

INSTITUTO POLITÉCNICO NACIONAL

Unidad Profesional Interdisciplinaria en
Ingeniería y Tecnologías Avanzadas



UNIDAD DE APRENDIZAJE:

SISTEMAS OPERATIVOS EN TIEMPO REAL

TRABAJO:

REPORTE DE LA CREACION DE UNA ALARMA
USANDO EL SISTEMA OPERATIVO EN TIEMPO REAL
“MaRTE OS”

ALUMNO:

- **Ruiz Hernandez David Israel**

GRUPO:

3MV11

PROFESOR:

Ing. LAMBERTO MAZA CASAS

FECHA DE ENTREGA: Jueves 28 de Noviembre del 2019



Contenido

A. Introducción.....	3
B. Desarrollo.....	4
C. Referencias Electrónicas.....	13

A. 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.

Características

Entre las características principales del kernel Marte OS se tiene:

- Soporta aplicaciones de lenguaje mixto en Ada, C y C++ (soporte experimental para Java también).
- Ofrece los servicios definidos en POSIX.13: pthreads, exclusiones mutuas, condvars.
- Todos los servicios tienen una respuesta de tiempo limitado (incluida la asignación de memoria dinámica con TLSF).
- Espacio de direcciones de memoria individual compartida por la aplicación multi-hilo y Marte OS.
- Disponible bajo la Licencia Pública General GNU 2.
- Sobre la base de la cadena de herramientas GNU AdaCore.
- Implementa el anexo Ada2005 Tiempo Real

Contribuciones por usuarios

Algunas contribuciones por usuarios de Marte OS

- Protocolos de comunicación y middleware
- PolyORB (DSA y CORBA) y GLADE para MaRTE OS x86 para equipos sin Sistema Operativo con RT-EP
- RT-WMP Real-Time Wireless protocol (UNIZAR)
- Protocolo RT-EP con una capa de ancho de banda reservado, algoritmos de exclusión mutua (Mutex) distribuidos, servicios de broadcast
- FRESCAN protocolo de red para bus CAN con ancho de banda reservado y servidores esporádicos
- DTM, el Gestor de Transacciones Distribuidas de FRESCOR
- MyCCM, OMG CCM (THALES)
- Drivers
- CAN bus
- Wireless ralink rt61
- SVGA, BTTV, Soundblaster 16
- Mouse, Keyboard, Joystick
- Ethernet drivers (intel eepro100, rtl8139, SiS900)
- Serial port driver
- I2C protocol, compass CMPS03 driver
- IDE disk driver (CompactFlash and HD) and FAT 16 filesystem

B. Desarrollo

Teniendo instalado el sistema operativo MaRTE OS dentro de la aplicación de DEBIAN para WINDOWS nos dirigimos a la siguiente dirección usando el comando cd de la siguiente forma: “cd /myapps/marte_2.0_22Feb2017/examples”, ahora tecleamos el comando vim para crear un nuevo archivo con el nombre que nosotros deseemos para colocar el código: “vim segundos_con_scanf.c”.

```
#include <stdio.h>
#include <time.h>
#include <math.h>
int main(){
    int t_en_seg=0,        //Tiempo en Segundos
        t_final=0,        //Tiempo al que se desea llegar
        s_a_contar=0,      //Variable que guarda los segundos que ingresa el usuario
        contador=0;        //Variable que imprime cuantos segundos han pasado

    time_t seconds;        //returns the time since the Epoch (January 1, 1970)
    seconds = time(NULL);   //Actualiza los segundos

    //-----
    printf("\nIndique el numero de segundos para configurar la alarma: ");
    scanf("%d", &s_a_contar);
    printf("\nSe ha configurado la alarma dentro de %d segundos\n", s_a_contar);
    //-----

    seconds = time(NULL);   //Actualiza los segundos
    t_final = seconds+s_a_contar; //Tiempo al que se desea llegar

    //-----
    while(seconds < t_final){ //Compara los segundos transcurridos con el tiempo a llegar
        t_en_seg = seconds+1;
        while(seconds < t_en_seg){ //Compara la variable seconds para saber si paso un segundo
            seconds = time(NULL); //Actualiza los segundos
        }
        seconds = time(NULL); //Actualiza los segundos
        contador++;           //incrementa la variable porque ya paso un segundo.
        printf("\nSegundos transcurridos: %i\n", contador);
        printf("%ld\n", seconds);
    }
    //-----
    printf("\n\t\t\t\t TIEMPO AGOTADO!!!");
    printf("\n\t\t\t\t TIEMPO AGOTADO!!!");
    printf("\n\t\t\t\t TIEMPO AGOTADO!!!\n\n");
    return (0);
}
```

Ilustración 1 Código de la Alarma

Al terminar de escribir el código para la Alarma presionamos la tecla “Esc” para salir del modo “Insert”, para después teclear “:x” para salir del Editor “vim” y guardar el archivo. Ahora usaremos el siguiente comando “mgcc segundos_con_scanf.c” para compilar el archivo que acabamos de crear, para verificar que la compilación fue exitosa se debe crear un archivo con el nombre “a.out”, para verificar la existencia del archivo usamos el comando “ls”.

```
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$ ls
80x86      a.out      flynshoot  hola       oscilloscope  segundos_sin_scanf.c
a          appsched   games      include     posix          segundos_sin_scanf.exe
ada        clock_modulation  hardware_interrupts leer_texto.c  qt            speaker
ada_P1.adb drivers     hello_world.adb  leer_texto.exe  README       Threads.c
alarma_2.c ejemplo_difftime.c hello_world_c.c  logger         segundos.c   Threads.exe
alarma.c   ejemplo_difftime.exe hello_world_cc.cc Makefile        segundos_con_scanf.c time_measurement
alarma.exe flynshoot   hello_world_c.exe mprogram       segundos.exe  widgets
```

Ilustración 2 Verificación de la creación del archivo "a.out". usando el comando "ls".

Ahora ejecutamos la instrucción `"mgcc segundos_con_scanf.c -o mprogram"` para crear un archivo de nombre "mprogram" que esta asociado a la compilación del archivo `"segundos_con_scanf.c"`, para verificar que el archivo se creó correctamente usamos el comando `"ls"`.

```
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$ mgcc segundos_con_scanf.c -o mprogram
Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string at /home/usuario/myapps/marte_2.0_22Feb2017/utis/global.pl line 16.
gcc -nostdinc -I/home/usuario/myapps/marte_2.0_22Feb2017/arch/include segundos_con_scanf.c -m32 -march=i686 -o mprogram /home/usuario/myapps/marte_2.0_22Feb2017/arch/call_main/wrapper_main.c.o -Wl,-T,/home/usuario/myapps/marte_2.0_22Feb2017/utis/linker.lds -static -nostartfiles -L/home/usuario/myapps/marte_2.0_22Feb2017/lib -L/home/usuario/myapps/marte_2.0_22Feb2017/gnat_rts/adalib -L/home/usuario/myapps/gnat/lib/gcc/x86_64-pc-linux-gnu/4.9.4 -lmarte -lgnat -lgnat -lmarte -lgcc_sjlj
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$ ls
80x86      a.out      flynshoot  hola       oscilloscope  segundos_sin_scanf.c
a          appsched   games      include     posix          segundos_sin_scanf.exe
ada        clock_modulation  hardware_interrupts leer_texto.c  qt            speaker
ada_P1.adb drivers     hello_world.adb  leer_texto.exe  README       Threads.c
alarma_2.c ejemplo_difftime.c hello_world_c.c  logger         segundos.c   Threads.exe
alarma.c   ejemplo_difftime.exe hello_world_cc.cc Makefile        segundos_con_scanf.c time_measurement
alarma.exe flynshoot   hello_world_c.exe mprogram       segundos.exe  widgets
```

Ilustración 3 Creación del archivo "mprogram" usando la instrucción `"mgcc segundos_con_scanf -o mprogram"`.

El siguiente paso es ejecutar la instrucción `"make segundos_con_scanf.exe"`, con esto crearemos un archivo ejecutable asociado con el archivo C, con el comando `"ls"` corroboramos que se halla creado el archivo.

```
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$ make segundos_con_scanf.exe
>> Compiling segundos_con_scanf.exe: Use of uninitialized value $GNAT_LIBS_PATH{"rpi"} in concatenation (.) or string at /home/usuario/myapps/marte_2.0_22Feb2017/utis/global.pl line 16.
segundos_con_scanf.c: In function 'main':
segundos_con_scanf.c:31:17: warning: format '%ld' expects argument of type 'long int', but argument 2 has type 'time_t' [-Wformat=]
    printf("%ld\n", seconds);
    ^
[OK]
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$ ls
80x86      appsched   hardware_interrupts  leer_texto.exe  segundos.c  Threads.exe
a          clock_modulation  hello_world.adb      logger          segundos_con_scanf.c  time_measurement
ada        drivers        hello_world_c.c      Makefile        segundos_con_scanf.exe
ada_P1.adb ejemplo_difftime.c hello_world_cc.cc    mprogram       segundos.exe
alarma_2.c ejemplo_difftime.exe hello_world_c.exe    oscilloscope   segundos_sin_scanf.c
alarma.c   flynshoot      hola                 posix          segundos_sin_scanf.exe
alarma.exe flynshoot      include              qt            speaker
a.out      games          leer_texto.c         README         Threads.c
```

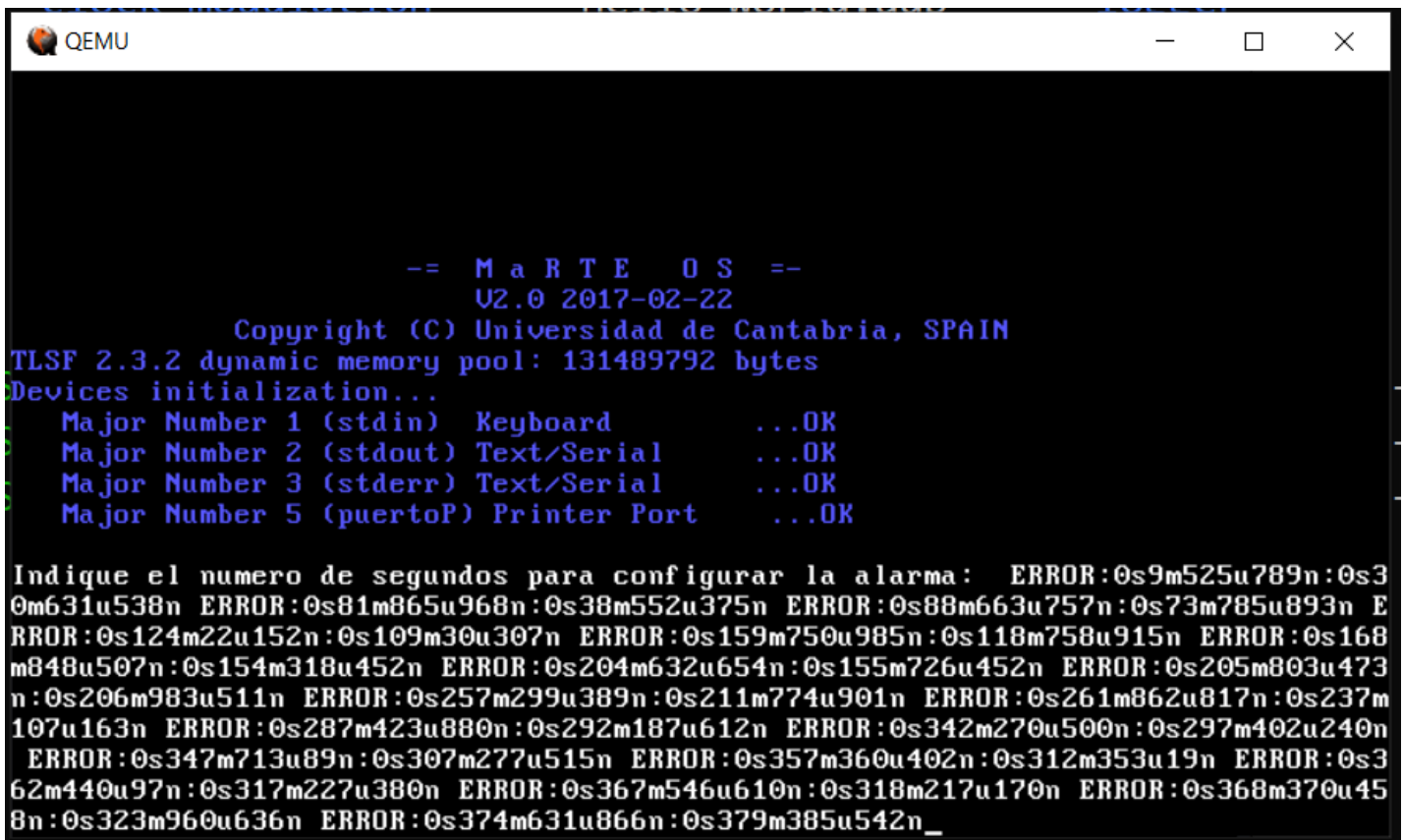
Ilustración 4 Creación del archivo "segundos_con_scanf.exe"

Para simular el programa que se acaba de crear se debe instalar la aplicación “Xming” de la siguiente dirección:
<https://sourceforge.net/projects/xming/>



Ilustración 5 Icono de la aplicación "Xming".

Después de que termina de instalar la aplicación debemos ejecutarla, esta lo hará en segundo plano por lo que solamente veremos el icono en la “Barra de iconos”. Ahora dentro de la aplicación “Debian” tecleamos la siguiente instrucción “qemu-system-i386 -kernel segundos_con_scanf.exe”.



```
-- 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: 131489792 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 (puertoP) Printer Port ...OK

Indique el numero de segundos para configurar la alarma: ERROR:0s9m525u789n:0s3
0m631u538n ERROR:0s81m865u968n:0s38m552u375n ERROR:0s88m663u757n:0s73m785u893n E
RROR:0s124m22u152n:0s109m30u307n ERROR:0s159m750u985n:0s118m758u915n ERROR:0s168
m848u507n:0s154m318u452n ERROR:0s204m632u654n:0s155m726u452n ERROR:0s205m803u473
n:0s206m983u511n ERROR:0s257m299u389n:0s211m774u901n ERROR:0s261m862u817n:0s237m
107u163n ERROR:0s287m423u880n:0s292m187u612n ERROR:0s342m270u500n:0s297m402u240n
ERROR:0s347m713u89n:0s307m277u515n ERROR:0s357m360u402n:0s312m353u19n ERROR:0s3
62m440u97n:0s317m227u380n ERROR:0s367m546u610n:0s318m217u170n ERROR:0s368m370u45
8n:0s323m960u636n ERROR:0s374m631u866n:0s379m385u542n_
```

Ilustración 6 Ejecución del programa "Segundos_con_scanf.exe" usando la aplicación "Xming".

Como se puede observar en la “Ilustración 6” se presentan algunos errores que no dejan que el programa se ejecute correctamente, estos suelen desaparecer cuando se crea la Live USB.

Lo siguiente es crear la Live USB usando la aplicación “Lili USB Creator” el cual se debe descargar del siguiente link:
<http://www.linuxliveusb.com/en/download>



Ilustración 7 Icono de la aplicación Linux Live USB Creator.

Además, debemos descargar y descomprimir en el escritorio el siguiente link, donde tenemos el “kernel.elf” y el “Makefile” para poder crear la USB.

https://github.com/sotrteacher/sotr_201808_201812/tree/master/PRACTICA_1_Instalacion_de_un_SOTR

Al término de los pasos anteriores, nos dirigimos a Debian y copiamos el archivo “segundos_con_scanf.exe” al siguiente directorio “C:\Users\Isra_el\Desktop\sotr_201808_201812-master\PRACTICA_1_Instalacion_de_un_SOTR\MAKEFILE_GRUB_Legacy_Kernel_elf” usando el comando “cp -v”.

```
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$ cp -v segundos_con_scanf.exe /mnt/c/Users/Isra_el/Desktop/sotr_201808_201812-master/PRACTICA_1_Instalacion_de_un_SOTR/MAKEFILE_GRUB_Legacy_Kernel_elf/segundos_con_scanf.exe
'segundos_con_scanf.exe' -> '/mnt/c/Users/Isra_el/Desktop/sotr_201808_201812-master/PRACTICA_1_Instalacion_de_un_SOTR/MAKEFILE_GRUB_Legacy_Kernel_elf/segundos_con_scanf.exe'
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$
```

Ilustración 8 Uso del comando "cp -r" para copiar archivos a un directorio.

Ahora necesitamos acceder a ese directorio usando el comando “cd”, para después el archivo .exe al “kernel.elf” usando el comando “cp -v”.

```
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017/examples$ cd /mnt/c/Users/Isra_el/Desktop/sotr_201808_201812-master/PRACTICA_1_Instalacion_de_un_SOTR/MAKEFILE_GRUB_Legacy_Kernel_elf/
usuario@DESKTOP-QPJ6FR4:/mnt/c/Users/Isra_el/Desktop/sotr_201808_201812-master/PRACTICA_1_Instalacion_de_un_SOTR/MAKEFILE_GRUB_Legacy_Kernel_elf$ cp -v segundos_con_scanf.exe kernel.elf
'segundos_con_scanf.exe' -> 'kernel.elf'
usuario@DESKTOP-QPJ6FR4:/mnt/c/Users/Isra_el/Desktop/sotr_201808_201812-master/PRACTICA_1_Instalacion_de_un_SOTR/MAKEFILE_GRUB_Legacy_Kernel_elf$
```

Ilustración 9 Creación del archivo "kernel.elf" usando el archivo "segundos_con_scanf".

Por último, debemos crear el archivo “ISO” usando el comando “make”.

```
usuario@DESKTOP-QPJ6FR4:/mnt/c/Users/Isra_el/Desktop/sotr_201808_201812-master/PRACTICA_1_Instalacion_de_un_SOTR/MAKE
FILE_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
```

Ilustración 10 Creación del archivo "ISO" usando el comando "make".

Para crear la Live USB debemos conectar una usb a la computadora y ejecutar el programa “Linux Live USB Creator”, donde debemos elegir la unidad que acabamos de conectar y el archivo ISO que acabamos de crear, por recomendación se debe seleccionar la pestaña de “Formatear Dispositivo”, por ultimo debemos hacer clic en el rayo para comenzar la instalación.



Ilustración 11 Programa para crear la Live USB.

El siguiente paso es reiniciar la computadora y presionar la tecla “F12” para entrar a la configuración de arranque y poner en primer lugar el puerto usb. De esta forma podemos ver como se empieza a ejecutar nuestro programa sobre el sistema operativo tiempo real MaRTE OS.

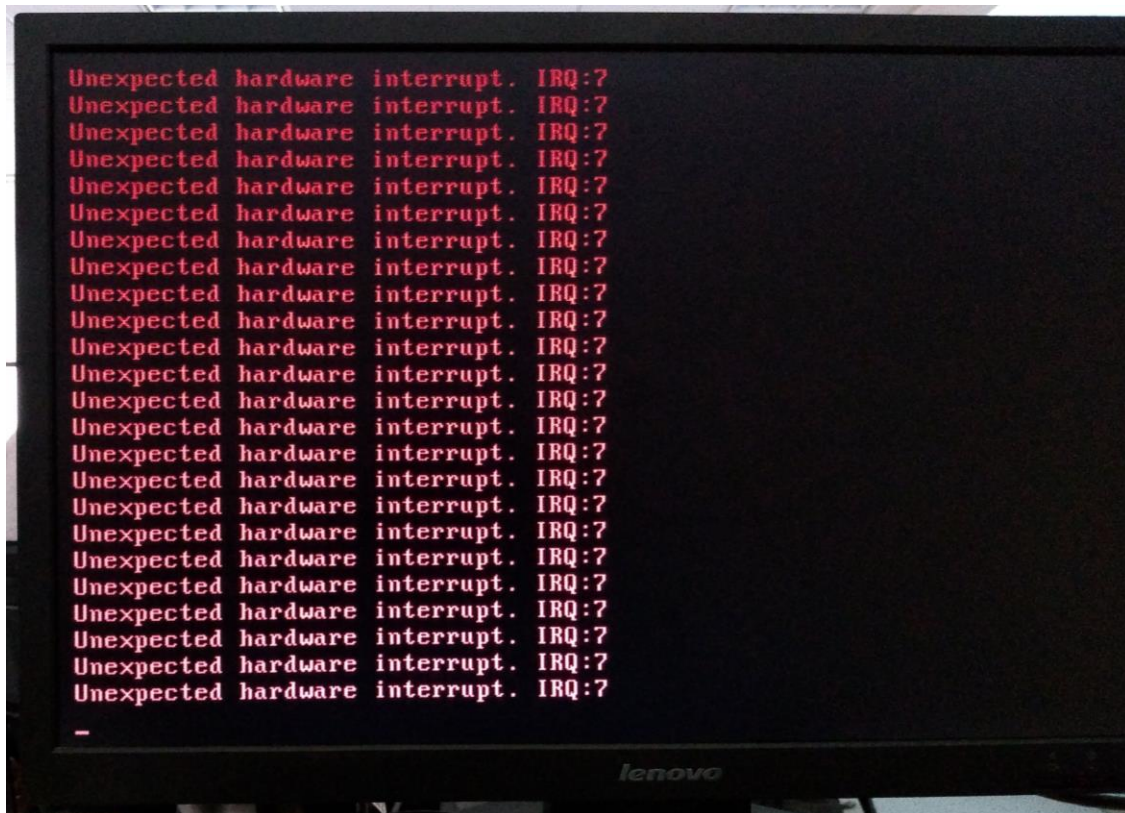


Ilustración 12 Se muestran los errores que aparecen al intentar ejecutar el programa “segundos_con_scanf”.

Como se puede observar en el “Ilustración 12” al ejecutar el programa aparece el error: “Unexpected hardware interrup. IRQ:7”. Para evitar que este error apareciera en la pantalla se decidió comentar esas líneas. Usando la instrucción “grep -r ‘Unexpected hardware’ /home/usuario/myapps/marte_2.0_22Feb2017”.

```
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017$ grep -r 'Unexpected hardware' /home/usuario/myapps/marte_2.0_22Feb2017
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/Threads.exe matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/a.out matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/alarma.exe matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/ejemplo_difftime.exe matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/hello_world_c.exe matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/leer_texto.exe matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/mprogram matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/segundos.exe matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/examples/segundos_sin_scanf.exe matches
/home/usuario/myapps/marte_2.0_22Feb2017/linux_arch/hwi/boot/base_irq_default_handler.c: printe ("Unexpected hardware interrupt. IRQ:%d\n", ts->err);
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/objs/x86_objs/base_irq_default_handler.o matches
Binary file /home/usuario/myapps/marte_2.0_22Feb2017/objs/x86_objs/libmarte.a matches
/home/usuario/myapps/marte_2.0_22Feb2017/x86_arch/hwi/boot/base_irq_default_handler.c: printe("Unexpected hardware interrupt. IRQ:%d\n", ts->err);
```

Ilustración 13 Uso del comando “grep -r” para buscar palabras dentro de documentos.

De acuerdo a la “Ilustración 13” tenemos 2 archivos que debemos modificar y comentar las líneas donde aparece “Unexpected hardware interrupt. IRQ:7”. El primero esta dentro de la carpeta “linux_arch” y el segundo “x86_arch” y para ambos el nombre del archivo es “base_irq_default_handler.c”.

```
/*  
 * Default IRQ handler for unexpected hardware interrupts  
 */  
  
/*#include <stdio.h>  
#include <oskit/x86/base_trap.h>  
#include <oskit/x86/pc/base_irq.h>  
*/  
  
#include "struct_trap.h"  
#include <stdio.h>  
#include "marte_functions.h"  
  
void base_irq_default_handler(struct trap_state *ts)  
{  
    //printf ("Unexpected hardware interrupt. IRQ:%d\n", ts->err);  
}
```

Ilustración 14 Modificación del archivo "base_irq_default_handler.c".

Ahora debemos usar el comando “mkmarte” para volver a compilar el sistema operativo en tiempo real MaRTE OS y que los cambios que acabamos de hacer se apliquen, si el proceso fue exitoso debe aparecer un mensaje como el de la “Ilustración 15”.

```
make[1]: Leaving directory '/home/usuario/myapps/marte_2.0_22Feb2017/lang_support/cxx/libsupc++'  
make -C ustl-src/ all  
make[1]: Entering directory '/home/usuario/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'  
make[1]: Nothing to be done for 'all'.  
make[1]: Leaving directory '/home/usuario/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'  
make -C ustl-src/ install  
make[1]: Entering directory '/home/usuario/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/usuario/myapps/marte_2.0_22Feb2017/lang_support/ustl-src'  
C++ language support library DONE  
  
mkmarte: work done :-)  
usuario@DESKTOP-QPJ6FR4:~/myapps/marte_2.0_22Feb2017$
```

Ilustración 15 Mensaje "work done" que aparece al termino de usar el comando "mkmarte".

Al término de la compilación de MaRTE, debemos de volver a realizar los pasos desde que usamos la instrucción “mgcc segundos_con_scanf.c” para crear el archivo .exe con los cambios que se realizaron al sistema operativo en tiempo real. También debemos de crear la LIVE USB con el nuevo archivo “segundos_con_scanf.exe”. Al reiniciar nuestra computadora y tener conectada la memoria con el nuevo archivo .iso nuestro programa de la ALARMA se debe ejecutar correctamente.

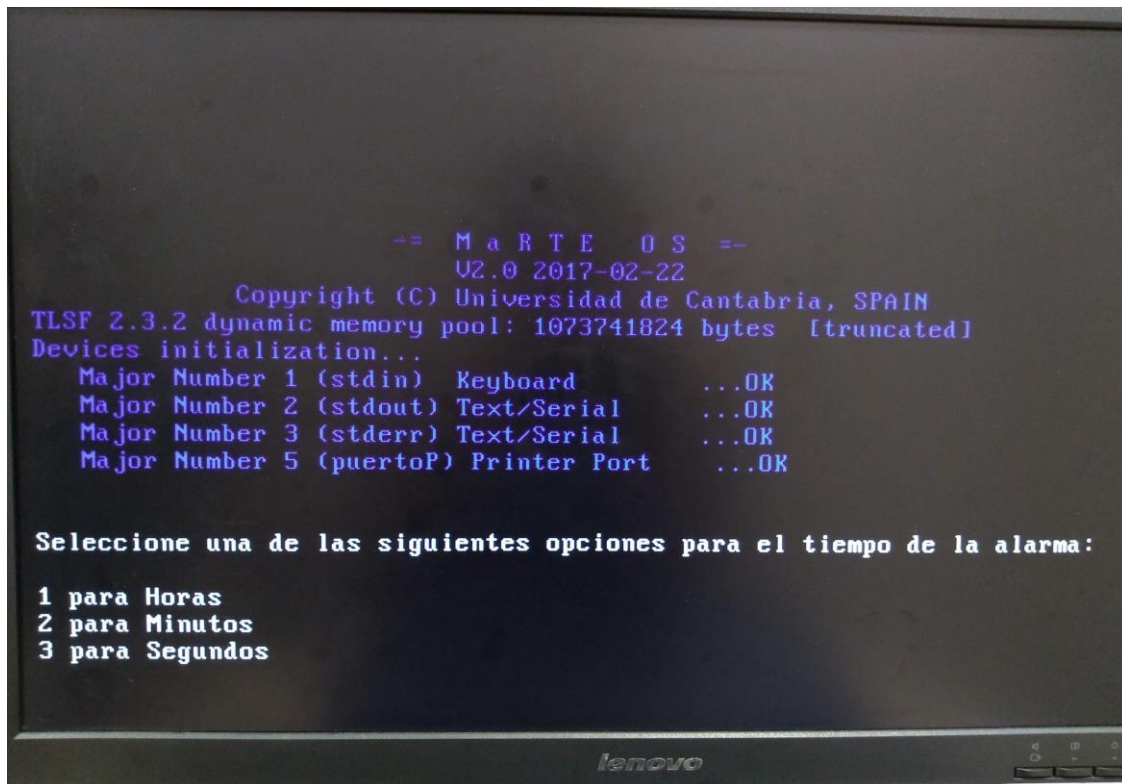


Ilustración 16 Ejecución del programa "segundos_con_scanf.c". usando MaRTE OS.

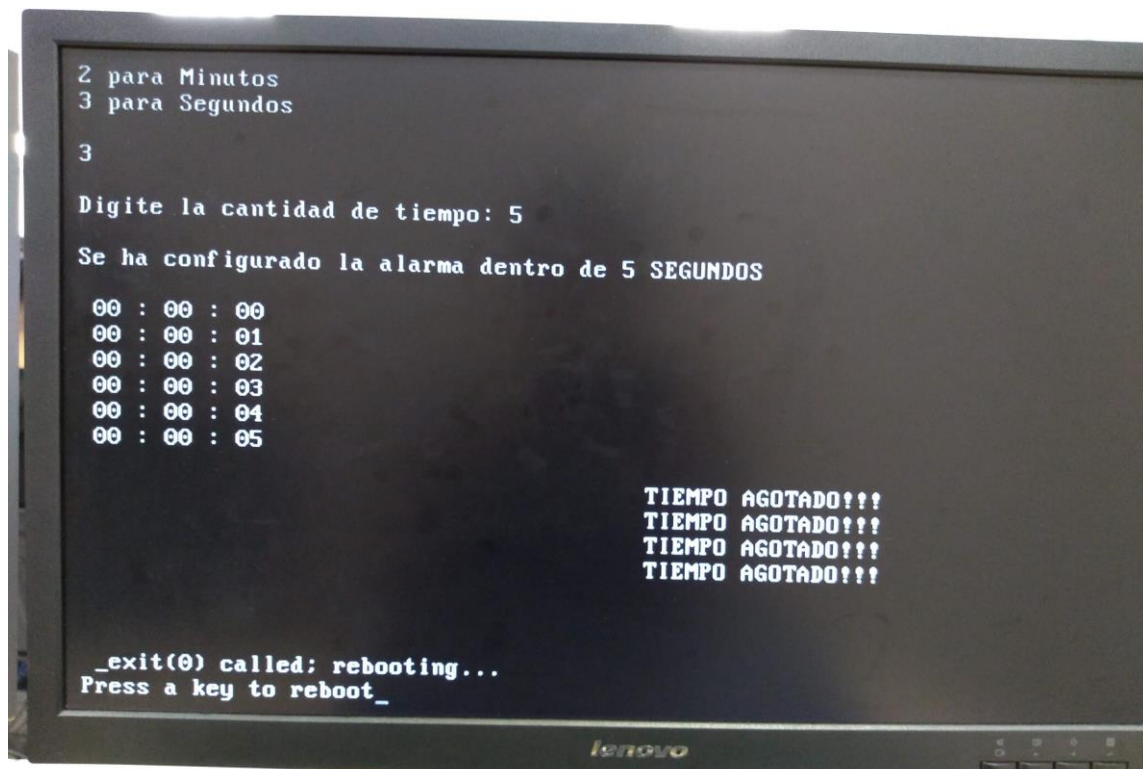


Ilustración 17 Configuración de la alarma usando MaRTE OS.

Observando la *“Ilustracion 16”* nos damos cuenta que se pudo ejecutar correctamente el programa *“segundos_con_scanf.c”* sin que se mostrara ningún error en la pantalla. En la *“Ilustración 17”* se configura la alarma dentro de 5 SEGUNDOS, pasando el tiempo se muestra en pantalla un mensaje de *“TIEMPO AGOTADO”* lo que nos demuestra que nuestra alarma funciona correctamente.



C. Referencias Electrónicas.

https://es.wikipedia.org/wiki/MaRTE_OS

<https://martel.unican.es/>

https://www.ctr.unican.es/asignaturas/MC_ProCon/Doc/ProCon_II_02-marte_3en1.pdf

<https://prezi.com/uklrnt37zeg6/marte-os/>

