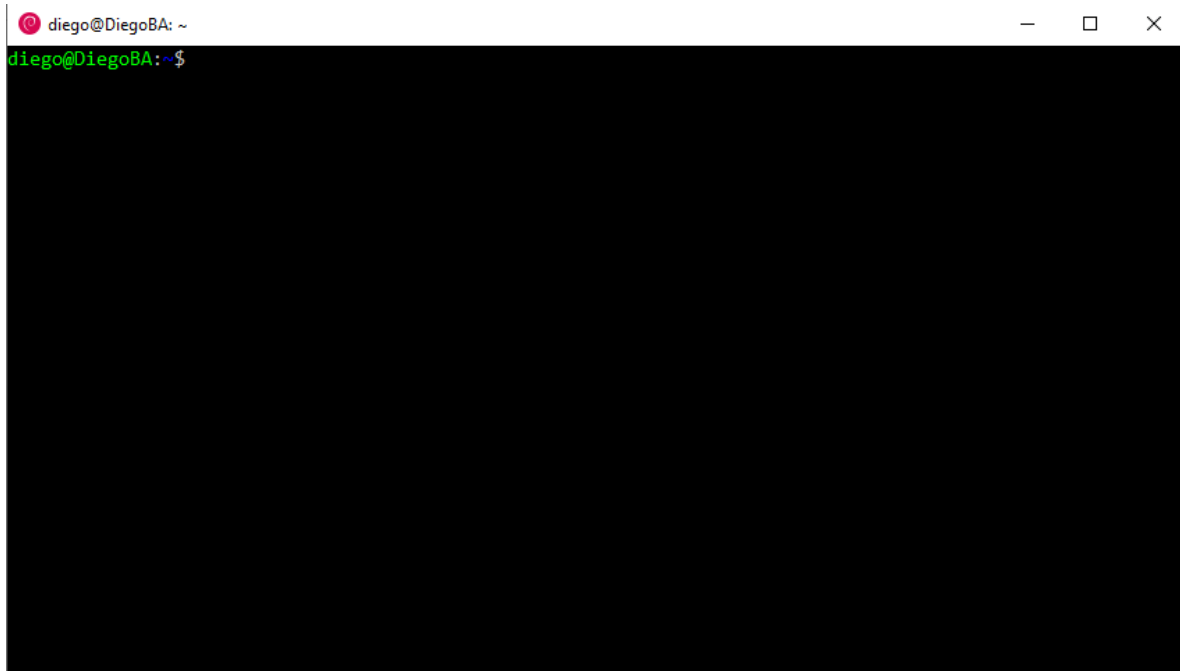




Figura 2. Debian corriendo sobre Windows.

- Procederemos a obtener el compilador GNAT de la siguiente liga: <http://mirrors.cdn.adacore.com/art/5739cefdc7a447658e0b016b>
- De igual manera nos dirigiremos a la siguiente liga para descargar el sistema operativo en tiempo real a descargar: https://martel.unican.es/martel/martel 2.0 22Feb2017_src.tar.gz

ste equipo > Descargas				Buscar
Nombre	Fecha de modificación	Tipo	Tamaño	
▼ hoy (2)				
 marte_2.0_22Feb2017_src.tar	30/10/2019 03:04 p. m.	Archivo WinRAR	6,156 KB	
 gnat-gpl-2016-x86_64-linux-bin.tar	30/10/2019 02:46 p. m.	Archivo WinRAR	364,076 KB	
> al principio de esta semana (2)				
> la semana pasada (4)				

- Realizados los pasos anteriores, nos dirigimos a la terminal de Debian y creamos una carpeta llamada *myapps* donde comenzaremos a trabajar. Para lograr lo anterior escribimos el comando *mkdir* y a continuación el nombre del directorio.

```
~$ mkdir myapps
```

- Comprobamos que se haya creado el directorio desplegando el contenido de la carpeta principal utilizando el comando *ls*. Deberíamos poder visualizar el nombre de la carpeta en los elementos enlistados.

```
diego@DiegoBA: ~  
diego@DiegoBA:~$ mkdir myapps  
diego@DiegoBA:~$ ls  
exer_hu myapps nano.save sotr_201808_201812 xv6 xv6-public  
diego@DiegoBA:~$
```

Figura 3. Carpeta "myapps".

- Lo siguiente a realizar es mover los dos archivos .tar que descargamos previamente. Sabiendo que se encuentran en el directorio de Descargas, nos movemos a dicho lugar con la siguiente instrucción.

```
$ cd /mnt/c/Users/"Nombre del Usuario"/Downloads
```

- Las instrucciones para mover dichos archivos a la carpeta *myapps* previamente creada son las siguientes:

```
$ mv marte_2.0_22Feb2017_src.tar.gz /home/"usuario"/myapps/
```

```
$ mv gnat-gpl-2016-x86_64-linux-bin.tar.gz /home/"usuario"/myapps
```

- Para comprobar que se movieron correctamente, nos movemos al directorio de my apps:

```
$ cd
```

```
$cd myapps/
```

- Con `/ls` verificamos que los archivos se encuentren ahí. Podremos visualizar los siguiente:

```
diego@DiegoBA: ~/myapps  
diego@DiegoBA:~$ ls  
exer_hu FLYANDSHOOT myapps nano.save ORGANIZACION sotr_201808_201812 xv6 xv6-public  
diego@DiegoBA:~$ cd /mnt/c/Users/Diego/Downloads/  
diego@DiegoBA:/mnt/c/Users/Diego/Downloads$ mv gnat-gpl-2016-x86_64-linux-bin.tar.gz /home/diego/myapps/  
diego@DiegoBA:/mnt/c/Users/Diego/Downloads$ cd  
diego@DiegoBA:~$ ls  
exer_hu FLYANDSHOOT myapps nano.save ORGANIZACION sotr_201808_201812 xv6 xv6-public  
diego@DiegoBA:~$ cd myapps/  
diego@DiegoBA:~/myapps$ ls  
gnat-gpl-2016-x86_64-linux-bin.tar.gz marte_2.0_22Feb2017_src.tar.gz  
diego@DiegoBA:~/myapps$
```

Figura 4. Contenido de carpeta *myapps*.

- Ahora creamos una nueva carpeta dentro de *myapps* con el nombre *gnat*.

```
$ mkdir gnat
```

- De igual manera rectificamos que la carpeta se haya creado con la instrucción `/ls`.

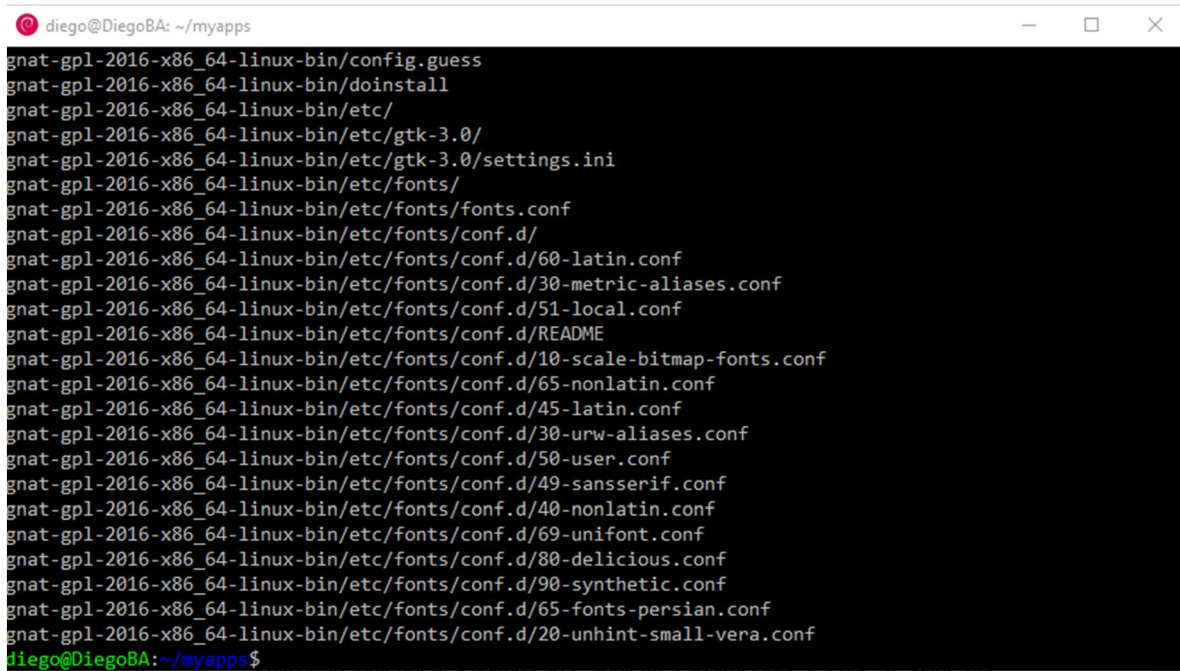
```
diego@DiegoBA:~/myapps$ ls
gnat-gpl-2016-x86_64-linux-bin.tar.gz  marte_2.0_22Feb2017_src.tar.gz
diego@DiegoBA:~/myapps$ mkdir gnat
diego@DiegoBA:~/myapps$ ls
gnat  gnat-gpl-2016-x86_64-linux-bin.tar.gz  marte_2.0_22Feb2017_src.tar.gz
diego@DiegoBA:~/myapps$
```

Figura 5. Directorio *myapps*.

- Procederemos a descomprimir los archivos `.tar` dentro de la carpeta *myapps*, para realizar lo anterior escribimos las siguientes instrucciones; cabe resaltar que tenemos que esperar a que terminen cada una de las operaciones para proceder. Puede demorar algunos minutos.

```
$ tar xvf gnat-gpl-2016-x86_64-linux-bin.tar.gz
```

```
$ tar xvf marte_2.0_22Feb2017_src.tar.gz
```



```
diego@DiegoBA: ~/myapps
gnat-gpl-2016-x86_64-linux-bin/config.guess
gnat-gpl-2016-x86_64-linux-bin/doinstall
gnat-gpl-2016-x86_64-linux-bin/etc/
gnat-gpl-2016-x86_64-linux-bin/etc/gtk-3.0/
gnat-gpl-2016-x86_64-linux-bin/etc/gtk-3.0/settings.ini
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/fonts.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/60-latin.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/30-metric-aliases.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/51-local.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/README
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/10-scale-bitmap-fonts.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/65-nonlatin.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/45-latin.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/30-urw-aliases.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/50-user.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/49-sansserif.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/40-nonlatin.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/69-unifont.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/80-delicious.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/90-synthetic.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/65-fonts-persian.conf
gnat-gpl-2016-x86_64-linux-bin/etc/fonts/conf.d/20-unhint-small-vera.conf
diego@DiegoBA:~/myapps$
```

Figura 6. Pantalla final al descomprimir *gnat*.

```
diego@DiegoBA: ~/myapps
marte_2.0_22Feb2017/x86_arch/include/misc/linux_list.h
marte_2.0_22Feb2017/x86_arch/include/misc/generic_lists.h
marte_2.0_22Feb2017/x86_arch/include/misc/marte_non_local_jump.h
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_hwttime.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/unistd.h
marte_2.0_22Feb2017/x86_arch/include/dirent.h
marte_2.0_22Feb2017/x86_arch/include/stdlib.h
diego@DiegoBA:~/myapps$
```

Figura 7. Pantalla final al descomprimir Marte.

- Lo siguiente a realizar es modificar el archivo `bashrc`, para lograrlo escribimos `cd` para ubicarnos en el directorio raíz. Posteriormente escribimos:

`$ nano .bashrc`

- Abrirá el siguiente script.

```
diego@DiegoBA: ~
GNU nano 3.2 .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
    *i*) ;;
    *) 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
HISTFILESIZE=2000

[ Read 123 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line M-E Redo
```

Figura 8. Bashrc

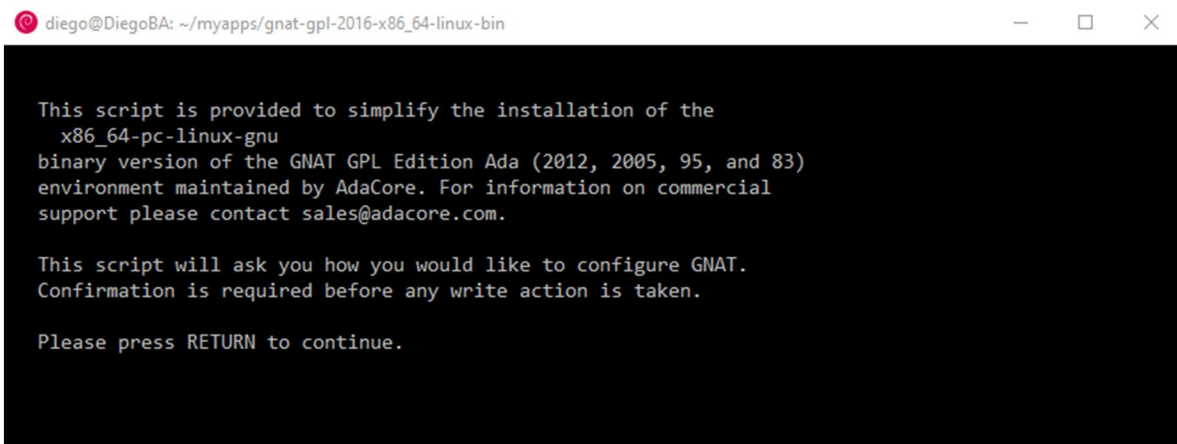
- Al final del archivo escribimos las siguientes líneas

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

- Para salir y guardar los cambios ejecutamos Ctrl+O y Ctrl+X. Es importante cerrar la terminal en la que estábamos trabajando para realizar los siguientes pasos. Con una terminal nueva abierta, ingresamos a la carpeta *gnat-gpl-2016x86_64-linux-bin*. Para lograrlo escribimos lo siguiente:

```
$ cd myapps/gnat-gpl-2016x86_64-linux-bin
```

- Posteriormente escribimos *./doinstall* y se nos abrirá la siguiente ventana a la cuál le daremos Enter.



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

This script is provided to simplify the installation of the
  x86_64-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.

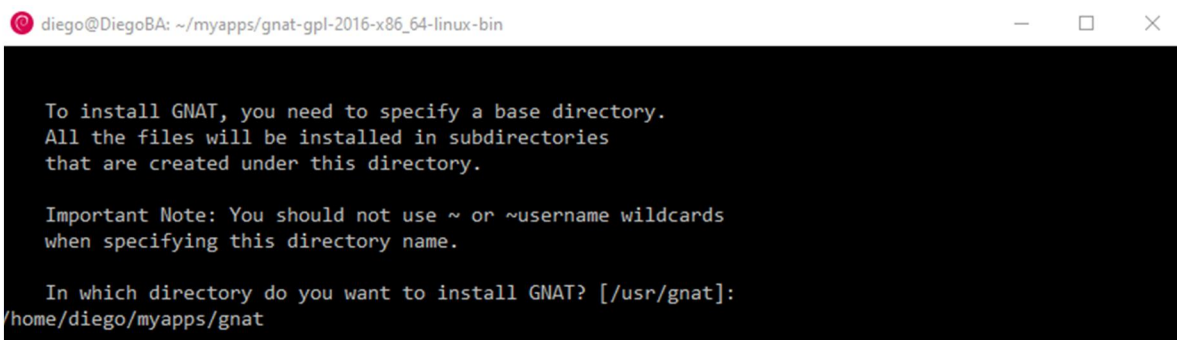
This script will ask you how you would like to configure GNAT.
Confirmation is required before any write action is taken.

Please press RETURN to continue.
```

Figura 9. *./doinstall*

- Nos preguntará la ubicación del directorio en donde instalar el compilador la cual será la carpeta *gnat* creada al principio. Por lo tanto en la dirección escribimos lo siguiente:

```
/home/usuario/myapps/gnat
```



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

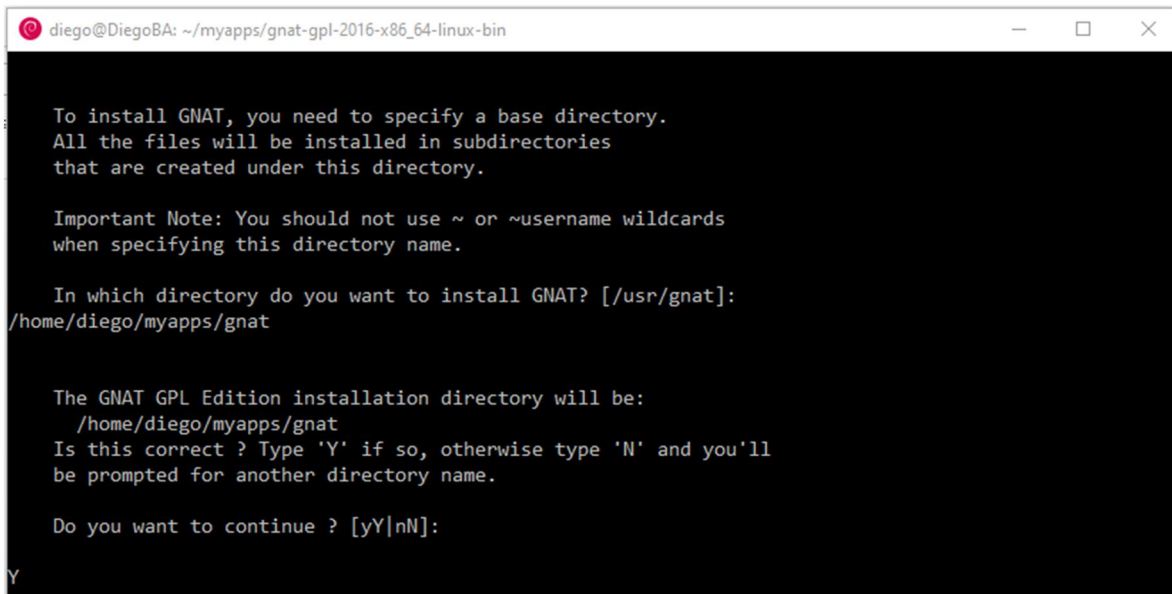
To install GNAT, you need to specify a base directory.
All the files will be installed in subdirectories
that are created under this directory.

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/diego/myapps/gnat
```

Figura 10. Instalación del compilador GNAT.

- Nos preguntará una serie de permisos a conceder a las cuales debemos aceptar escribiendo “Y” o “y”.



```
diego@DiegoBA: ~/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.

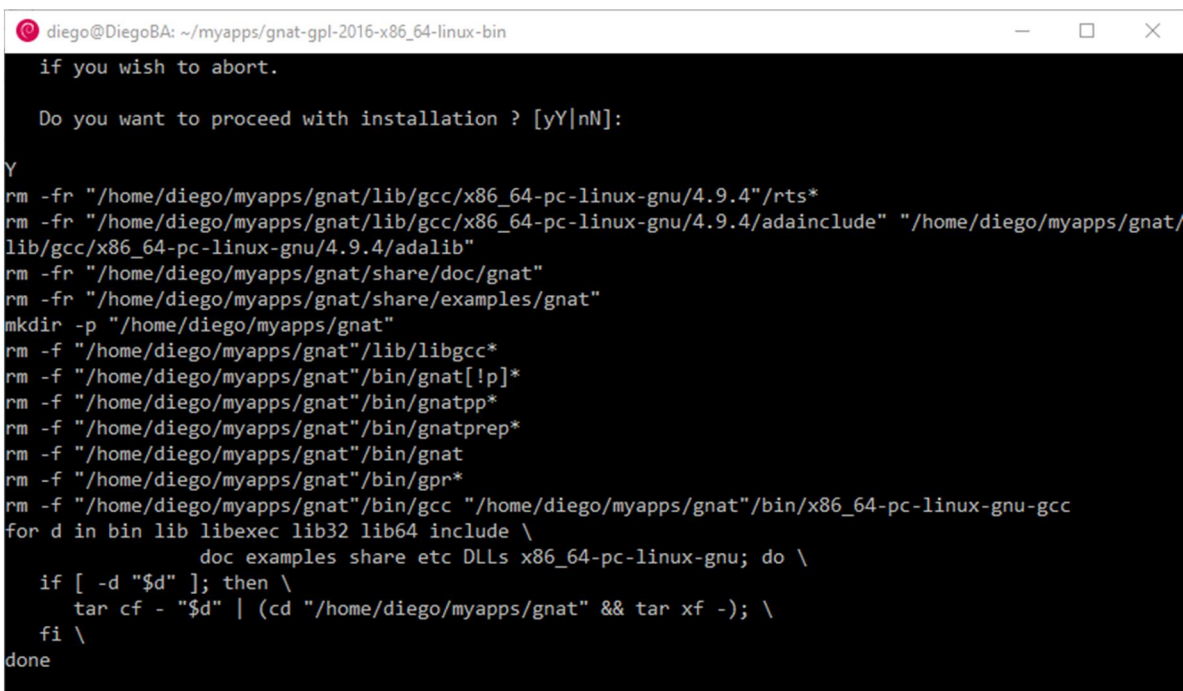
Important Note: You should not use ~ or ~username wildcards
when specifying this directory name.

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

The GNAT GPL Edition installation directory will be:
/home/diego/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
```

Figura 11. Instalación compilador GNAT.



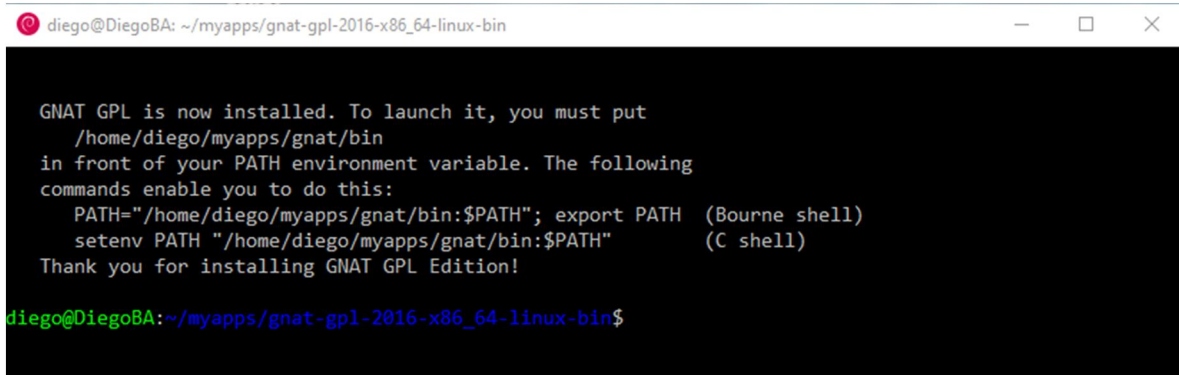
```
diego@DiegoBA: ~/myapps/gnat-gpl-2016-x86_64-linux-bin

if you wish to abort.

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

Figura 12. Instalación compilador GNAT.

- Sabremos que la instalación ha terminado cuando se nos aparezca la siguiente ventana.



```

diego@DiegoBA: ~/myapps/gnat-gpl-2016-x86_64-linux-bin

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

diego@DiegoBA:~/myapps/gnat-gpl-2016-x86_64-linux-bin$

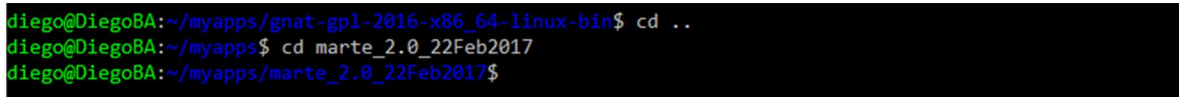
```

Figura 13. Compilador GNAT instalado.

- Lo siguiente es movernos a la carpeta *marTE_2.0_22Feb2017*, para ello utilizamos el comando *cd* para movernos.

```
$ cd ..
```

```
$ cd marTE_2.0_22Feb2017
```



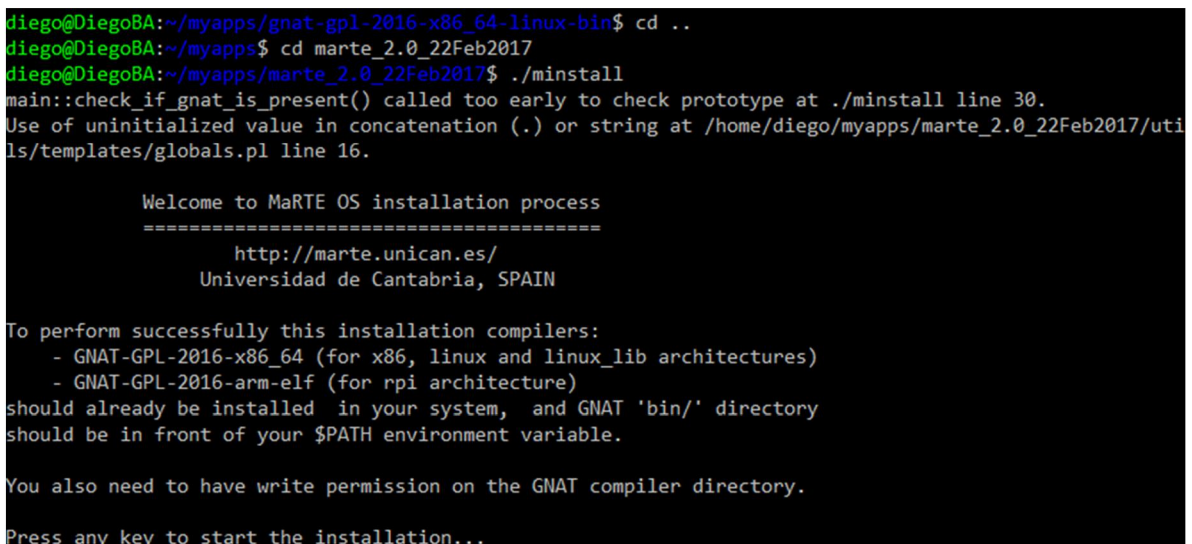
```

diego@DiegoBA:~/myapps/gnat-gpl-2016-x86_64-linux-bin$ cd ..
diego@DiegoBA:~/myapps$ cd marTE_2.0_22Feb2017
diego@DiegoBA:~/myapps/marTE_2.0_22Feb2017$

```

Figura 14. Carpeta *marTE_2.0_22Feb2017*

- Una vez ahí instalaremos el SO ejecutando el comando *./minstall* y ejecutando las instrucciones que nos indique la terminal.



```

diego@DiegoBA:~/myapps/gnat-gpl-2016-x86_64-linux-bin$ cd ..
diego@DiegoBA:~/myapps$ cd marTE_2.0_22Feb2017
diego@DiegoBA:~/myapps/marTE_2.0_22Feb2017$ ./minstall
main::check_if_gnat_is_present() called too early to check prototype at ./minstall line 30.
Use of uninitialized value in concatenation (.) or string at /home/diego/myapps/marTE_2.0_22Feb2017/uti
ls/templates/globals.pl line 16.

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...

```

Figura 15. Instalación de MarTEOS.

```
diego@DiegoBA: ~/myapps/marte_2.0_22Feb2017
Set 32bits libs in /home/diego/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4

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

You may want to add "/home/diego/myapps/marte_2.0_22Feb2017/utils"
to your $PATH environment variable to have direct access to MaRTE
tools (mgnatmake, mgcc, mkmarte, mkrtsmarteuc, msetcurrentarch, etc.)

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

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

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

diego@DiegoBA:~/myapps/marte_2.0_22Feb2017$
```

Figura 16. Instalación completa de MarTEOS.

- Lo siguiente es movernos a la carpeta *utils* ubicada dentro de *marte_2.0_22Feb2017*, por lo tanto solo es necesario escribir *cd utils/*. Una vez ahí ejecutamos la instrucción *msetcurrentarch* la cual nos permitirá seleccionar entre las diferentes arquitecturas incluidas en el sistema operativo. Podemos elegir entre x86, Linux, Linux_lib y rpi; optaremos por la última escribiendo la siguiente instrucción.

```
$ msetcurrentarch x86 i386
```

```
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017$ cd utils/
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/utils$ msetcurrentarch
Use of uninitialized value in concatenation (.) or string at /home/diego/myapps/marte_2.0_22Feb2017/uti
ls/globals.pl line 16.
Current architecture:none

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

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

Figura 17. Elección de arquitectura.

- Nos desplegará la siguiente pantalla la cual nos invita a ejecutar las instrucciones *mkrtsmarteuc* y *mkmarte* que son necesarias para finalizar con éxito la instalación del sistema operativo.

```

diego@DiegoBA: ~/myapps/marte_2.0_22Feb2017/utils
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/sys_marte/marte_*.h /home/diego/myapps/marte_2.0_22Feb2017/arch/include/sys/
rm -f /home/diego/myapps/marte_2.0_22Feb2017/misc//console_management_c.c
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-configuration_parameters.ads /home/diego/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/libm.a /home/diego/myapps/marte_2.0_22Feb2017/objs/x86_objs/
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/console_management.ads /home/diego/myapps/marte_2.0_22Feb2017/misc/
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-kernel-file_system.ads /home/diego/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/console_management.adb /home/diego/myapps/marte_2.0_22Feb2017/misc/
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-direct_io.ads /home/diego/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/diego/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-kernel-devices_table.ads /home/diego/myapps/marte_2.0_22Feb2017/kernel/
rm -f /home/diego/myapps/marte_2.0_22Feb2017/lib && ln -s /home/diego/myapps/marte_2.0_22Feb2017/objs/x86_objs /home/diego/myapps/marte_2.0_22Feb2017/lib
cd /home/diego/myapps/marte_2.0_22Feb2017/gnat_rts/ && rm -f rts && ln -s rts-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'
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/utils$

```

Figura 18. Elección de arquitectura.

```

diego@DiegoBA: ~/myapps/marte_2.0_22Feb2017/utils
ego/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4/adainclude/a-diroro.adb
ar r libgnarl.a a-dispat.o a-dynpri.o a-interr.o a-intnam.o a-reatim.o a-retide.o a-rttiev.o a-synbar.o
a-sytaco.o a-tasatt.o a-taside.o a-taster.o g-boubuf.o g-boumai.o g-semaph.o g-signal.o g-tastus.o g-t
hread.o s-innaop.o s-interr.o s-intman.o s-mudido.o s-osinte.o s-proinf.o s-solita.o s-stusta.o s-taenc
a.o s-taprob.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-taasde.o s-tadeca.o s-tadert.o s-tataat.o s-tpinop.o s-tp
oben.o s-tpobop.o s-tposen.o s-tratas.o thread.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/diego/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ && ln -f -s ../../../../lib/lib
marte.a libmarte.a
cd /home/diego/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ && ln -f -s libmarte.a libc.
a
cd /home/diego/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ && ln -f -s ../../../../lib/mul
tiboote.o multiboote.o
cd /home/diego/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && rm adainclude adalib
cd /home/diego/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && ln -s rts-native/adainclude adainclude
cd /home/diego/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && ln -s rts-native/adalib adalib
mkrtsmarteuc: rts-marteuc_x86 done :)

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

```

Figura 19. Elección de arquitectura.


```
diego@DiegoBA: ~/myapps/marte_2.0_22Feb2017/utls
}
inline const value_type Mask (uoff_t n) const { assert (n < Size); return (1 << (n % s_WordBits));
}
^
ubitset.h:61:41: warning: type qualifiers ignored on function return type [-Wignored-qualifiers]
inline const bool test (uoff_t n) const { return (BitRef(n) & Mask(n)); }
^
ubitset.h:62:47: warning: type qualifiers ignored on function return type [-Wignored-qualifiers]
inline const bool operator[] (uoff_t n) const { return (test(n)); }
^
In file included from ubitset.cc:9:0:
ubitset.h:68:49: warning: type qualifiers ignored on function return type [-Wignored-qualifiers]
inline const value_type to_value (void) const { return (m_Bits[0]); }
^
Linking libustl.a ...
ar: creating libustl.a
make[1]: Leaving directory '/home/diego/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'
make -C ustl-src/ install
make[1]: Entering directory '/home/diego/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'
Installing headers to ../../x86_arch/include/ustl ...
Installing libustl.a to ../../lib ...
make[1]: Leaving directory '/home/diego/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'
C++ language support library DONE
mkmmarte: work done :-)
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/utls$
```

Figura 20. Finalización de la instalación.

- Hemos realizado con éxito la instalación del sistema operativo por lo que solo queda demostrarlo corriendo un programa sobre dicho sistema. La carpeta de marte tiene algunos ejemplos en C que podemos compilar para correrlo sobre el emulador *qemu* el cual tenemos que tener instalado (en dado caso que no, solo es necesario ejecutar en el directorio raíz la instrucción *sudo apt-get install qemu*). Para correr *qemu* necesitaremos un software llamado Xming el cual podemos descargar de la siguiente liga: <https://sourceforge.net/projects/xming/files/latest/download> . La descarga se realizará automáticamente.

The screenshot shows the SourceForge website interface. At the top, there's a navigation bar with 'Open Source Software', 'Business Software', 'Services', and 'Resources'. Below this, a search bar and social media icons are visible. The main content area features a large 'Thank you for downloading Xming X Server for Windows' message. Below this, there's a 'Spread the Word' section with social media icons. The 'Keep Me Updated!' section contains a form with fields for 'Enter your email address', 'Full name', 'Phone', 'Ext', 'Job Title', 'Industry', 'Company', and 'Company Size'. A checkbox for 'I agree to receive these communications from SourceForge.net' is also present. At the bottom, a taskbar shows the Xming application running.

Figura 21. Instalación de Xming.

- Procedemos a instalarlo de la manera habitual. Al final de la instalación no esperamos que se nos despliegue ninguna ventana, sin embargo sabremos que el software está corriendo por el ícono en nuestra barra de tareas.



Figura 22. XMLing corriendo.

- Procedemos a ubicarnos en la carpeta de ejemplos del directorio de marte. Recordemos que nos encontrábamos en la carpeta *utils* por lo que para colocarnos en *examples* es necesario ejecutar lo siguiente.

```
$ cd ..
```

```
$ cd examples/
```

- Enlistamos los elementos del directorio y nos encontraremos un archivo con extensión *.c* con el nombre *hello_world_c.c*.

```
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/utils$ cd ..
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017$ cd examples/
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$ ls
ada          drivers      hello_world.adb  logger        posix        time_measurement
appsched     games       hello_world_c.c  Makefile      README      widgets
clock_modulation hardware_interrupts hello_world_cc.cc oscilloscope  speaker
```

Figura 23. Carpeta examples.

- Compilaremos el archivo ejecutando la instrucción *mgcc hello_world_c.c*, después verificaremos que la compilación haya sido exitosa enlistando de nuevo los elementos de la carpeta y encontraremos un archivo llamado *a.out*.

```
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$ mgcc hello_world_c.c
Use of uninitialized value in concatenation (.) or string at /home/diego/myapps/marte_2.0_22Feb2017/uti
ls/globals.pl line 16.
gcc -nostdinc -I/home/diego/myapps/marte_2.0_22Feb2017/arch/include hello_world_c.c -m32 -march=i686
/home/diego/myapps/marte_2.0_22Feb2017/arch/call_main/wrapper_main.o -Wl,-T,/home/diego/myapps
/marte_2.0_22Feb2017/utils/linker.lds -static -nostartfiles -L/home/diego/myapps/marte_2.0_22Feb2017/li
b -L/home/diego/myapps/marte_2.0_22Feb2017/gnat_rts/rts/adalib -L/home/diego/myapps/gnat/lib/gcc/x86_64
-pc-linux-gnu/4.9.4 -lmarte -lgnarl -lgnat -lmarte -lgcc_sjlj
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$ ls
ada          clock_modulation hardware_interrupts hello_world_cc.cc oscilloscope  speaker
a.out        drivers          hello_world.adb    logger          posix         time_measurement
appsched     games           hello_world_c.c    Makefile        README        widgets
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$
```

Figura 24. Compilación exitosa.

- Para generar el archivo ejecutable procedemos a escribir las siguientes instrucciones.

```
$ mgcc hello_world_c.c -o mprogram
```

```
$ make hello_world_c.exe
```

- Una vez más enlistamos los elementos y podremos observar la existencia de los archivos *mprogram* y *hello_world_c.exe*. Este último es el ejecutable que podremos correr sobre *qemu*.

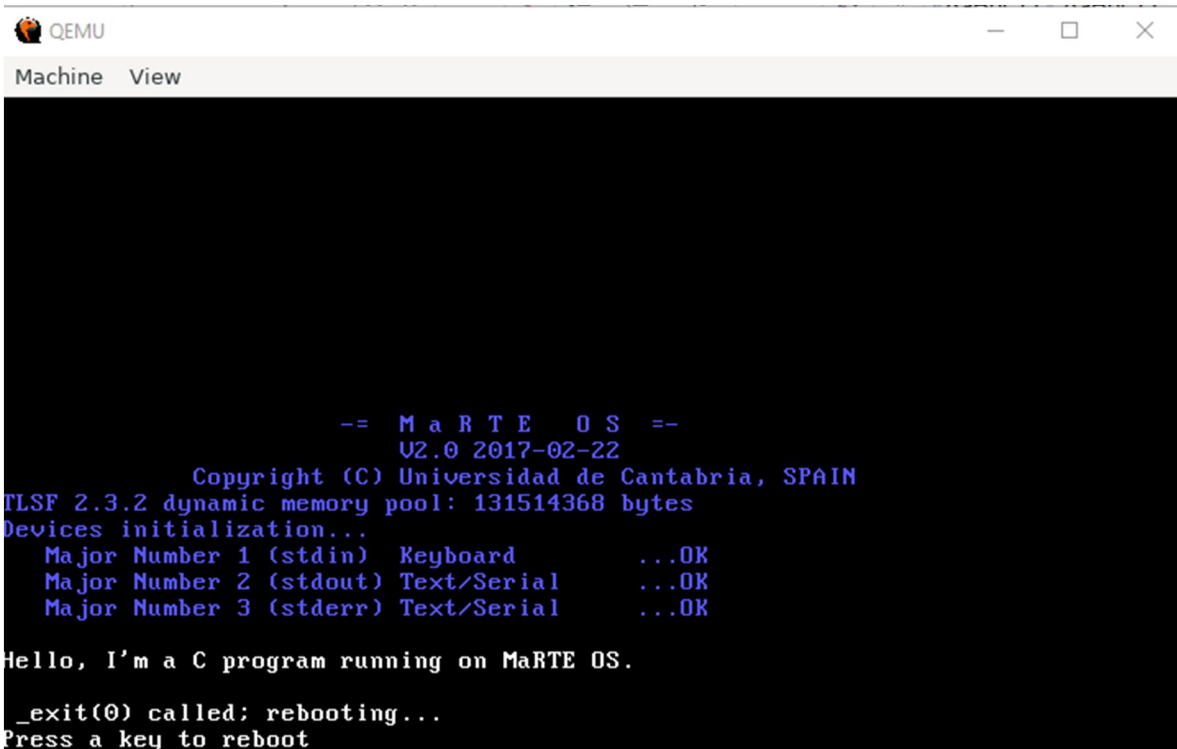
```
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$ mgcc hello_world_c.c -o mprogram
Use of uninitialized value in concatenation (.) or string at /home/diego/myapps/marte_2.0_22Feb2017/uti
ls/globals.pl line 16.
gcc -nostdinc -I/home/diego/myapps/marte_2.0_22Feb2017/arch/include hello_world_c.c -m32 -march=i68
6 -o mprogram /home/diego/myapps/marte_2.0_22Feb2017/arch/call_main/wrapper_main_c.o -Wl,-T,/home/
diego/myapps/marte_2.0_22Feb2017/utls/linker.lds -static -nostartfiles -L/home/diego/myapps/marte_2.0_
22Feb2017/lib -L/home/diego/myapps/marte_2.0_22Feb2017/gnat_rts/rts/adalib -L/home/diego/myapps/gnat/li
b/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnat -lgnat -lmarte -lgcc_sjlj
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$ make hello_world_c.exe

>> Compiling hello_world_c.exe: Use of uninitialized value in concatenation (.) or string at /home/dieg
o/myapps/marte_2.0_22Feb2017/utls/globals.pl line 16.
[OK]
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$ ls
ada          drivers      hello_world_c.c  Makefile        README
a.out        games        hello_world_cc.cc mprogram        speaker
appsched     hardware_interrupts hello_world_c.exe oscilloscope     time_measurement
clock_modulation hello_world.adb logger           posix            widgets
diego@DiegoBA:~/myapps/marte_2.0_22Feb2017/examples$
```

Figura 25. Archivos a visualizar.

- Por último corremos *qemu* con la instrucción indicada posteriormente la cual nos abrirá el emulador con nuestro programa corriendo. Es muy importante tener corriendo Xming ya que sin él no se nos desplegará el emulador.

```
$ qemu-system-i386 -kernel hello_world_c.exe
```



```
QEMU
Machine View

-- M a R T E   O S --
U2.0 2017-02-22
Copyright (C) Universidad de Cantabria, SPAIN
TLFS 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
```

Figura 26. Programa corriendo sobre MaRTEOS.

Referencias

- [1] C. University, «MaRTE OS,» [En línea]. Available: <https://martel.unican.es/>. [Último acceso: Octubre 2019].