



INSTITUTO POLITECNICO NACIONAL  
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN  
INGENIERÍA Y TECNOLOGÍAS AVANZADAS



SISTEMAS OPERATIVOS EN TIEMPO REAL  
PROFESOR: MAZA CASAS LAMBERTO

INSTALACIÓN DEL SISTEMA OPERATIVO MARTE  
OS Y EJECUCIÓN DE UNA ALARMA SOBRE ESTE  
SISTEMA

JOSÉ ALBERTO RODRÍGUEZ GARCÍA

3MV11

NOVIEMBRE 2019

## PRELIMINARES

Antes de comenzar con la instalación del sistema operativo, es necesario contar con una distribución de Linux. Para este trabajo utilizamos el sistema UBUNTU, disponible en <https://ubuntu.com/download/desktop> y que corrió como máquina virtual en el software Virtual Box de Oracle, disponible en <https://www.virtualbox.org/wiki/Downloads>.

Una vez que se tiene la máquina virtual en funcionamiento, se puede proceder con la instalación del sistema Marte OS.

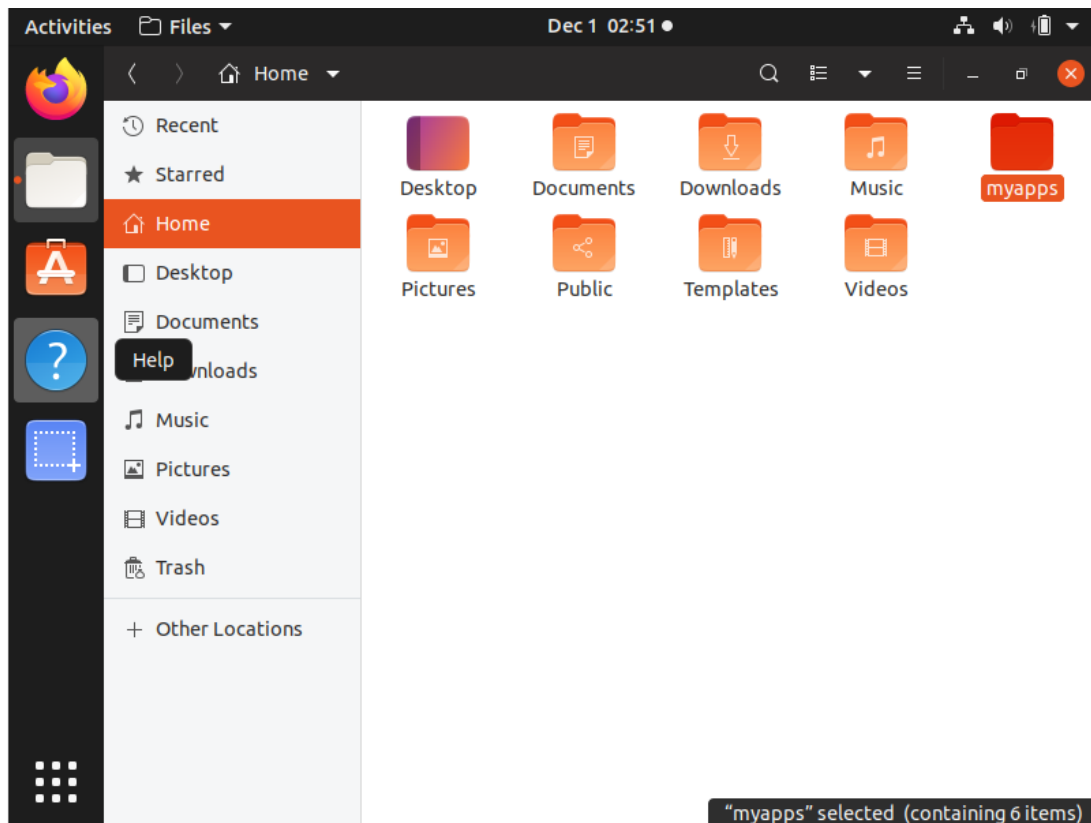
Como paso cero, es necesario descargar dos archivos, el compilador GNAT y los archivos de instalación del sistema Marte OS. Los enlaces para descargarlos se muestran a continuación.

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

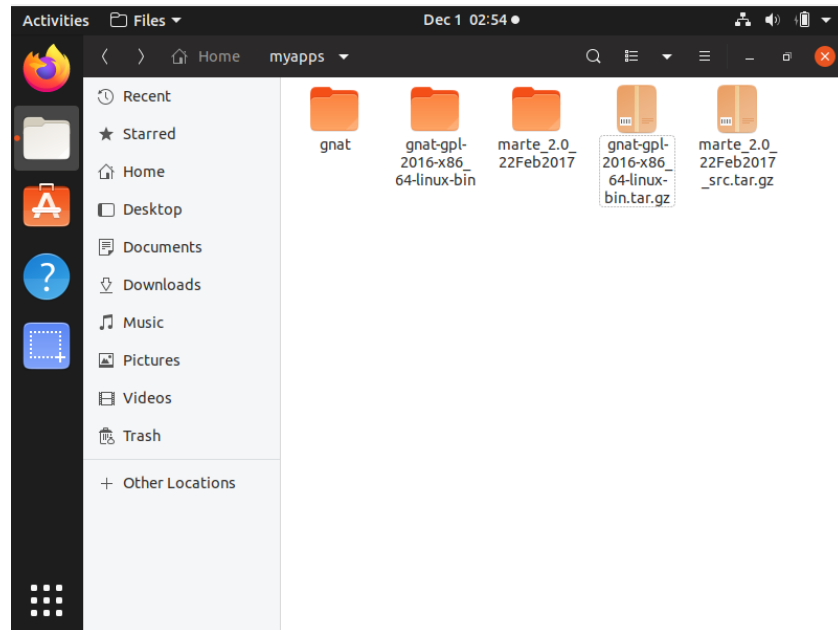
[https://marte.unican.es/marte/marte\\_2.0\\_22Feb2017\\_src.tar.gz](https://marte.unican.es/marte/marte_2.0_22Feb2017_src.tar.gz)

## INSTALACIÓN DE MARTE OS

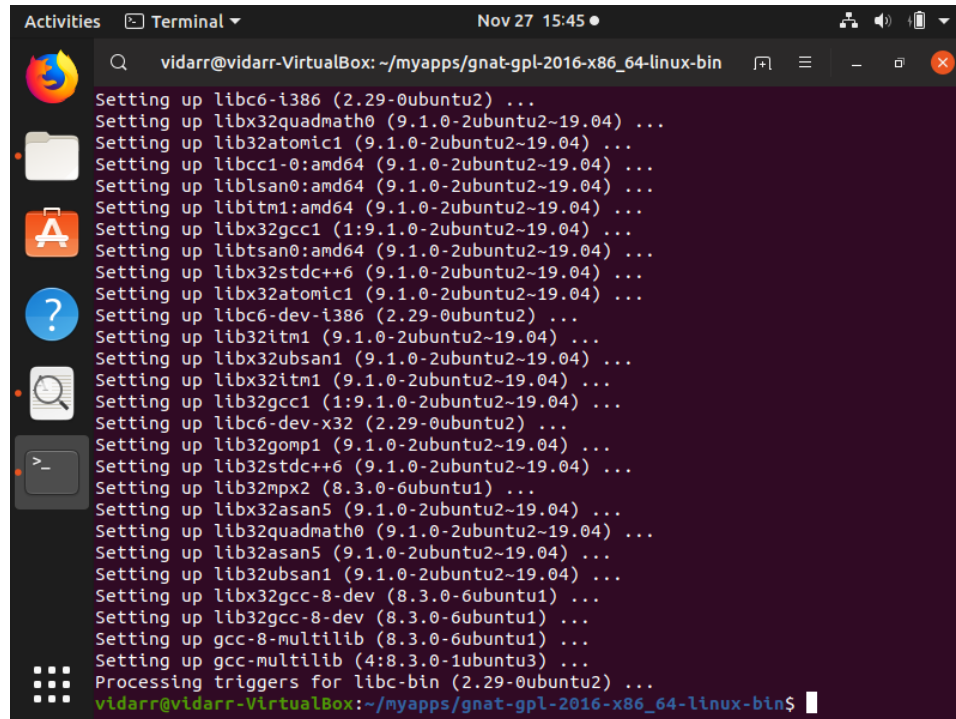
En primer lugar debemos crear una carpeta llamada myapps en el directorio home como se muestra a continuación.



Posteriormente, se deben copiar los ficheros del compilador GNAT y del sistema operativo descargados en la sección *preliminares* a la carpeta recién creada. Después de copiarlos se descomprimen, y adicionalmente se crea una carpeta llamada “gnat”. La carpeta *myapps* se deberá ver como la siguiente imagen.



Con esto listo, estamos preparados para comenzar la instalación. Accedemos a la carpeta *gnat-gpl-2016-x86-64-linux-bin* y abrimos una terminal desde ahí. Una vez en la terminal, debemos instalar dos paquetes para el correcto desarrollo de la instalación, para ello, ingresamos el comando `sudo apt-get install libc6-dev- i386`. Ingresamos la contraseña del usuario y observaremos una pantalla como la siguiente que no indica que se instaló con éxito.

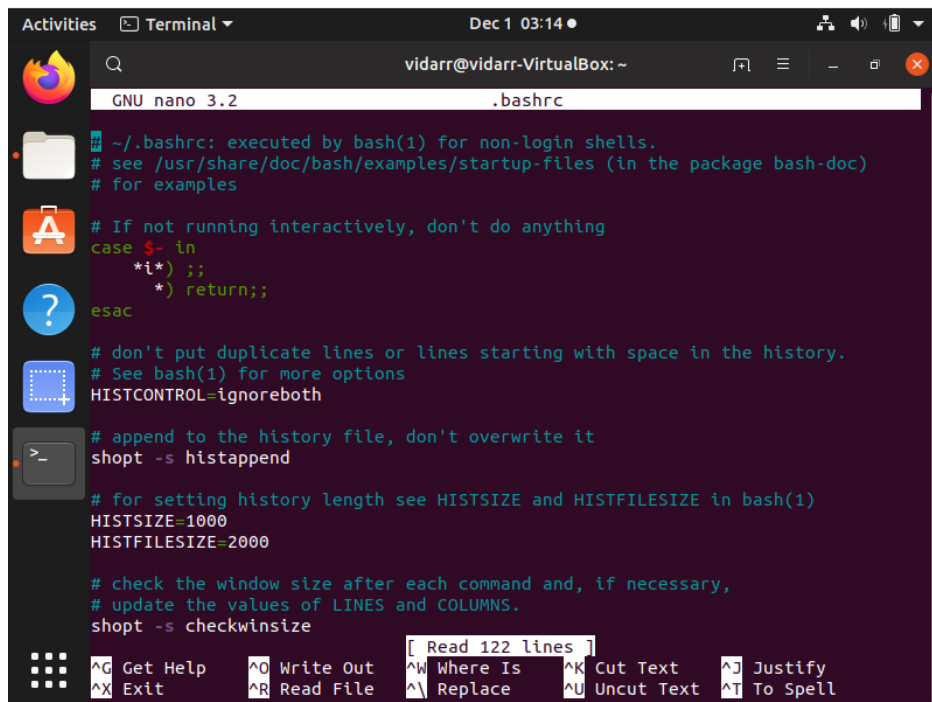


A terminal window titled "vidarr@vidarr-VirtualBox: ~/myapps/gnat-gpl-2016-x86\_64-linux-bin" showing the installation of various libraries. The output lists the following libraries being set up:

```
Setting up libc6-i386 (2.29-0ubuntu2) ...
Setting up libx32quadmath0 (9.1.0-2ubuntu2~19.04) ...
Setting up lib32atomic1 (9.1.0-2ubuntu2~19.04) ...
Setting up libgcc1-0:amd64 (9.1.0-2ubuntu2~19.04) ...
Setting up liblsan0:amd64 (9.1.0-2ubuntu2~19.04) ...
Setting up libitm1:amd64 (9.1.0-2ubuntu2~19.04) ...
Setting up libx32gcc1 (1:9.1.0-2ubuntu2~19.04) ...
Setting up libtsan0:amd64 (9.1.0-2ubuntu2~19.04) ...
Setting up libx32stdc++6 (9.1.0-2ubuntu2~19.04) ...
Setting up libx32atomic1 (9.1.0-2ubuntu2~19.04) ...
Setting up libc6-dev-i386 (2.29-0ubuntu2) ...
Setting up lib32itm1 (9.1.0-2ubuntu2~19.04) ...
Setting up libx32ubsan1 (9.1.0-2ubuntu2~19.04) ...
Setting up libx32itm1 (9.1.0-2ubuntu2~19.04) ...
Setting up lib32gcc1 (1:9.1.0-2ubuntu2~19.04) ...
Setting up libc6-dev-x32 (2.29-0ubuntu2) ...
Setting up lib32gomp1 (9.1.0-2ubuntu2~19.04) ...
Setting up lib32stdc++6 (9.1.0-2ubuntu2~19.04) ...
Setting up lib32mpx2 (8.3.0-6ubuntu1) ...
Setting up libx32asan5 (9.1.0-2ubuntu2~19.04) ...
Setting up lib32quadmath0 (9.1.0-2ubuntu2~19.04) ...
Setting up lib32asan5 (9.1.0-2ubuntu2~19.04) ...
Setting up lib32ubsan1 (9.1.0-2ubuntu2~19.04) ...
Setting up libx32gcc-8-dev (8.3.0-6ubuntu1) ...
Setting up lib32gcc-8-dev (8.3.0-6ubuntu1) ...
Setting up gcc-8-multilib (8.3.0-6ubuntu1) ...
Setting up gcc-multilib (4:8.3.0-1ubuntu3) ...
Processing triggers for libc-bin (2.29-0ubuntu2) ...
```

The prompt is now `vidarr@vidarr-VirtualBox:~/myapps/gnat-gpl-2016-x86_64-linux-bin$`.

Ahora es necesario realizar un preparativo más. Se ingresa el comando `cd`, con esto nos movemos al directorio raíz. Posteriormente se ingresa el comando `nano .bashrc`. Se mostrará una pantalla como la que sigue:



A terminal window titled "vidarr@vidarr-VirtualBox: ~" showing the `nano` editor editing the `.bashrc` file. The editor title bar says "GNU nano 3.2 .bashrc". The content of the file is as follows:

```
~/.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
HISTFILESIZE=2000

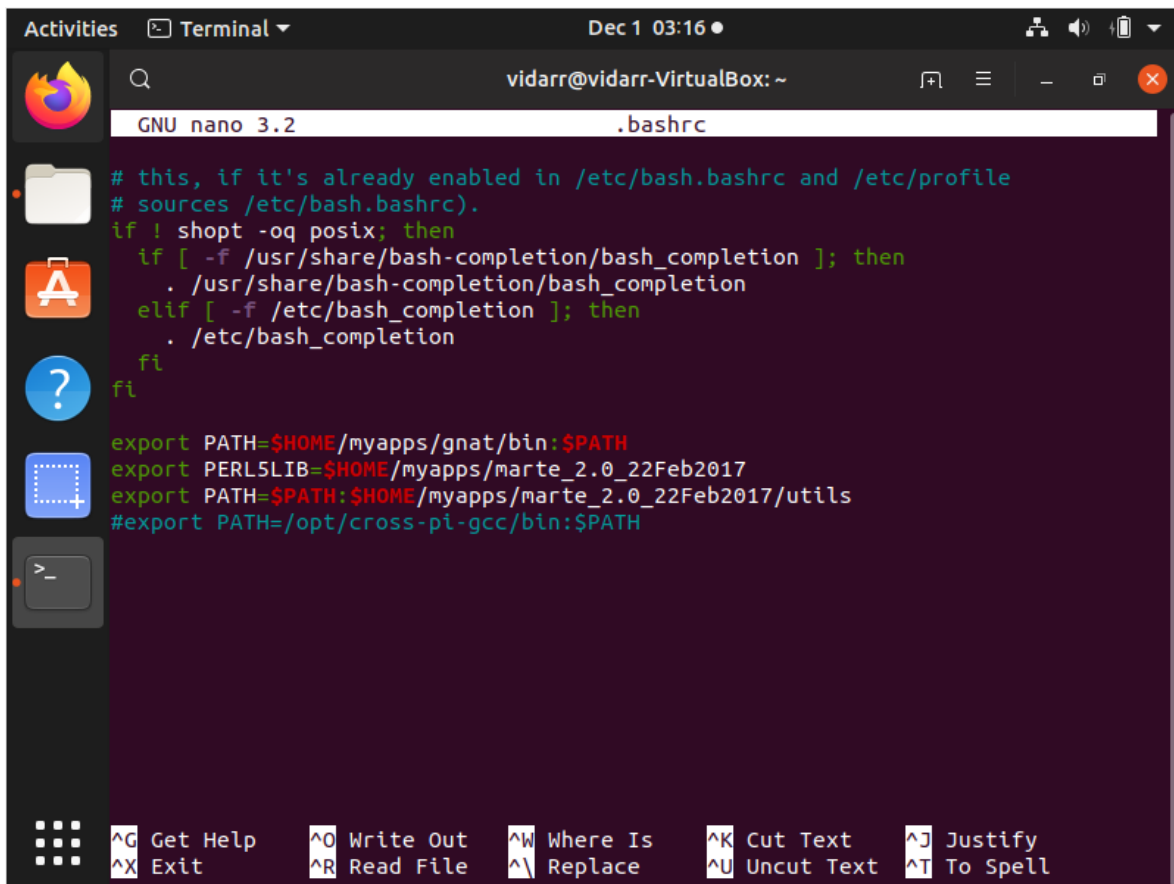
# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize
```

The bottom of the screen shows the nano editor's help menu:

```
[ Read 122 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^_ Replace       ^U Uncut Text   ^T To Spell
```

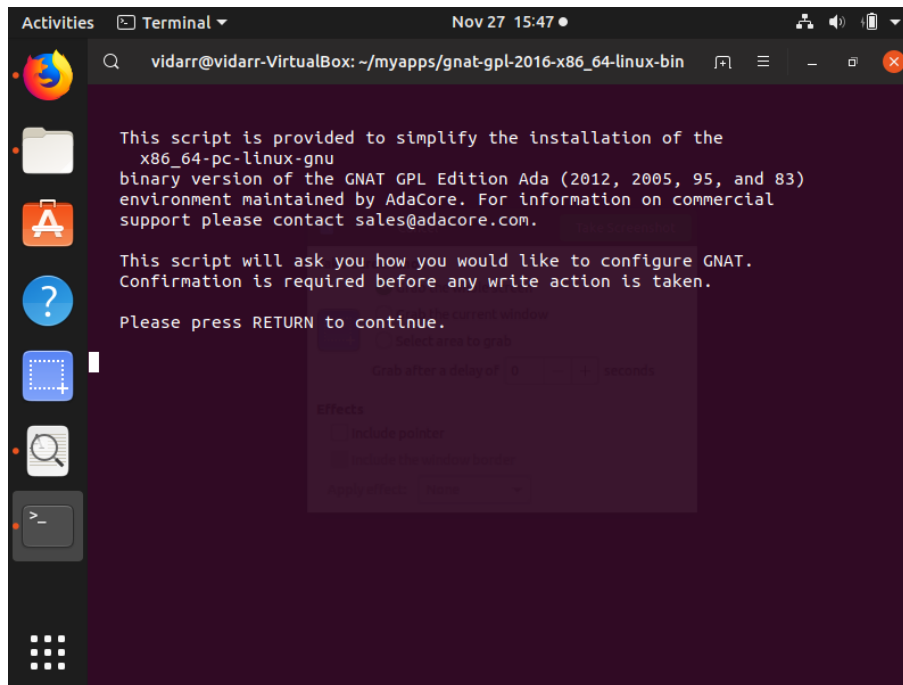
Nos desplazamos hasta el final del archivo y añadimos las siguientes líneas:

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



```
Activities  Terminal  Dec 1 03:16  vidarr@vidarr-VirtualBox: ~  
GNU nano 3.2 .bashrc  
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc).  
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 PERLSLIB=$HOME/myapps/marte_2.0_22Feb2017  
export PATH=$PATH:$HOME/myapps/marte_2.0_22Feb2017/Utils  
#export PATH=/opt/cross-pi-gcc/bin:$PATH  
  
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

Una vez hecho esto, guardamos los cambios con la combinación de teclas **Ctrl+o** y salimos con la combinación **Ctrl+x**. Es necesario cerrar la terminal y volverla a abrir para que se guarden los cambios. Para abrir la terminal, se regresa a la carpeta de *gnat-gpl-2016-x86-64-linux-bin* y se abre la terminal de nuevo. Posteriormente escribimos el siguiente comando: *./doinstall*. Al hacerlo, se desplegará la siguiente información en la terminal:



Activities Terminal Nov 27 15:47

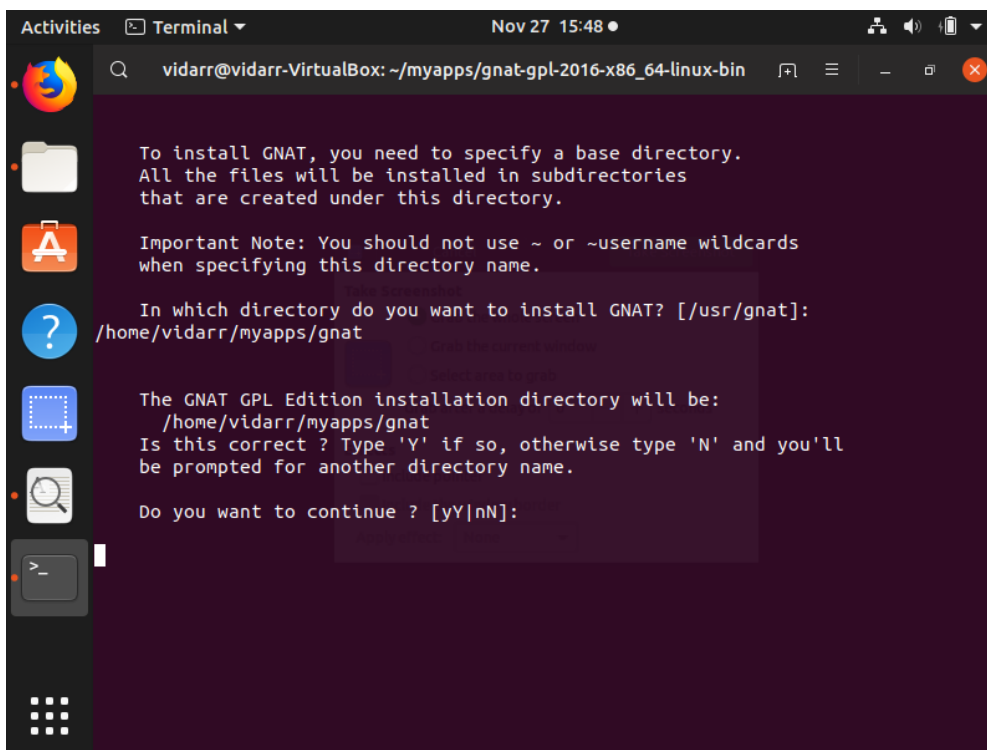
vidarr@vidarr-VirtualBox: ~/myapps/gnat-gpl-2016-x86\_64-linux-bin

```
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.
```

Al presionar enter como nos indica la pantalla, observaremos el siguiente texto en la terminal. Nos pedirá que ingresemos donde queremos instalar el compilador. Ingresamos la ruta de la carpeta que creamos dentro de myapps llamada gnat, como se observa en la imagen y presionamos enter. Después de esto, deberíamos ver la siguiente información.



Activities Terminal Nov 27 15:48

vidarr@vidarr-VirtualBox: ~/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.

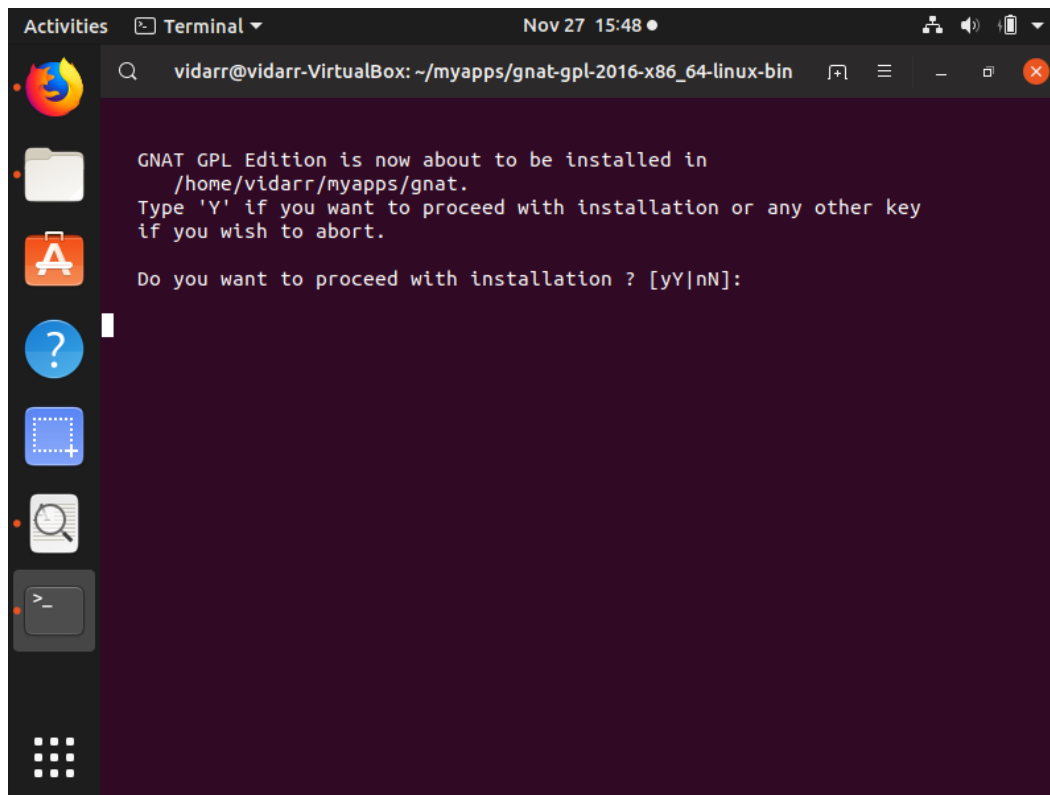
Important Note: You should not use ~ or ~username wildcards
when specifying this directory name.

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

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

Escribimos “Y” y enter. Al hacerlo, se comenzará el proceso de instalación. Habrá una última pantalla que nos pide confirmar que queremos realizar la instalación. Confirmamos que así es, y el proceso comenzará.



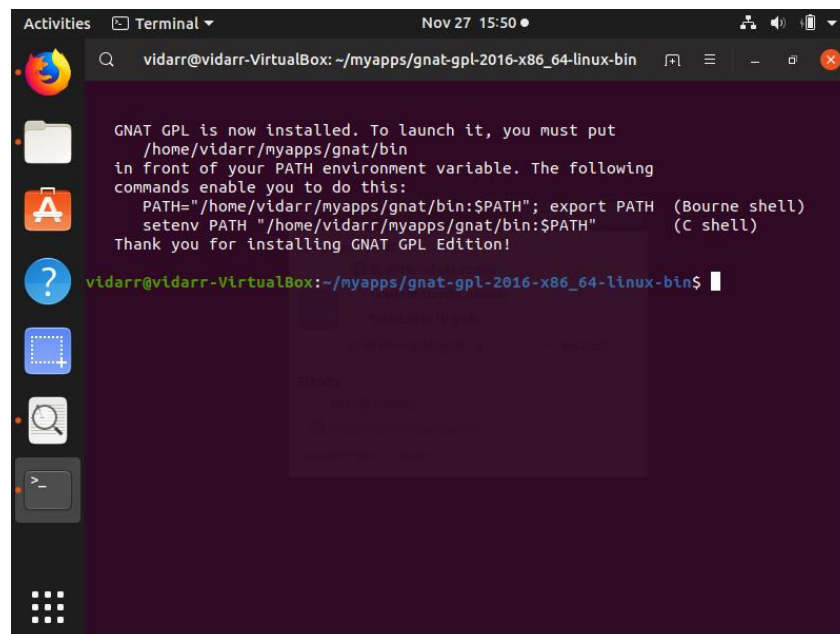
A terminal window titled "vidarr@vidarr-VirtualBox: ~/myapps/gnat-gpl-2016-x86\_64-linux-bin" with a search bar and window controls. The terminal text is as follows:

```
GNAT GPL Edition is now about to be installed in
/home/vidarr/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]:
```

A cursor is visible on the line following the prompt.

Este toma algunos minutos. Al finalizar la terminal indicará lo siguiente y de esta manera sabremos que la instalación fue correcta:

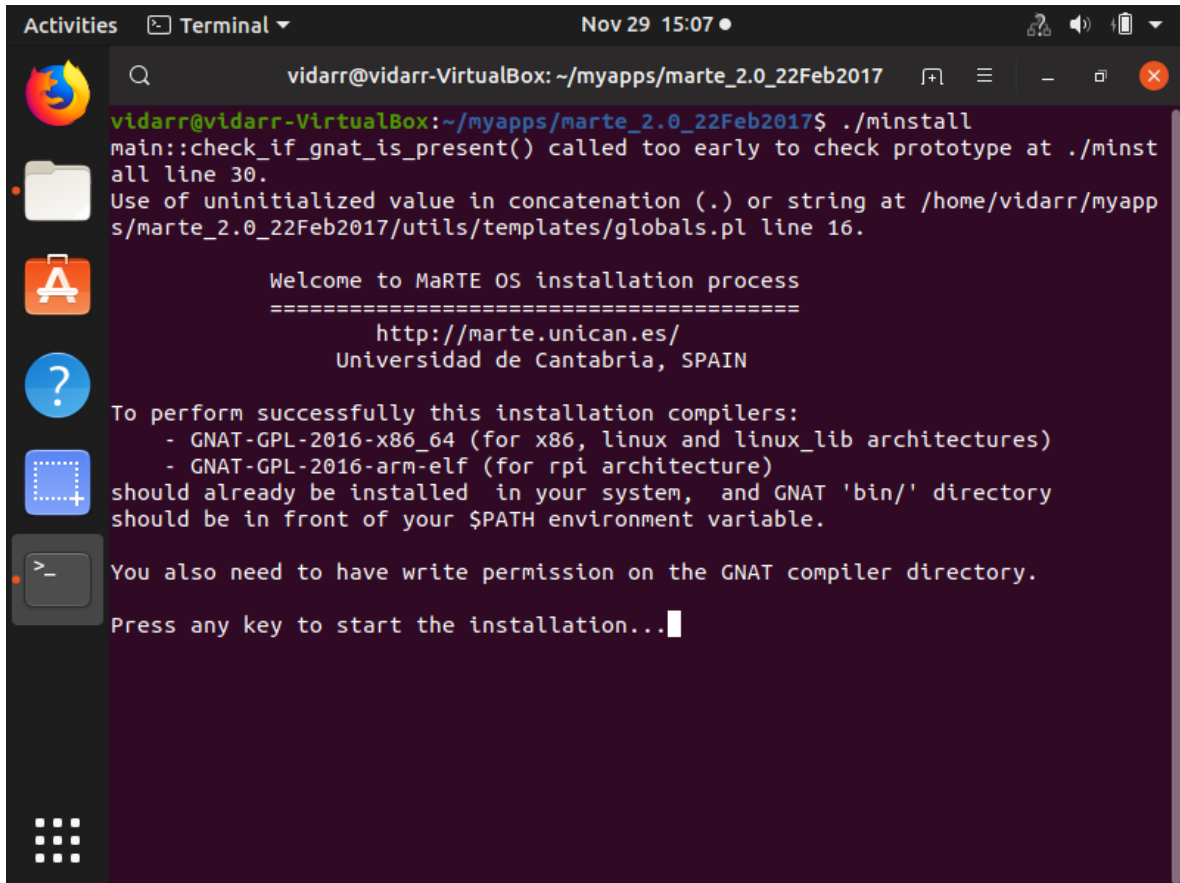


A terminal window titled "vidarr@vidarr-VirtualBox: ~/myapps/gnat-gpl-2016-x86\_64-linux-bin" with a search bar and window controls. The terminal text is as follows:

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

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

Ahora es momento de construir el sistema operativo. Es necesario acceder a la carpeta del compilador y abrir una terminal desde ahí o bien navegar desde la terminal mediante el comando `cd`. Una vez que la terminal está en este directorio, procedemos a instalar el sistema operativo ingresando el comando `./minstall`. Se desplegará la siguiente pantalla.



The screenshot shows a terminal window titled "vidarr@vidarr-VirtualBox: ~/myapps/marte\_2.0\_22Feb2017". The user has executed the command `./minstall`. The terminal output shows an error message from a previous run, followed by a welcome message for the MaRTE OS installation process. The welcome message includes the URL <http://marte.unican.es/> and identifies the user as being from the Universidad de Cantabria, SPAIN. It then lists the required compilers: GNAT-GPL-2016-x86\_64 for x86, linux and linux\_lib architectures, and GNAT-GPL-2016-arm-elf for the rpi architecture. It also states that these compilers should already be installed and that the GNAT 'bin/' directory should be in the user's \$PATH. Finally, it asks the user to press any key to start the installation.

```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017$ ./minstall
main::check_if_gnat_is_present() called too early to check prototype at ./minst
all line 30.
Use of uninitialized value in concatenation (.) or string at /home/vidarr/myapp
s/marte_2.0_22Feb2017/utils/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...
```

Como indica la terminal, presionamos cualquier tecla y el proceso de instalación comienza. Este es un proceso rápido. Al terminar, se despliega el siguiente mensaje.



```
Activities Terminal Nov 29 15:07
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017

Change gnat-gpl-2016 libraries for the 32bits versions
Use of uninitialized value in concatenation (.) or string at /home/vidarr/myapps/marte_2.0_22Feb2017/utlis/globals.pl line 16.
Set 32bits libs in /home/vidarr/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4

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

You may want to add "/home/vidarr/myapps/marte_2.0_22Feb2017/utlis"
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:

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

This is a MaRTE source distribution, so you must compile MaRTE libraries
before using them. For example, for "x86" architecture execute:

$ msetcurrentarch x86 && mkrtsmarteuc && mkmarte

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

vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$
```

El siguiente paso es ejecutar el comando *msetcurrentarch x86-i386* Como se muestra en la siguiente imagen, al ejecutar sólo *msetcurrentarch* nos indica para que sistemas es posible compilar este sistema.

```
Activities Terminal Nov 29 15:08
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017

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

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

This is a MaRTE source distribution, so you must compile MaRTE libraries
before using them. For example, for "x86" architecture execute:

$ msetcurrentarch x86 && mkrtsmarteuc && mkmarte

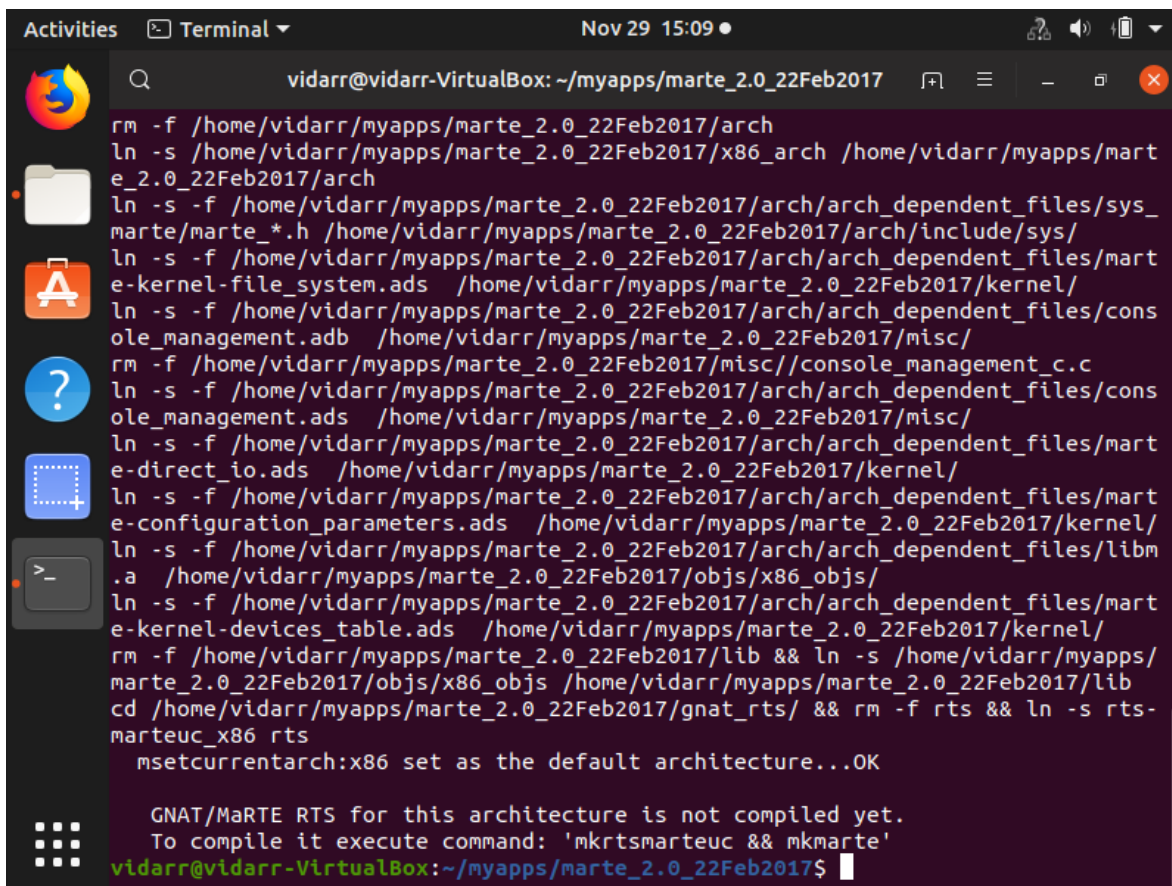
For more information go to chapter 1.2 of the 'INSTALL' document

vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$ msetcurrentarch
Use of uninitialized value in concatenation (.) or string at /home/vidarr/myapps/marte_2.0_22Feb2017/utlis/globals.pl line 16.
Current architecture:none

Available architectures status:
x86:      RTS (gnat_rts/rts-marteuc_x86): NOT Compiled
          Lib MaRTE (objs/x86_objjs):      NOT Compiled
linux:    RTS (gnat_rts/rts-marteuc_linux): NOT Compiled
          Lib MaRTE (objs/linux_objjs):     NOT Compiled
linux_lib: RTS (gnat_rts/rts-marteuc_linux_lib): NOT Compiled
          Lib MaRTE (objs/linux_lib_objjs): NOT Compiled
rpi:      NOT available

vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$
```

Al finalizar este proceso, nos desplegará la siguiente información.



```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017
rm -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch
ln -s /home/vidarr/myapps/marte_2.0_22Feb2017/x86_arch /home/vidarr/myapps/marte_2.0_22Feb2017/arch
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/sys_marte/marte_*.h /home/vidarr/myapps/marte_2.0_22Feb2017/arch/include/sys/
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-kernel-file_system.ads /home/vidarr/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/console_management.adb /home/vidarr/myapps/marte_2.0_22Feb2017/misc/
rm -f /home/vidarr/myapps/marte_2.0_22Feb2017/misc//console_management_c.c
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/console_management.ads /home/vidarr/myapps/marte_2.0_22Feb2017/misc/
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-direct_io.ads /home/vidarr/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-configuration_parameters.ads /home/vidarr/myapps/marte_2.0_22Feb2017/kernel/
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/libm.a /home/vidarr/myapps/marte_2.0_22Feb2017/objs/x86_objs/
ln -s -f /home/vidarr/myapps/marte_2.0_22Feb2017/arch/arch_dependent_files/marte-kernel-devices_table.ads /home/vidarr/myapps/marte_2.0_22Feb2017/kernel/
rm -f /home/vidarr/myapps/marte_2.0_22Feb2017/lib && ln -s /home/vidarr/myapps/marte_2.0_22Feb2017/objs/x86_objs /home/vidarr/myapps/marte_2.0_22Feb2017/lib
cd /home/vidarr/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'
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$
```

Como indica la pantalla, hay que ejecutar dos comandos más: *mkrtsmarteuc* y *mkmarte*. Al ejecutar el primero, debemos visualizar la siguiente pantalla al finalizar.

```
Activities Terminal Nov 29 15:13
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017
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-thread.o s-inmaop.o s-interr.o s-intman.o s-mudido.o s-osinte.o s-proinf.o s-solita.o s-stusta.o s-taenca.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-tasde.o s-tadec.a.o s-tadert.o s-tataat.o s-tpinop.o s-tpoben.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-dlroro.o
ar: creating libgnarl.a
rm *.o
chmod 0444 *.ali *.a
cd /home/vidarr/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ && ln -f -s ../../../../lib/libmarte.a libmarte.a
cd /home/vidarr/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ && ln -f -s libmarte.a libc.a
cd /home/vidarr/myapps/marte_2.0_22Feb2017/gnat_rts/rts-marteuc_x86/adalib/ && ln -f -s ../../../../lib/multiboot.o multiboot.o
cd /home/vidarr/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && rm adainclude adalib
cd /home/vidarr/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 && ln -s rts-native/adainclude adainclude
cd /home/vidarr/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'
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$
```

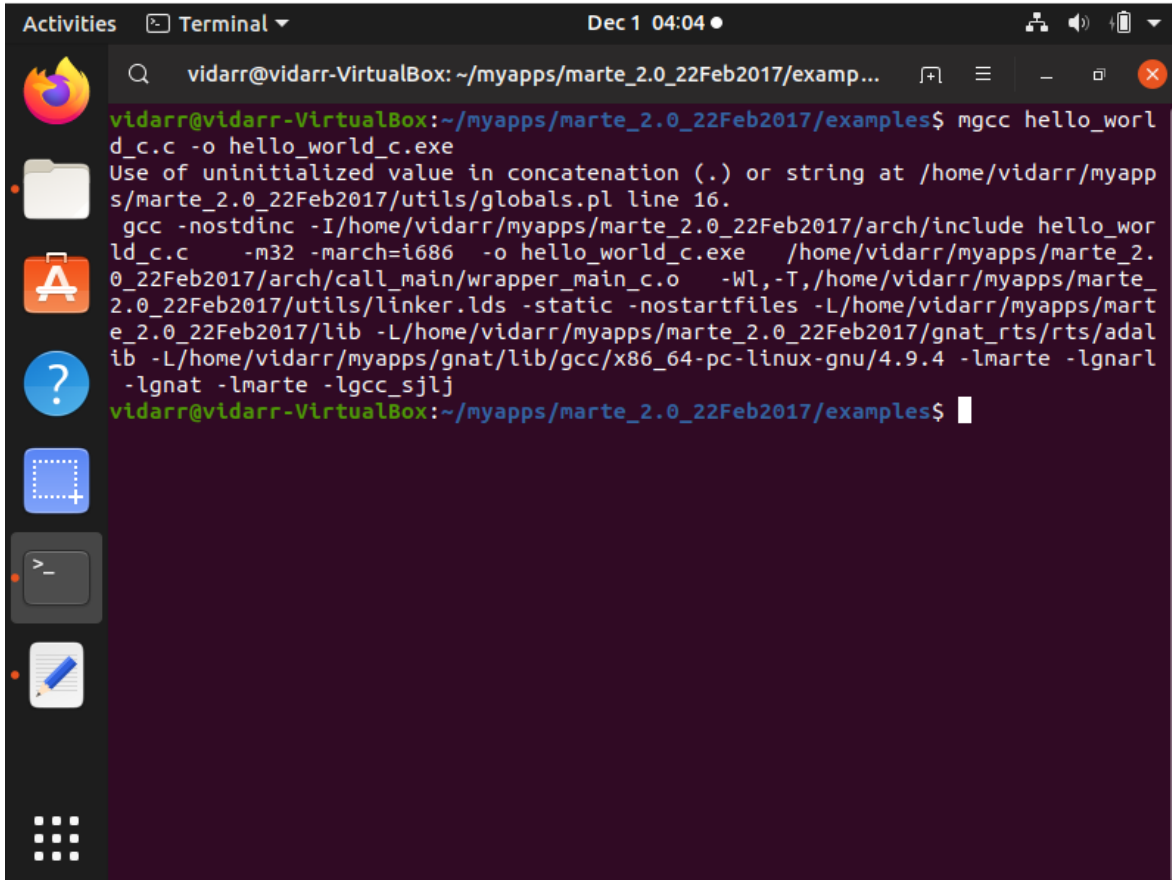
Después se ingresa mkmarte y al finalizar, se deberá visualizar la siguiente pantalla:

```
Activities Terminal Dec 1 03:30
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017
cd /home/vidarr/myapps/marte_2.0_22Feb2017/lang_support && export MARCH=x86 && make
make -C cxx/libsupc++/ all
make[1]: Entering directory '/home/vidarr/myapps/marte_2.0_22Feb2017/lang_support/cxx/libsupc++'
ar -r libsupc++.a eh_throw.o new_opvnt.o pure.o new_opnt.o tinfo.o eh_terminate.o eh_personality.o eh_catch.o eh_exception.o eh_unex_handler.o eh_globals.o del_opnt.o new_handler.o del_opvnt.o new_op.o eh_term_handler.o del_opv.o tinfo2.o eh_alloc.o new_opv.o del_op.o eh_aux_runtime.o
cp libsupc++.a ../../../../lib
make[1]: Leaving directory '/home/vidarr/myapps/marte_2.0_22Feb2017/lang_support/cxx/libsupc++'
make -C ustl-src/ all
make[1]: Entering directory '/home/vidarr/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/vidarr/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'
make -C ustl-src/ install
make[1]: Entering directory '/home/vidarr/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/vidarr/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'
C++ language support library DONE

mkmarte: work done :-)
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$
```

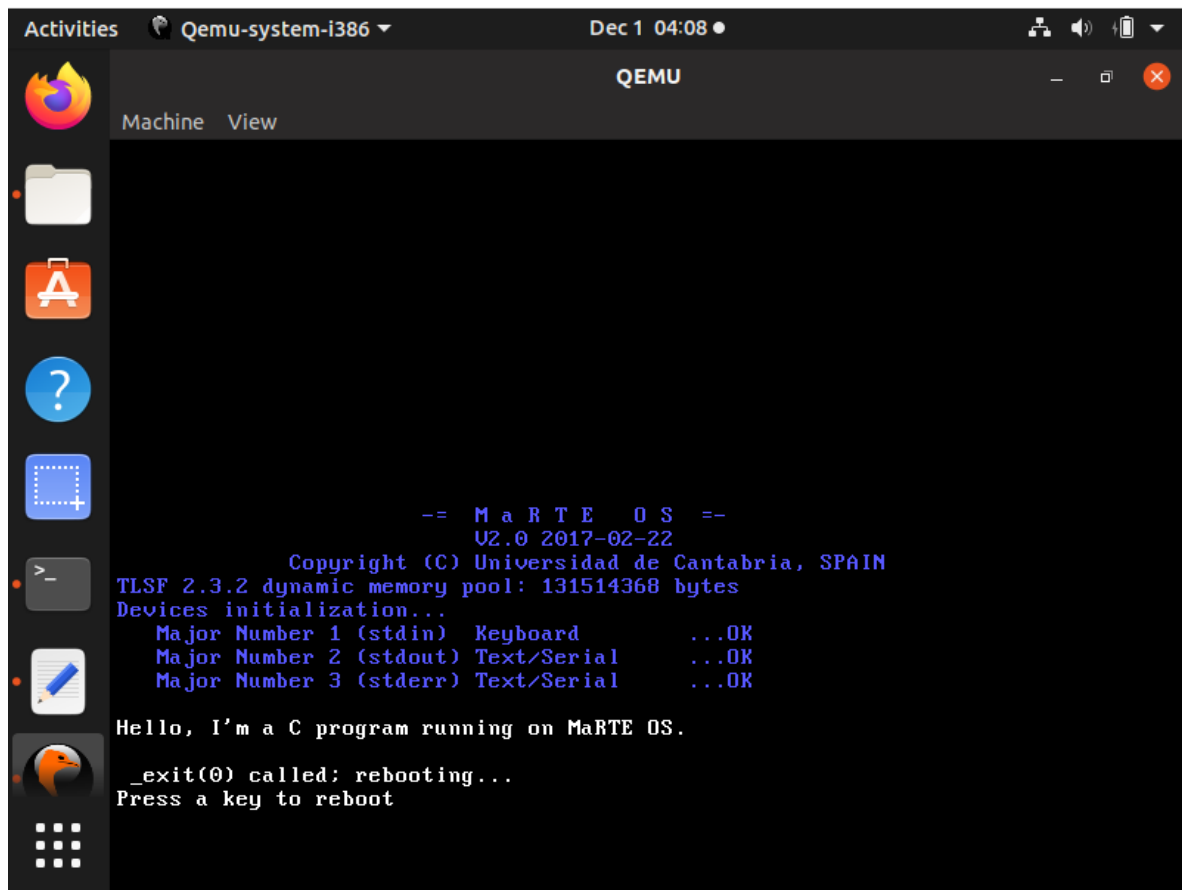
Y con este mensaje, se indica que el proceso de instalación y compilación de Marte Os ha terminado.

Para probar que se instaló correctamente, podemos usar un programa de prueba. Para ello, entramos a la carpeta *examples* mediante el comando *cd examples*. Una vez ahí, ejecutamos el comando *mgcc hello\_world\_c.c -o hello\_world\_c.exe*. Al hacerlo, se mostrará una pantalla como sigue:

A screenshot of a terminal window titled "vidarr@vidarr-VirtualBox: ~/myapps/marte\_2.0\_22Feb2017/examp...". The terminal shows the command `mgcc hello_world_c.c -o hello_world_c.exe` being executed. The output includes a warning: "Use of uninitialized value in concatenation (.) or string at /home/vidarr/myapps/marte\_2.0\_22Feb2017/utils/globals.pl line 16." followed by the compilation command: `gcc -nostdinc -I/home/vidarr/myapps/marte_2.0_22Feb2017/arch/include hello_world_c.c -m32 -march=i686 -o hello_world_c.exe /home/vidarr/myapps/marte_2.0_22Feb2017/arch/call_main/wrapper_main_c.o -Wl,-T,/home/vidarr/myapps/marte_2.0_22Feb2017/utils/linker.lds -static -nostartfiles -L/home/vidarr/myapps/marte_2.0_22Feb2017/lib -L/home/vidarr/myapps/marte_2.0_22Feb2017/gnat_rts/rts/adalib -L/home/vidarr/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnat -lgcc_sjlj`. The terminal ends with the prompt `vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples$`.

```
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples$ mgcc hello_world_c.c -o hello_world_c.exe
Use of uninitialized value in concatenation (.) or string at /home/vidarr/myapps/marte_2.0_22Feb2017/utils/globals.pl line 16.
gcc -nostdinc -I/home/vidarr/myapps/marte_2.0_22Feb2017/arch/include hello_world_c.c -m32 -march=i686 -o hello_world_c.exe /home/vidarr/myapps/marte_2.0_22Feb2017/arch/call_main/wrapper_main_c.o -Wl,-T,/home/vidarr/myapps/marte_2.0_22Feb2017/utils/linker.lds -static -nostartfiles -L/home/vidarr/myapps/marte_2.0_22Feb2017/lib -L/home/vidarr/myapps/marte_2.0_22Feb2017/gnat_rts/rts/adalib -L/home/vidarr/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnat -lgcc_sjlj
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples$
```

Podemos simular este código mediante la herramienta qemu. Para hacerlo, primero debemos verificar que esté instalado este paquete, o bien directamente ingresar los siguientes comandos *sudo apt-get install qemu*. Posteriormente, ingresamos el comando *sudo apt-get install qemu-system-i386*. Una vez finalizados estos procesos, ejecutamos el siguiente comando: *qemu-system-i386 -kernel hello\_world\_c.exe*. Con esto, se abrirá una nueva ventana y observaremos lo siguiente:



## TEMPORIZADOR SOBRE MARTE OS

Una vez finalizado el proceso anterior, se puede continuar con la compilación de un programa orientado a Marte OS. El primer paso será acceder a la carpeta *examples*, que se encuentra dentro de la carpeta *marke\_2.0\_22Feb2017*. Dentro de este directorio vamos a crear el archivo que contendrá el código fuente de nuestro programa. Para hacerlo, abrimos una terminal e ingresamos el comando *nano tempo.c*. En este contexto, capturamos el siguiente código:

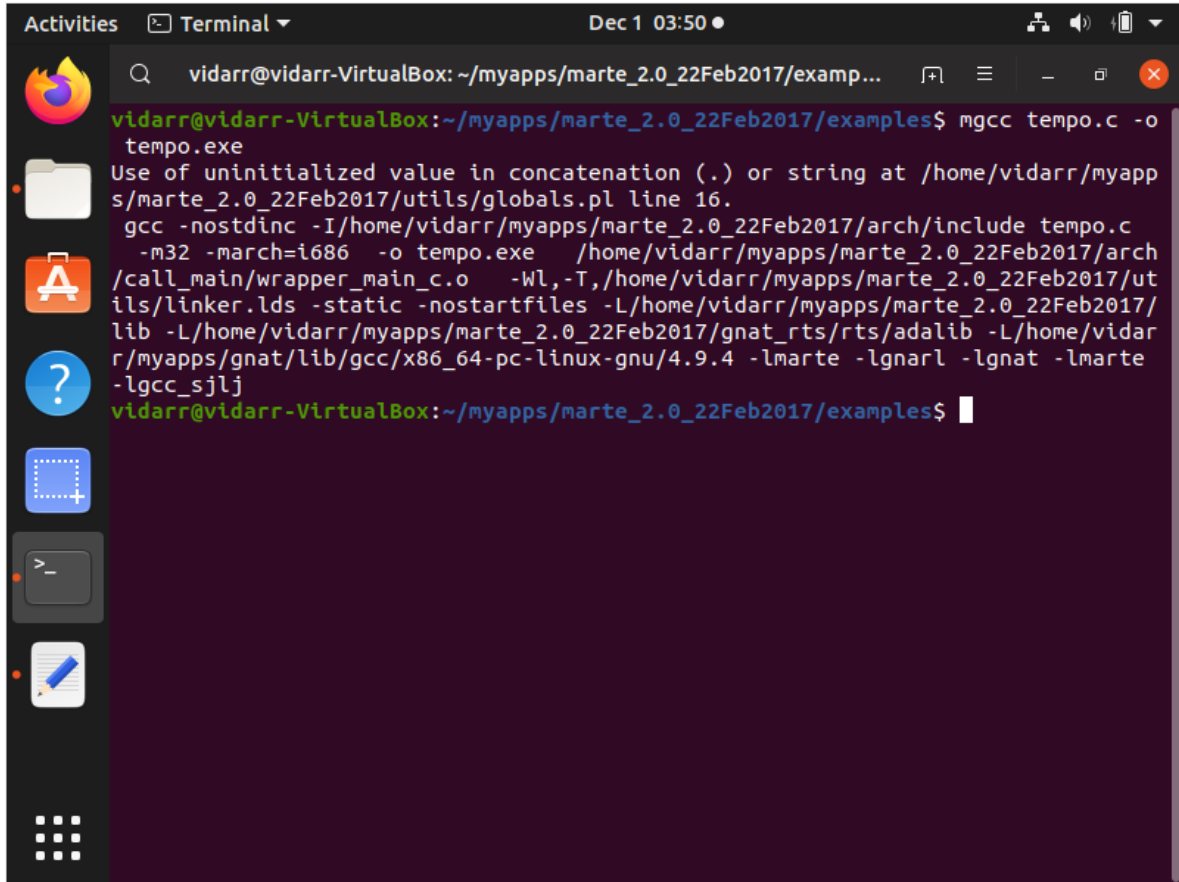
```
#include<stdio.h>
#include<time.h>//FUNCIONES DE TIEMPO
#include<unistd.h>//FUNCIONES PARA PAUSAR

int main()
{
    int min=0;//VARIABLE PARA MINUTOS
    int n=0;//VARIABLE PARA EL CONTEO
    int i=0;//VARIABLE AUXILIAR PARA DESPLIEGUE DE INFO
    printf("TEMPORIZADOR\n");//MENSAJE INICIAL
    printf("INTRODUCE EL TIEMPO EN MINUTOS");//INTERACCIÓN CON EL USUARIO
    scanf("%d",&min);

    while(n<=(60*min))//BUCLE DE CONTEO
    {
        sleep(1);//PAUSA DE 1 SEGUNDO
        if(n%60==0)//CADA MINUTO
        {
            i=n/60;//DESPLIEGA CUANTOS MINUTOS VAN
            printf("%d MINUTOS\n",i);
        }
        n=n+1;
    }

    printf("TERMINO EL TIEMPO, DESPIERTA!!!");//AL FINAL, MANDA UN MENSAJE AL USUARIO
    return 0;
}
```

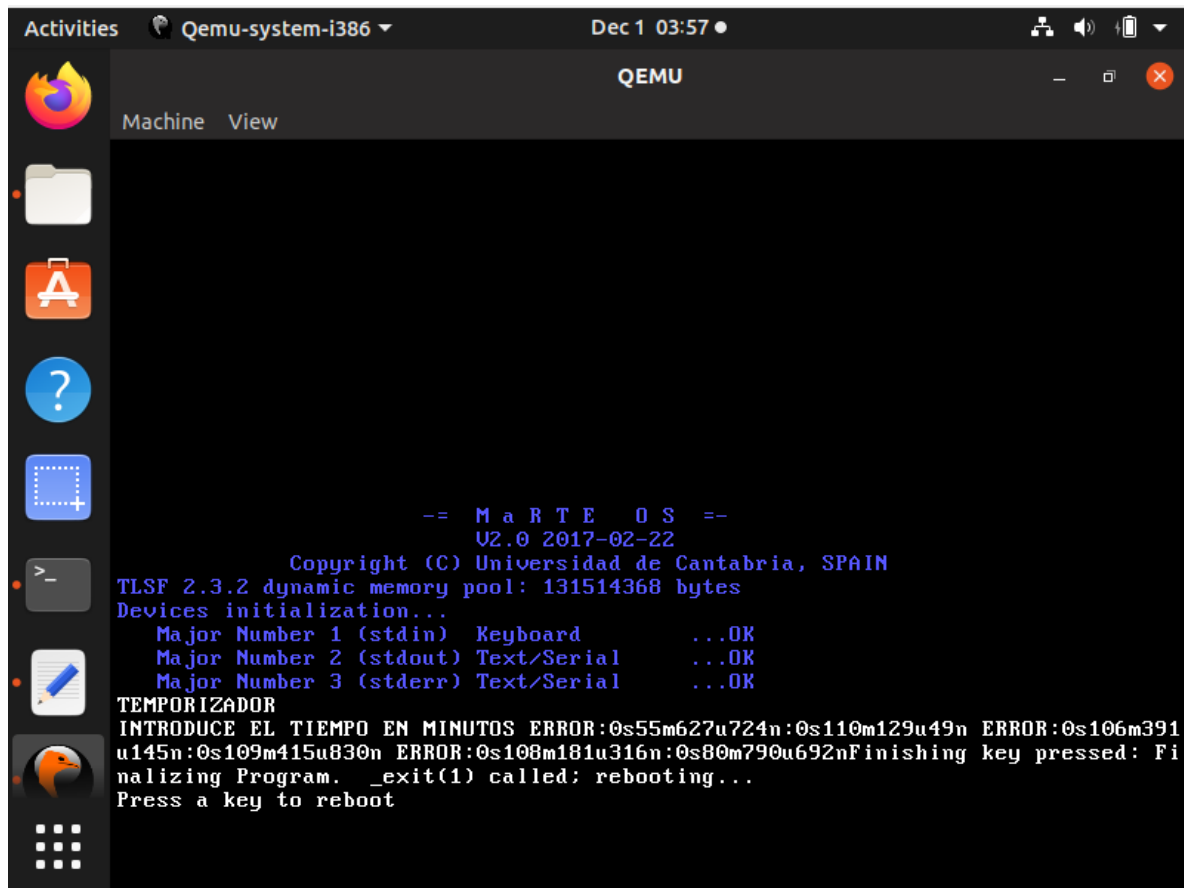
Una vez capturado el código, guardamos los cambios y cerramos el documento. Al regresar a la terminal, ahora podemos realizar la compilación del código. Para ello, escribimos el siguiente comando `mgcc tempo.c -o tempo.exe`. Observaremos la siguiente pantalla:



```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017/examples$ mgcc tempo.c -o
tempo.exe
Use of uninitialized value in concatenation (.) or string at /home/vidarr/myapp
s/marte_2.0_22Feb2017/utlis/globals.pl line 16.
gcc -nostdinc -I/home/vidarr/myapps/marte_2.0_22Feb2017/arch/include tempo.c
-m32 -march=i686 -o tempo.exe /home/vidarr/myapps/marte_2.0_22Feb2017/arch
/call_main/wrapper_main_c.o -WL,-T,/home/vidarr/myapps/marte_2.0_22Feb2017/ut
lis/linker.lds -static -nostartfiles -L/home/vidarr/myapps/marte_2.0_22Feb2017/
lib -L/home/vidarr/myapps/marte_2.0_22Feb2017/gnat_rts/rtlib/adalib -L/home/vidar
r/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnat -lgcc_sjlj
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017/examples$
```

Con esto, sabremos que el programa se compilo exitosamente. Para simularlo, usaremos la herramienta qemu. Para asegurarnos de que está instalado, ingresamos el comando `sudo apt-get install qemu`. Posteriormente, ingresamos el comando `sudo apt-get install qemu-system-i386`. Con esto estamos preparados para simular el código. Para hacerlo, ingresamos la siguiente línea `qemu-system-i386 -kernel tempo.exe`. Al hacerlo, se abrirá una nueva ventana y veremos una pantalla como la siguiente:



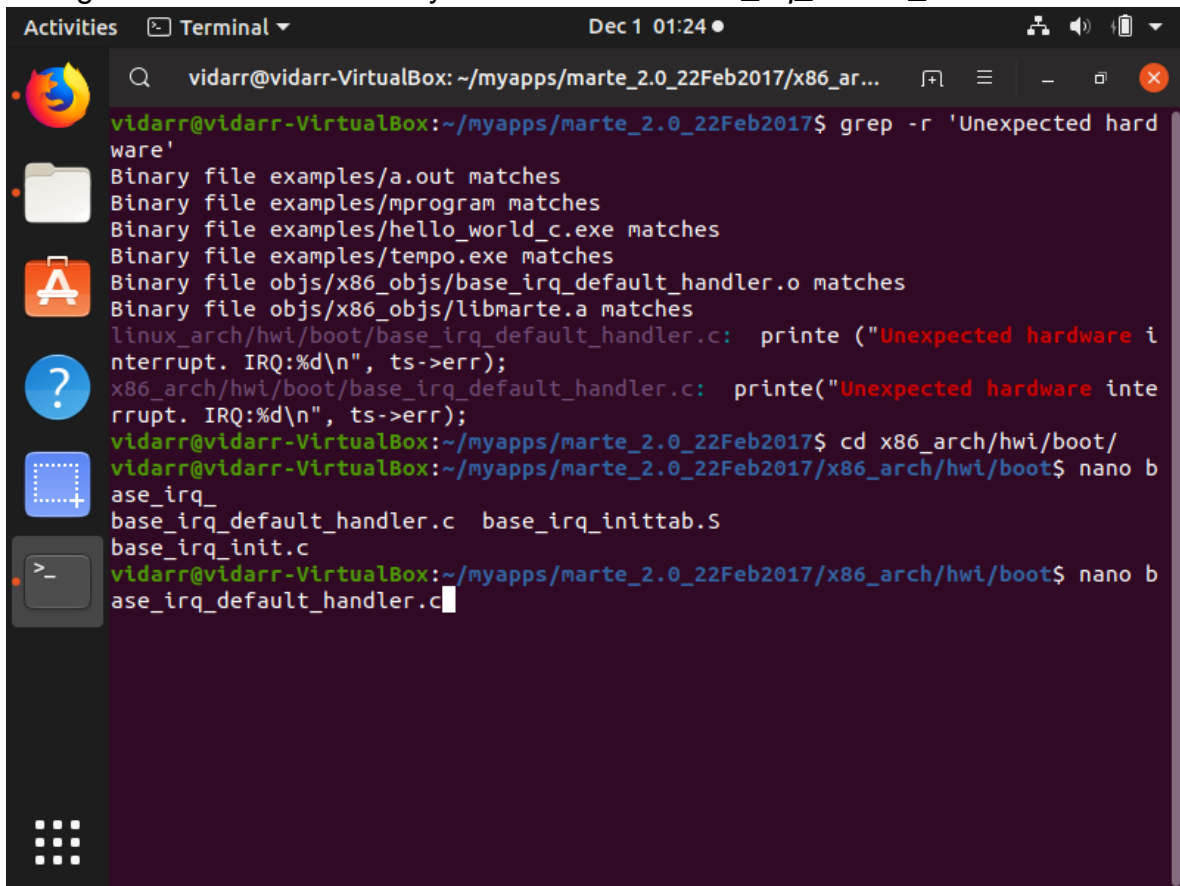


Los mensajes de error que se despliegan se deben a que el emulador qemu no contiene todas las variables y parámetros para simular el programa en Marte OS. Sin embargo, una vez que se ejecuta este programa estos mensajes no aparecen.

Si se desea correr este programa sobre hardware, el primer paso será modificar algunos archivos fuente de la instalación de marte OS, ya que de no hacerlo, aparecerá un mensaje de error al ejecutarlo. Para ello, usaremos el siguiente comando en la terminal: `grep -r 'Unexpected hardware'`. Se mostrará la siguiente información en pantalla, y como se muestra en la misma imagen, debemos

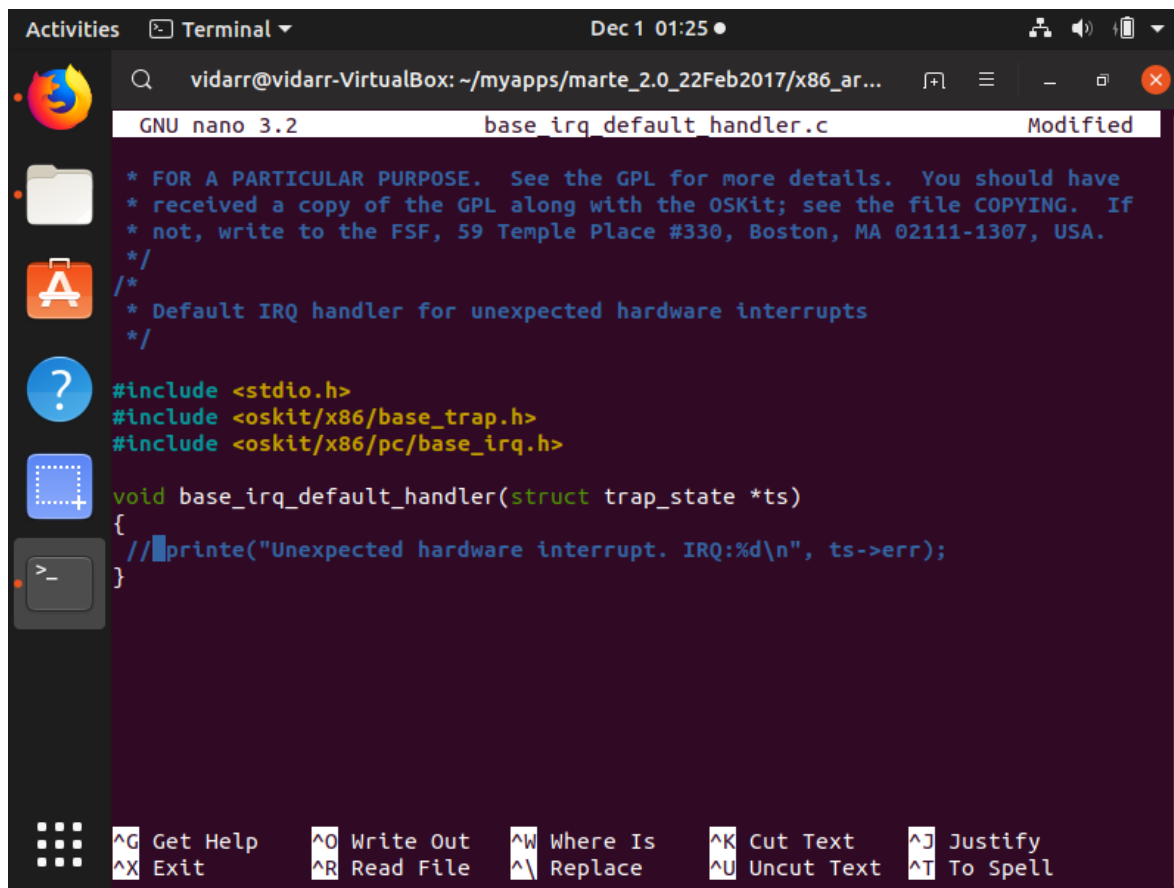


navegar hasta este directorio y abrir el archivo *base\_irq\_default\_handler.c*.



```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017/x86_ar...
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$ grep -r 'Unexpected hardware'
Binary file examples/a.out matches
Binary file examples/mprogram matches
Binary file examples/hello_world_c.exe matches
Binary file examples/tempo.exe matches
Binary file objs/x86_objcs/base_irq_default_handler.o matches
Binary file objs/x86_objcs/libmarte.a matches
linux_arch/hwi/boot/base_irq_default_handler.c:  printe ("Unexpected hardware interrupt. IRQ:%d\n", ts->err);
x86_arch/hwi/boot/base_irq_default_handler.c:  printe("Unexpected hardware interrupt. IRQ:%d\n", ts->err);
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017$ cd x86_arch/hwi/boot/
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/x86_arch/hwi/boot$ nano base_irq_default_handler.c base_irq_init.c
base_irq_init.c
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/x86_arch/hwi/boot$ nano base_irq_default_handler.c
```

Al abrirlo, notaremos que su única función es desplegar un mensaje. Comentamos estas líneas como se muestra a continuación.



```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017/x86_ar...
GNU nano 3.2      base_irq_default_handler.c      Modified

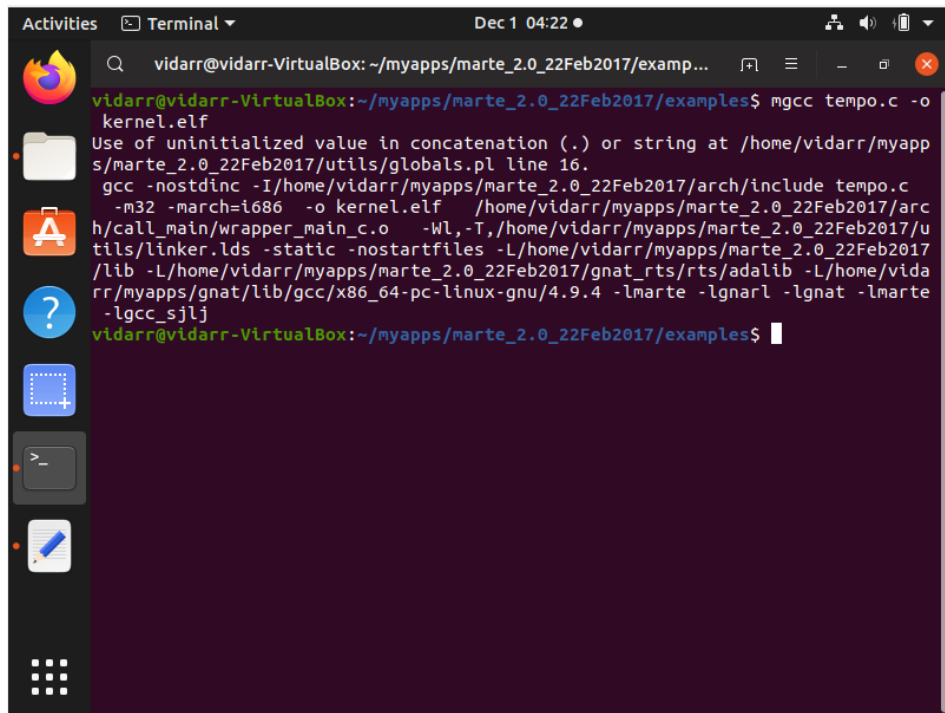
* FOR A PARTICULAR PURPOSE.  See the GPL for more details.  You should have
* received a copy of the GPL along with the OSKit; see the file COPYING.  If
* not, write to the FSF, 59 Temple Place #330, Boston, MA 02111-1307, USA.
*/
/*
* Default IRQ handler for unexpected hardware interrupts
*/

#include <stdio.h>
#include <oskit/x86/base_trap.h>
#include <oskit/x86/pc/base_irq.h>

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

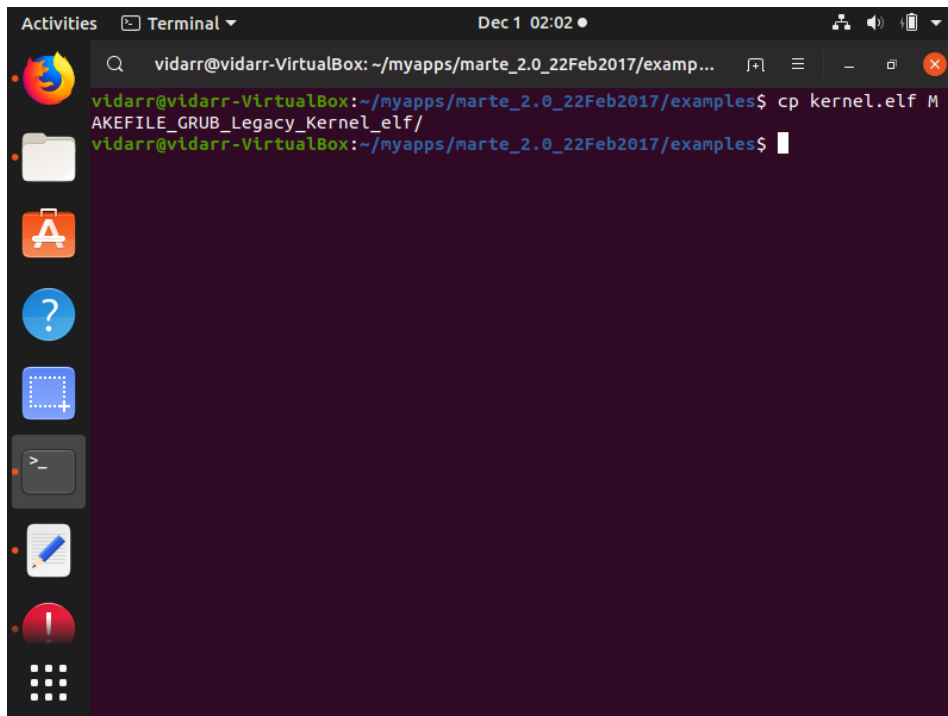
Guardamos los cambios y cerramos el archivo. Se debe volver a construir el sistema operativo y para ello, repetimos la instrucción *mkmart*. Una vez realizado este paso, es necesario descargar la carpeta *MAKEFILE\_GRUB\_Legacy\_Kernel\_Elf*, disponible en [https://github.com/GRUPO3MV11SOTR-2020-1/3MV11SOTR/tree/master/PRACTICA 1 Instalacion de un SOTR](https://github.com/GRUPO3MV11SOTR-2020-1/3MV11SOTR/tree/master/PRACTICA%201%20Instalacion%20de%20un%20SOTR). Una vez descargada se copia a la carpeta *examples*, la misma carpeta donde almacenamos el código fuente de *tempo.c*

Una vez realizado este proceso, procedemos a crear un archivo que pueda ser procesado para generar una imagen iso. Para ello, ejecutamos la siguiente línea: *mgcc tempo.c -o kernel.elf*.



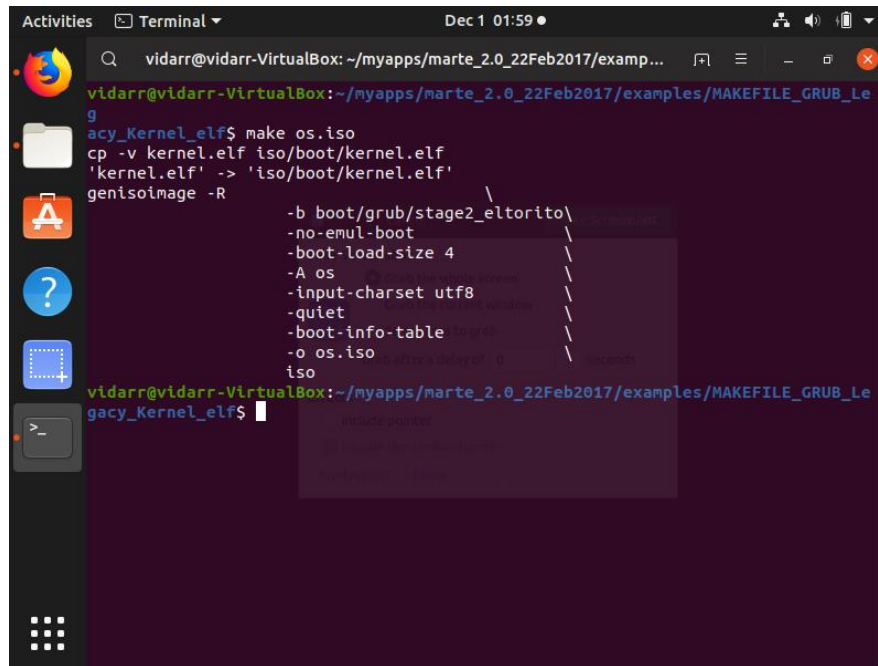
```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017/examp...  
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples$ mgcc tempo.c -o  
kernel.elf  
Use of uninitialized value in concatenation (.) or string at /home/vidarr/myapp  
s/marte_2.0_22Feb2017/utils/globals.pl line 16.  
gcc -nostdinc -I/home/vidarr/myapps/marte_2.0_22Feb2017/arch/include tempo.c  
-m32 -march=i686 -o kernel.elf /home/vidarr/myapps/marte_2.0_22Feb2017/arc  
h/call_main/wrapper_main.c.o -Wl,-T,/home/vidarr/myapps/marte_2.0_22Feb2017/u  
tils/linker.lds -static -nostartfiles -L/home/vidarr/myapps/marte_2.0_22Feb2017  
/lib -L/home/vidarr/myapps/marte_2.0_22Feb2017/gnat_rts/rts/adalib -L/home/vida  
rr/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnatl -lgnat -lmarte  
-lgcc_sjlj  
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples$
```

Después, copiamos este archivo al directorio *MAKEFILE\_GRUB*.. que  
descargamos mediante la siguiente instrucción:



```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017/examp...  
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples$ cp kernel.elf M  
AKEFILE_GRUB_Legacy_Kernel_elf/  
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples$
```

Finalmente, nos movemos a la carpeta *MAKEFILE\_GRUB\_Legacy\_Kernel\_Elf* y una vez ahí, ejecutamos el comando `make os.iso`. Obtendremos la siguiente pantalla.

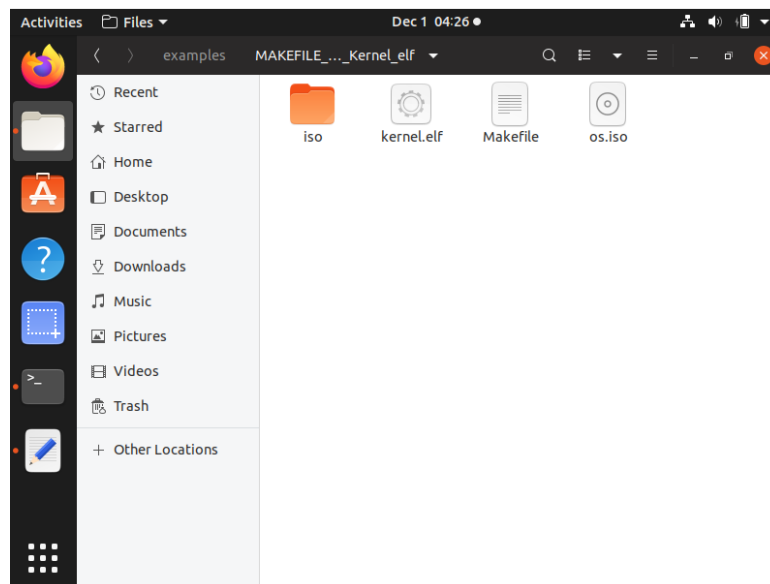


```
vidarr@vidarr-VirtualBox: ~/myapps/marte_2.0_22Feb2017/examp...
vidarr@vidarr-VirtualBox:~/myapps/marte_2.0_22Feb2017/examples/MAKEFILE_GRUB_Legacy_Kernel_Elf$ make os.iso
cp -v kernel.elf iso/boot/kernel.elf
'kernel.elf' -> 'iso/boot/kernel.elf'
genisoimage -R
-b boot/grub/stage2_eltorito\
-no-emul-boot
-boot-load-size 4
-A os
-input-charset utf8
-quiet
-boot-info-table
-o os.iso
iso
```

iso

File	Size	Permissions
iso	1.2 MB	drwxr-xr-x
kernel.elf	1.2 MB	-rwxr-xr-x
Makefile	1.2 MB	-rwxr-xr-x
os.iso	1.2 MB	-rwxr-xr-x

Con este se ha creado el archivo iso necesario. Si consultamos la carpeta, notaremos que se ha creado. Debemos copiarlo y tenerlo a disposición para crear la live-USB.



Con este archivo, podremos crear la Live-USB. Podemos usar la herramienta LiLi-USB creator, disponible en <http://www.linuxliveusb.com/en/download> , como se muestra en la siguiente imagen. Una vez creada la USB, sólo es cuestión de realizar el boot de la PC mediante la USB.

