Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

SISTEMAS OPERATIVOS EN TIEMPO REAL

REPORTE CREACION PROGRAMA ALARMA

Grupo: 3MV11

Alumno: Ocampo Delgado Osvaldo Josué

Profesor: Maza Casas Lamberto

03 de noviembre del 2019

En esta guía explicare los pasos a seguir en la instalación del sistema operativo Marte y también la compilación de un programa “alarma” que simule ser una alarma mandando un aviso en la ejecución del programa a la misma hora real que la que ingrese el usuario, será compilado en Marte OS

INSTALACION MARTE OS

Para empezar con la instalación de Marte OS tenemos que contar con los siguientes archivos: gnat-gpl-2016-x86_64-linux-bin.tar.gz y 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.

Para empezar a Instalar Marte OS

Una vez descargados los archivos los copiamos dentro de debían con el siguiente comando, indicando la ruta de ubicación de los archivos:

cp –rv /mnt/Users/Sala7/Downloads/ gnat-gpl-2016-x86_64-linux-bin.tar.gz gnat-gpl-2016-x86_64-linux-bin.tar.gz

cp –rv /mnt/Users/Sala7/Downloads/ marte_2.0_22Feb2017_src.tar.gz

Después de esto descomprimimos ambos archivos con el siguiente comando:

tar xvf gnat-gpl-2016-x86_64linux-bin.tar.gz

tar xvf marte_2.0_22Feb2017_src.tar.gz

A continuación utilizamos el comando "*cd*" dentro de debían y en seguida el comando "*nano .bashrc*". Con este comando ingresamos al archivo bashrc, iremos al final del archivo y editaremos el archivo para que queden las siguientes líneas de código:

```
# 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

#
alias ls='ls --color=no'
export PATH=$PATH:/opt/tcc-0.9.27/bin
#aca acaba lo agregado

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

Después de modificar el archivo escribimos Ctrl+O para guardar los cambios y reiniciamos Debian.

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

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

Lo siguiente es salir a la carpeta 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/".

Una vez ahí procederemos a instalar el SOTR, para lo cual ejecutaremos la instrucción "./minstall". Para continuar presionamos la tecla "ENTER". Y a diferencia del compilador este proceso es bastante rápido, por lo que pronto visualizaremos lo siguiente, en señal de que el proceso ha terminado exitosamente.

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 está dentro de la carpeta "marte_2.0_22Feb2017", una vez estando dentro de la carpeta utils, procederemos a definir la arquitectura sobre la que trabajara nuestro SOTR, para ello ingresaremos la siguiente instrucción a la terminal "msetcurrentarch", y podremos ver que tenemos varias arquitecturas para elegir.

Elegimos la arquitectura x86, por lo que ingresaremos la siguiente instrucción en la terminal "msetcurrentarch x86 i386".

Después de esto ejecutaremos las instrucciones “mkrtsmarteuc” y posteriormente “mkmarte”, en ese orden. Hasta este punto, terminamos con la instalación del SOTR MaRTE OS, lo siguiente es probar que realmente funcione, para poder hacerlo, requeriremos 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.

Lo siguiente es ir a la carpeta de ejemplos de marte con el comando “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.

Ejecutaremos la siguiente instrucción “mgcc hello_world_c.c” con esto compilamos el archivo. 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”.

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

En caso que muestre error será necesario descargar el software Xming para poder visualizar la ventana QEMU, se puede descargar en el siguiente link:

https://sourceforge.net/projects/xming/

Con esto terminamos la instalación de MarteOS, a continuación, explicaremos la construcción del programa alarma.

Construcción programa Alarb2.c

```
Archivo  Edición  Formato  Ver  Ayuda
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

time_t now;
struct tm *right_now;
int main()
{
    int hour, minutes;
    printf("Ingrese las Horas:");
    scanf("%d",&hour);
    printf("\nIngrese los Minutos: ");
    scanf("%d",&minutes);
    while(1)
    {
            time(&now);
            right_now=localtime(&now);

            if(hour == right_now->tm_hour & minutes == right_now->tm_min)
            {
                    printf("\nALARMA ALARMA");
                    printf("\a");
                    break;
             }
    }
    return 0;
}
```

Después de tener el código d ela alarma en C debemos dirigirnos a estos dos archivos:

- marte_2.0_22Feb2017/x86_arch/hwi/boot/base_irq_default_handler.c
- marte_2.0_22Feb2017/x86_arch/Linux_arch/hwi/boot/base_irq_default_handler.c

Para comentar las siguientes lineas, esto para que no se muestre el error *"Unexpected hardware interrup":*

```
 * Copyright (c) 1997-1998 University of Utah and the Flux Group.
 * All rights reserved.
 *
 * This file is part of the Flux OSKit.  The OSKit is free software, also known
 * as "open source;" you can redistribute it and/or modify it under the terms
 * of the GNU General Public License (GPL), version 2, as published by the Free
 * Software Foundation (FSF).  To explore alternate licensing terms, contact
 * the University of Utah at csl-dist@cs.utah.edu or +1-801-585-3271.
 *
 * The OSKit is distributed in the hope that it will be useful, but WITHOUT ANY
 * WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
 * 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);
}

~
```

Una vez hecho esto, nos vamos a la carpeta de examples en marte, en la dirección: cd marte/examples, donde se encuentra el programa alarb2.c.

Se modificarán los archivos marte-kernel-devices_table.adb y marte-kernel-devices_table.ads. Ambos se encuentran en el directorio: marte_2.0_22Feb2017/kernel.

En el archivo. ads se des comentarán las siguientes líneas de código:

```
--  configuration_parameters.ads) and recompile MaRTE OS as
--  explained in the user's guide (marte_ug.html).

The_Driver_Table :  K.File_System_Data_Types.Driver_Table_Type :=
  (
   --  Standard Input
   --  DO NOT COMMENT OUT. If you don't want to use this device
   --  just set all entries to 'null'.
   --  If you modify this device, you should also change
   --  'Stdin_Direct_Read' in 'kernel_console.ads'.
   1 => (Name   => "Keyboard        ",
         Create => Keyboard_Functions.Create'Access,
         Remove => null,
         Open   => null,
         Close  => null,
         Read   => Keyboard_Functions.Read'Access,
         Write  => null,
         Ioctl  => Keyboard_Functions.Ioctl'Access,
         Delete => null,
         Lseek  => null),

   --  Standard Output
   --  DO NOT COMMENT OUT. If you don't want to use this device
   --  just set all entries to 'null'.
   --  If you modify this device, you should also change
   --  'Stdout_Basic_Init' and 'Stdout_Direct_Write' in
   --  'kernel_console.ads'.
   2 => (Name   => "Text/Serial     ",
         Create => Console_Switcher_Import.Create_Ac,
         Remove => Console_Switcher_Import.Remove_Ac,
         Open   => Console_Switcher_Import.Open_Ac,
         Close  => Console_Switcher_Import.Close_Ac,
         Read   => null,
         Write  => Console_Switcher_Import.Write_Ac,
         Ioctl  => Console_Switcher_Import.Ioctl_Ac,
         Delete => null,
         Lseek  => null),
   --  2 => (Name   => "Text Console    ",
   --        Create => Text_Console_Driver_Import.Create_Ac,
   --        Remove => Text_Console_Driver_Import.Remove_Ac,
   --        Open   => Text_Console_Driver_Import.Open_Ac,
```

```
   --  'Stderr_Basic_Init' and 'Stderr_Direct_Write' in
   --  'kernel_console.ads'.
   3 => (Name   => "Text/Serial     ",
         Create => Console_Switcher_Import.Create_Ac,
         Remove => Console_Switcher_Import.Remove_Ac,
         Open   => Console_Switcher_Import.Open_Ac,
         Close  => Console_Switcher_Import.Close_Ac,
         Read   => null,
         Write  => Console_Switcher_Import.Write_Error_Ac,
         Ioctl  => Console_Switcher_Import.Ioctl_Ac,
         Delete => null,
         Lseek  => null),
   --  3 => (Name   => "Text Console    ",
   --        Create => Text_Console_Driver_Import.Create_Ac,
   --        Remove => Text_Console_Driver_Import.Remove_Ac,
```

```
     --         Delete => null,
     --         Lseek  => null),

     --  Printer Port
5 => (Name   => "Printer Port    ",
           Create => Printer_Port_Functions.Create'Access,
           Remove => null,
           Open   => null,
           Close  => null,
           Read   => Printer_Port_Functions.Read'Access,
           Write  => Printer_Port_Functions.Write'Access,
           Ioctl  => Printer_Port_Functions.Ioctl'Access,
           Delete => null,
           Lseek  => null),

     --  Generic Ethernet Card (Sis900, Intel EtherExpresss 100 or RTL8139)
     --  6 => (Name   => "Ether Drv       ",
     --        Create => Ethernet_Create_Ac
```

```
     --         Lseek  => null),

    others =>
       ("               ",
       null, null, null, null, null, null, null, null, null));

  ------------------------------------------------------------------------
  --  Device files table  -------------------------------------------------
  ------------------------------------------------------------------------
  --
  --  This table defines the files in the file-system. To create a new
  --  file, add a new entry to the table using the appropriate
  --  parameters:
  --
  --  The major number must be the one that points to the driver in
  --  The_Driver_Table rows
  --
  --  The minor number may be anyone, as it's used as a minor number
  --  to differentiate assignments of the same Major_Number
  --
  --  The Boolean indicates whether the assignment is in use or not

  The_Device_Files_Table :
    K.File_System_Data_Types.Device_Files_Table_Type :=
    (
     --        File path  Major  Minor   Used   Type   Del  Count
     --  Standard files. DO NOT COMMENT OUT
     1  => ("/dev/stdin      ", 1, 0, True, Device, False, 0),
     2  => ("/dev/stdout     ", 2, 0, True, Device, False, 0),
     3  => ("/dev/stderr     ", 3, 0, True, Device, False, 0),
     --  User defined files
     --  4  => ("/dev/ttyS0      ", 4, 0, True, Device, False, 0),
     --  5  => ("/dev/ttyS1      ", 4, 1, True, Device, False, 0),
     --  6  => ("/dev/ttyS2      ", 4, 2, True, Device, False, 0),
     --  7  => ("/dev/ttyS3      ", 4, 3, True, Device, False, 0),
     8  => ("/dev/lpt0       ", 5, 0, True, Device, False, 0),
     --  9  => ("/dev/eth0       ", 6, 0, True, Device, False, 0),
     --  10 => ("/dev/buffer     ", 7, 0, True, Device, False, 0),
     --  11 => ("/dev/demo_ada   ", X, 0, True, Device, False, 0),
     --  12 => ("/dev/demo_c     ", X, 0, True, Device, False, 0),
     --  13 => ("/fat/           ", 9, 0, True, Mount_Point, False, 0),
     --  14 => ("/dev/dio        ", 10, 0, True, Device, False, 0),
     --  15 => ("/dev/daq        ", 11, 0, True, Device, False, 0),
     --  16 => ("/dev/cmps03     ", 12, 0, True, Device, False, 0),
     --  17 => ("/dev/membuff    ", 13, 0, True, Device, False, 0),
       18 => ("/dev/can0       ", 14, 0, True, Device, False, 0),
     --  19 => ("/dev/can1       ", 14, 1, True, Device, False, 0),
     --  20 => ("/dev/ps2mouse   ", 15, 0, True, Device, False, 0),
     others => ("               ", 9, 0, False, Device, False, 0)
     );
```

En el archivo. adb se agregarán y des comentarán las siguientes líneas:

```
--  Typical standard devices (standard input, output and error)
with Keyboard_Functions;          --  standard input
pragma Elaborate(Keyboard_Functions);
--  with Text_Console_Driver_Import;  --  standard output and error
with Console_Switcher_Import;  --  standard output and error
--  with Serial_Port_Driver_Import;  --  standard output and error

--  User's drivers "withs" (add line 'with {your_driver}')
--  with Dynamic_Buffer_Driver.Functions;
--  with Serial_Port_Driver_Import;
 with Printer_Port_Functions;
 pragma Elaborate(Printer_Port_Functions);
--  with Ethernet;
with Demo_Driver_Ada_Functions;
--with Demo_Driver_C_Import;:
--  with Fat_Driver_Functions;
--  pragma Elaborate (Fat_Driver_Functions);
--  with pcm3718_functions;
--  with cmps03_functions;
--  with Membuffer_Driver_Import;
--  with CAN_Driver_Import;
--  with PS2_Mouse.Functions;

--  MaRTE OS "withs" (Do not edit)
pragma Elaborate(Demo_Driver_Ada_Functions);
with Demo_Driver_C_Import;
with Membuffer_Driver_Import;

with MaRTE.Kernel.File_System_Data_Types;
use MaRTE.Kernel.File_System_Data_Types;


package MaRTE.Kernel.Devices_Table is
   package K renames MaRTE.Kernel;

   ------------------------------------------------------------------------
   --  Drivers Table  ------------------------------------------------------
```

```
47 --  executable file might be covered by the GNU Public License.
48 ------------------------------------------------------------------------------
49
50 with MaRTE.Configuration_Parameters;
51 with MaRTE.Direct_IO; use MaRTE.Direct_IO;
52
53 package body MaRTE.Kernel.Devices_Table is
54
55    --------------------------
56    --  Initialize_Devices  --
57    --------------------------
58    procedure Initialize_Devices is
59       use type Int;
60    begin
61       if MaRTE.Configuration_Parameters.Use_Devices_Filesystem then
62          Put ("Devices initialization..."); New_Line;
63          for I in Major loop
64             if The_Driver_Table (I).Create /= null then
65                Put ("   Major Number "); Put (Integer (I));
66                case I is
67                   when 1 =>      Put (" (stdin)  ");
68                   when 2 =>      Put (" (stdout) ");
69                   when 3 =>      Put (" (stderr) ");
70                   when 4 =>      Put ("(Ipt driver)");
71                                  when others => Put ("          ");
72                end case;
73                Put (The_Driver_Table (I).Name);
74                if The_Driver_Table (I).Create.all = 0 then
75                   Put ("...OK");
76                else
77                   Put ("...*** FAIL!! ***");
78                end if;
79                New_Line;
80             end if;
81          end loop;
82       end if;
83    end Initialize_Devices;
84
85 end MaRTE.Kernel.Devices_Table;
~
~
```

Creación .ISO

Por último, se escribirá el comando "*mgcc alarb2.c"* para compilar el archivo. Para revisar que la compilación haya sido exitosa, se busca un archivo de nombre "*a.out"*.

A continuación se escribirá la instrucción *"mgcc alarma2.c –o mprogram"* para crear un archivo de tipo mprogram asociado a alarb2.c.

Después se escribirá la instrucción "*make alarb2.exe*" para construir un ejecutable.

Este se debe mover a la carpeta so2 para poder crear el os.iso. Para ello, nos colocamos dentro de so2 y escribimos el siguiente comando:

"cp –v ../ marte_2.0_22Feb2017/examples/alarb2.exe kernel.elf".

Esto copiará el archivo alarb2.c a la carpeta so2 y lo renombrará como *kernel.elf*.

Por último, se procede a escribir la instrucción "*make"* dentro de so2, lo que creará el archivo os.iso.