Introduccion

Antes de comenzar revise los siguientes apendices y complete las instrucciones que se indican

Para una explicacion detallada sobre como instalar un ambiente EDK2 en una Raspberry Pi 3, consulte el Apendice A.

Si desea instalar la maquina virutal Qemu consulte el Apendice B

Para habilitar el ambiente EDK2 y compilar para x86_64 consulte el apendice C

Una vez que tenga instalado y funcional el ambiente para compilar en arquitectura x86_64, procedemos a mostrar como compilar el paquete de ejemplo *MdeModulePkg.dsc*, el cual contine una descripcion de los archivos requeridos para construir la aplicacion UEFI:

```
rodrigo@raspberrypi:~/src/edk2 $ cat MdeModulePkg/MdeModulePkg.dsc
## @file
# EFI/PI Reference Module Package for All Architectures
#
# (C) Copyright 2014 Hewlett-Packard Development Company, L.P.<BR>
# Copyright (c) 2007 - 2021, Intel Corporation. All rights reserved.<BR>
# Copyright (c) Microsoft Corporation.
# Copyright (C) 2024 Advanced Micro Devices, Inc. All rights reserved.<BR>
#
#    SPDX-License-Identifier: BSD-2-Clause-Patent
#
##

[Defines]
  PLATFORM_NAME                  = MdeModule
  PLATFORM_GUID                  = 587CE499-6CBE-43cd-94E2-186218569478
  PLATFORM_VERSION               = 0.98
  DSC_SPECIFICATION              = 0x00010005
  OUTPUT_DIRECTORY               = Build/MdeModule
  SUPPORTED_ARCHITECTURES        = IA32|X64|EBC|ARM|AARCH64|RISCV64|LOONGARCH64
  BUILD_TARGETS                  = DEBUG|RELEASE|NOOPT
  SKUID_IDENTIFIER               = DEFAULT

!include MdePkg/MdeLibs.dsc.inc

[LibraryClasses]
  #
  # Entry point
  #
  PeiCoreEntryPoint|MdePkg/Library/PeiCoreEntryPoint/PeiCoreEntryPoint.inf
  PeimEntryPoint|MdePkg/Library/PeimEntryPoint/PeimEntryPoint.inf
  DxeCoreEntryPoint|MdePkg/Library/DxeCoreEntryPoint/DxeCoreEntryPoint.inf
```

Para esto usaremos el script *build* que se genera por defecto con la instalación de la paqueteria EDK2, antes de comenzar a usar el script debemos asegurar que las variables de ambiente que usa el proyecto EDK2 esten configuradas correctamente, ejecutamos el script de configuración:

```
rodrigo@raspberrypi:~/src/edk2 $ . ./edksetup.sh
Loading previous configuration from /home/rodrigo/src/edk2/Conf/BuildEnv.sh
Using EDK2 in-source Basetools
WORKSPACE: /home/rodrigo/src/edk2
EDK_TOOLS_PATH: /home/rodrigo/src/edk2/BaseTools
CONF_PATH: /home/rodrigo/src/edk2/Conf
rodrigo@raspberrypi:~/src/edk2 $
```

El cual nos configura las variables a los directorios correctos, despues de eso utilizaremos el script *build* con los siguientes parametros:

```
rodrigo@raspberrypi:~/src/edk2 $ build -a X64 -t GCC5 -b DEBUG -p MdeModulePkg/MdeModulePkg.dsc
```

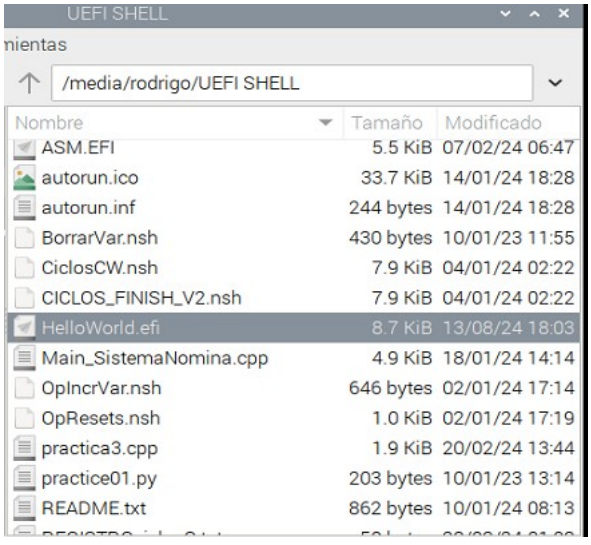
El parametro *-a X64* indica que tipo de arquitectura estamos "apuntando" compilar nuestra aplicacion (es decir al microprocesador que ejecutara la aplicacion), la opcion -t GCC5 se refiere a usar el conjunto de herramientas de compilacion GCC5 (el anexo C describe cual compilador es el usado), y por ultimo la orden *-p MdeModulePkg/MdeModulePkg.dsc* indica la dirección del archivo de descripcion que contiene el listado de los archivos usados para la compilación de nuestra aplicación, cuando la compilacion termine enviara un mensaje de exito.

```
make: No se hace nada para 'tbuild'.
make: No se hace nada para 'tbuild'.
Building ... /home/rodrigo/src/edk2/MdeModulePkg/Universal/CapsulePei/CapsulePei.inf [X64]
Building ... /home/rodrigo/src/edk2/MdeModulePkg/Universal/Variable/MmVariablePei/MmVariablePei.inf [X64]
make: No se hace nada para 'tbuild'.
Building ... /home/rodrigo/src/edk2/MdeModulePkg/Bus/Pci/UfsPciHcPei/UfsPciHcPei.inf [X64]
make: No se hace nada para 'tbuild'.
make: No se hace nada para 'tbuild'.
make: No se hace nada para 'tbuild'.
make: No se hace nada para 'tbuild'.

- Done -
Build end time: 22:19:09, Aug.13 2024
Build total time: 00:01:12

rodrigo@raspberrypi:~/src/edk2 $
```

Ahora para usar nuestro paquete debemos de copiarlo o moverlo a una USB con sistema de archivo FAT32, el cual se encuentra en la direcccion siguiente:

```
rodrigo@raspberrypi:~/src/edk2/Build/MdeModule/DEBUG_GCC5/X64 $ pwd
/home/rodrigo/src/edk2/Build/MdeModule/DEBUG_GCC5/X64
rodrigo@raspberrypi:~/src/edk2/Build/MdeModule/DEBUG_GCC5/X64 $ ls HelloWorld.efi
HelloWorld.efi
rodrigo@raspberrypi:~/src/edk2/Build/MdeModule/DEBUG_GCC5/X64 $
```

El nombre del paquete construido fue *HelloWorld.efi*. Lo copiamos a nuestra USB:



Y ejecutamos nuestra maquina virtual con acceso a la unidad USB, para esto necesitamos averiguar el puerto en el que se encuentra nuestra usb:

```
rodrigo@raspberrypi:~/src/edk2 $ lsusb
Bus 001 Device 004: ID ffff:5678 Blackpcs
Bus 001 Device 003: ID 0424:ec00 Microchip Technology, Inc. (formerly SMSC) SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. (formerly SMSC) SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
rodrigo@raspberrypi:~/src/edk2 $
```

En este caso es el puerto 4, tomamos esta informacion y la agregamos a los parametros de nuestra llamada a Qemu:

```
rodrigo@raspberrypi:~/src/edk2 $ lsusb
Bus 001 Device 004: ID ffff:5678 Blackpcs
Bus 001 Device 003: ID 0424:ec00 Microchip Technology, Inc. (formerly SMSC) SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. (formerly SMSC) SMC9514 Hub
rodrigo@raspberrypi:~/src/edk2 $ sudo qemu-system-x86_64 -bios OVMF.fd -m 256M -net none -nographic -usb -device usb-ehci,id=ehci -device usb-host,hostbus=1,hostaddr=4
```

Observe como el numero de Bus y el numero de direccion de puerto coinciden, una vez iniciada la sesion virtual del UEFI-Shell:

```
UEFI Interactive Shell v2.2
edk2-stable202311 (https://github.com/pbatard/UEFI-Shell)
UEFI v2.70 (EDK II, 0x00010000)
Mapping table
      FS0: Alias(s):HD1a0b:;BLK2:
           PciRoot(0x0)/Pci(0x3,0x0)/USB(0x0,0x0)/HD(1,MBR,0x0674A67F,0x800,0x3A97800)
     BLK0: Alias(s):
           PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
     BLK1: Alias(s):
           PciRoot(0x0)/Pci(0x3,0x0)/USB(0x0,0x0)

Press ESC in 4 seconds to skip startup.nsh or any other key to continue.
Shell> fs0:
FS0:\> HelloWorld.efi
UEFI Hello World!
FS0:\>
```

Accedemos a la memoria USB con *fs0:* indicado en (A), escribimos el nombre de nuestra aplicacion *HelloWorld.efi* (B) y observamos el resultado en (C) donde se muestra el mensaje UEFI Hello World!.

APENDICE A

Como instalar un ambiente de diseño usando el proyecto EDK2

Comenzamos creando un directorio de trabajo, para este ejemplo lo llamaremos EDK2

```
rodrigo@raspberrypi:~ $ ls
0         coreboot  Documents  edk2        edk2-platforms  file  LCD-show  Pictures  src       thinclient_drives  UEFI-GPT-image-creator  VMs
Bookshelf  Desktop  Downloads  edk2-docker  exor_uav        ha_tb  Music    Public    Templates  uefi-dev           Videos                  ZHORKSPACE
rodrigo@raspberrypi:~ $ sudo mkdir EDK2
rodrigo@raspberrypi:~ $ ls
0         coreboot  Documents  edk2  edk2-docker     exor_uav  ha_tb     Music     Public  Templates          uefi-dev              Videos  ZHORKSPACE
Bookshelf  Desktop  Downloads  EDK2  edk2-platforms  file      LCD-show  Pictures  src     thinclient_drives  UEFI-GPT-image-creator  VMs
rodrigo@raspberrypi:~ $ cd EDK2/
rodrigo@raspberrypi:~/EDK2 $ ls
rodrigo@raspberrypi:~/EDK2 $
```

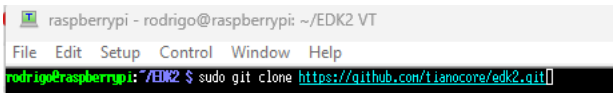
Dentro clonamos el repositorio de Git Hub, con las siguientes instrucciones:

sudo git clone --recurse -submodules https://github.com/tianocore/edk2.git

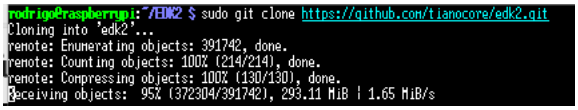
Si presenta problemas de conección use los comandos separados:

git clone https://github.com/tianocore/edk2.git

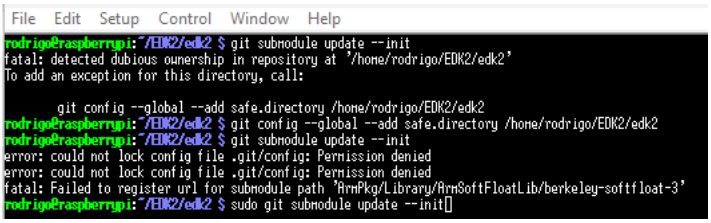
git submodule update --init #si ocurre un problema solo siga las instrucciones en pantalla



Dependiendo de su conección a Internet esta operacion tardara algún tiempo en completarse



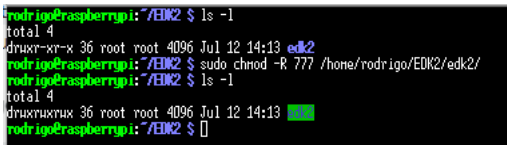
Nota: en caso de algun problema siga las intrucciones que le indican



Esta operacion tomara cierto tiempo en completarse si falla en algún punto simplemente re-ejecute el comando.

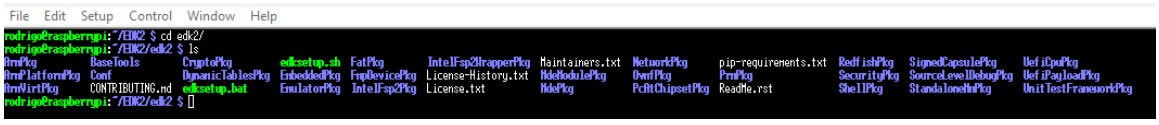
Tras completar ejecute el segundo comando para actualizar el repositorio, sera necesario otorgar permisos completos a todos los grupos a la carpeta recien creada (edk2):

sudo chmod -R 777 /home/rodrigo/EDK2/edk2/



Esto permitira que los diferentes scripts puedan terminar la configuración sin reestriccion alguna. Ingrese al directorio que se genero tras clonar el repositorio:

cd ./edk2



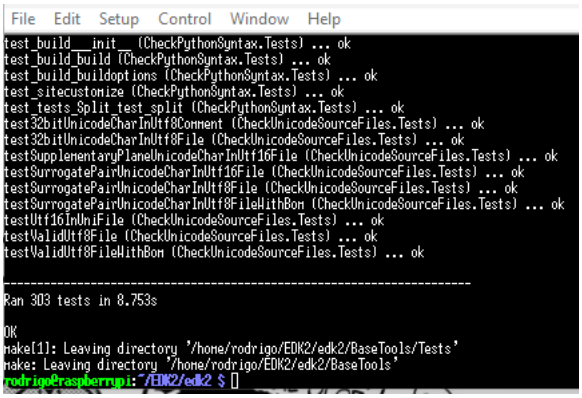
Y ejecute el comando (observe que la carpeta esta llena de los archivos del proyecto):

sudo ./edksetup.sh

Para asegurar que la configuracion de los directorios de trabajo es la correcta, confirmada la configuracion ejecute el comando siguiente para crear las herramientas basicas de compilación:

sudo make -C BaseTools/

La operacion tomara tiempo en completarse



Observe que no haya habido errores durante la ejecución, encaso de existir vuelva e ejecutar el comando y preste atención a los mensajes de error.

Ahora realicemos los ajustes en el archivo de configuracion para indicar que tipo de archivo EDK2 UEFI de arranque queremos crear, utilice el comando:

sudo vi Conf/target.txt

En el archivo encuentre las variables que se muestran en la siguiente tabla:

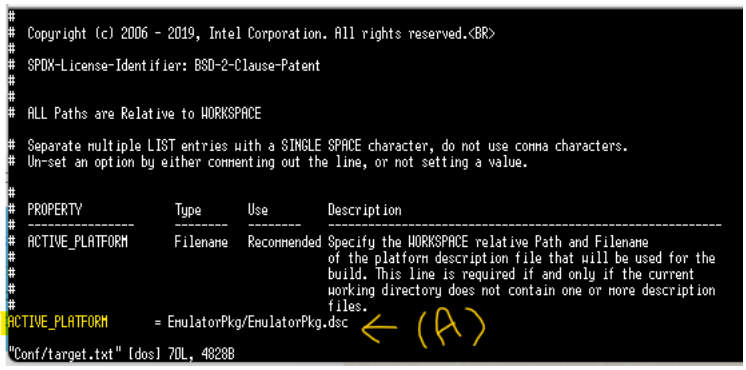
ACTIVE_PLATFORM = ArmVirtPkg/ArmVirtQemu.dsc

TARGET = DEBUG

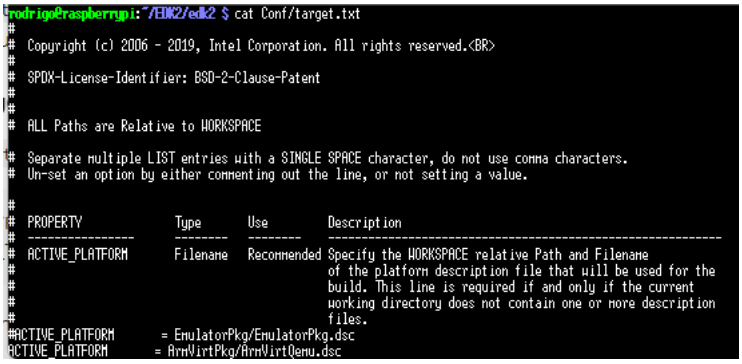
TARGET_ARCH = AARCH64

TOOL_CHAIN_TAG = GCC5

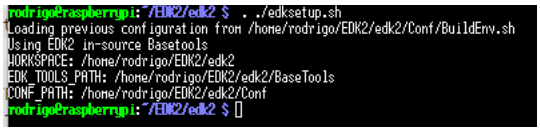
Modifique los valores en caso de ser necesario:



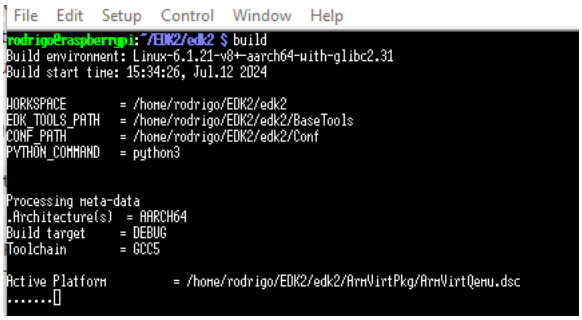
La figura en (A) muestra el valor de configuración de ACTIVE_PLATFORM, en este caso debe cambiarse al valor anteriormente indicado (en este caso comentamos la linea y sustituimos con una que tiene el valor requerido).



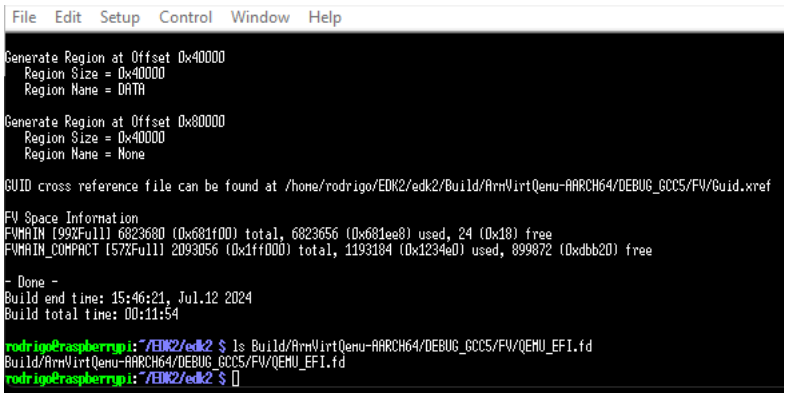
Tras realizar la misma accion en todas las lineas requeridas, volvemos a ejecutar el comando de configuracion:



Seguido de el comando "Build" (el cual es de python) para comenzar la creación de nuestro archivo de EDK2-BIOS para nuestra Maquina Virtual.



Comenzara el proceso de compilación de nuestro archivo indicado por la "barra de progreso", este proceso puede tardar un momento el cual dependera de las capacidades de computo de nuestro hardware.

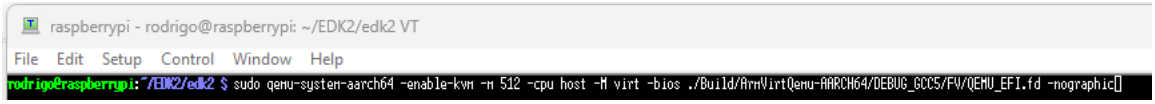


Cuando la operación termine pude ubicar el archivo usando:

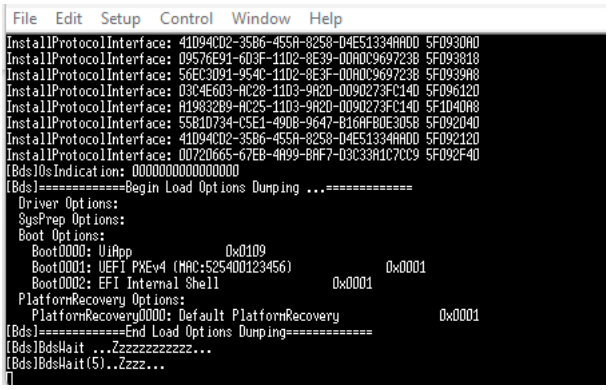
ls Build/ArmVirtQemu-AARCH64/DEBUG_GCC5/FV/QEMU_EFI.fd

Puede ejecutar el archivo en la maquina virtual de Qemu, con el siguiente comando:

sudo qemu-system-aarch64 -enable-kvm -m 512 -cpu host -M virt -bios ./Build/ArmVirtQemu-AARCH64/DEBUG_GCC5/FV/QEMU_EFI.fd -nographic



Cuando termine la ejecucion veremos la salida que nos mostraria el puerto de Debug de nuesta Maquina Virtual.



Hemos completado la creación de un IFWI-BIOS para nuestra maquina virtual

APENDICE B

Instalacion del software de virtualización Qemu

Antes de realizar accion alguna asegure que su sistema operativo esta actualizado:

Posteriormente descargue en instale los paquetes requeridos para el funcionamiento correcto del software de virtualizacion:

```
rodrigo@raspberrypi:~ $ sudo apt-get install qemu-system libvirt-daemon-system virt-manager
Leyendo lista de paquetes... Hecho
Creando ├írbol de dependencias... Hecho
Leyendo la informaci├│n de estado... Hecho
libvirt-daemon-system ya est├í en su versi├│n m├ís reciente (7.0.0-3+deb11u2).
qemu-system ya est├í en su versi├│n m├ís reciente (1:5.2+dfsg-11+deb11u3).
virt-manager ya est├í en su versi├│n m├ís reciente (1:3.2.0-3).
0 actualizados, 0 nuevos se instalar├ín, 0 para eliminar y 0 no actualizados.
```

Los tres paquetes basicos son "*quemu-system*", *"libvirt-daemon-system" y "virt-manager"*, asi mismo si requiere instale el paqute de Qemu:

```
rodrigo@raspberrypi:~ $ sudo apt-get install qemu
Leyendo lista de paquetes... Hecho
Creando ├írbol de dependencias... Hecho
Leyendo la informaci├│n de estado... Hecho
qemu ya est├í en su versi├│n m├ís reciente (1:5.2+dfsg-11+deb11u3).
0 actualizados, 0 nuevos se instalar├ín, 0 para eliminar y 0 no actualizados.
```

Tras haber instalado los paquetes de de alta su usuario del sistema para que pueda acceder a los servicios de virtualizacion:

```
rodrigo@raspberrypi:~ $ sudo adduser rodrigo libvirt
El usuario 'rodrigo' ya es un miembro de 'libvirt'.
```

Sustituya su nombre de ususario en la linea de comando anterior, y para que los programas de segundo plano del software virtual se activen reincie el sistema:

```
rodrigo@raspberrypi:~ $ sudo reboot
```

Cuando el sistema vuelva a iniciar, compruebe que funciona con la siguiente linea de comandos:

```
rodrigo@raspberrypi:~ $ sudo qemu-system-x86_64 -bios OVMF.fd -m 256M -net none -nographic
```

Esto inicializara una sesion virtual del UEFI-Shell:

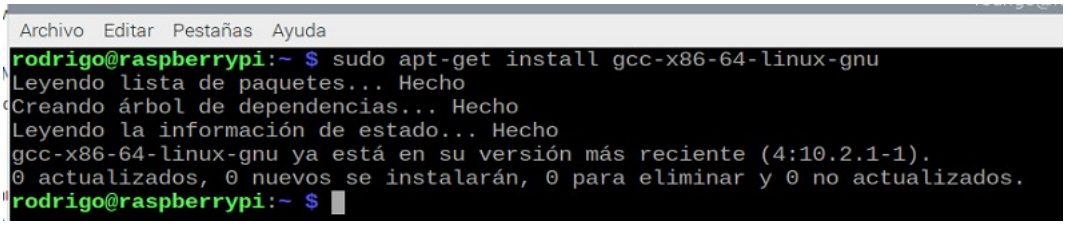
```
        PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
Press ESC in 1 seconds to skip startup.nsh or any other key to continue.
Shell> help reset
Resets the system.

RESET [-w [string]]
RESET [-s [string]]
RESET [-c [string]]

  -s      - Performs a shutdown.
  -w      - Performs a warm boot.
  -c      - Performs a cold boot.
  string - Describes a reason for the reset.

NOTES:
  1. This command resets the system.
  2. The default is to perform a cold reset unless the -w parameter is
     specified.
  3. If a reset string is specified, it is passed into the ResetSystem()
     function, and the system records the reason for the system reset.
Shell> reset -s
Reset with <null string> (0 bytes)rodrigo@raspberrypi:~ $
```

Para terminar la sesion use la orden "*reset -s*" en el prompt del UEFI-Shell y regresera al Prompt de la raspberry pi.

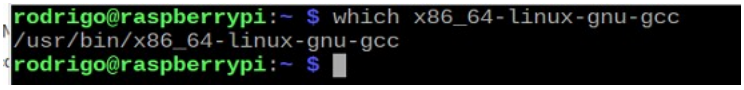
APENDICE C

Compilacion de modulos EDK2 para arquitectura X86_64 usando una raspberry pi

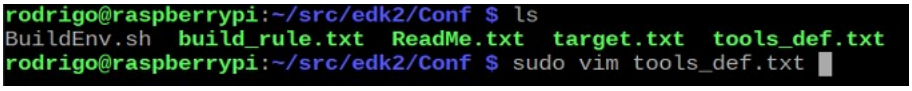
Tras haber completado las instrucciones del **Apendice A**, y verificar que contamos con un ambiente de desarrollo funcional (al menos para Arquitectura AARCH64=ARM), debemos realizar unos ajustes adicionales a las herramientas del paquete EDK2 para poder compilar para arquitectura x86_64, lo primero es instalar la familia de compilacion GCC requerida por las herramientas:



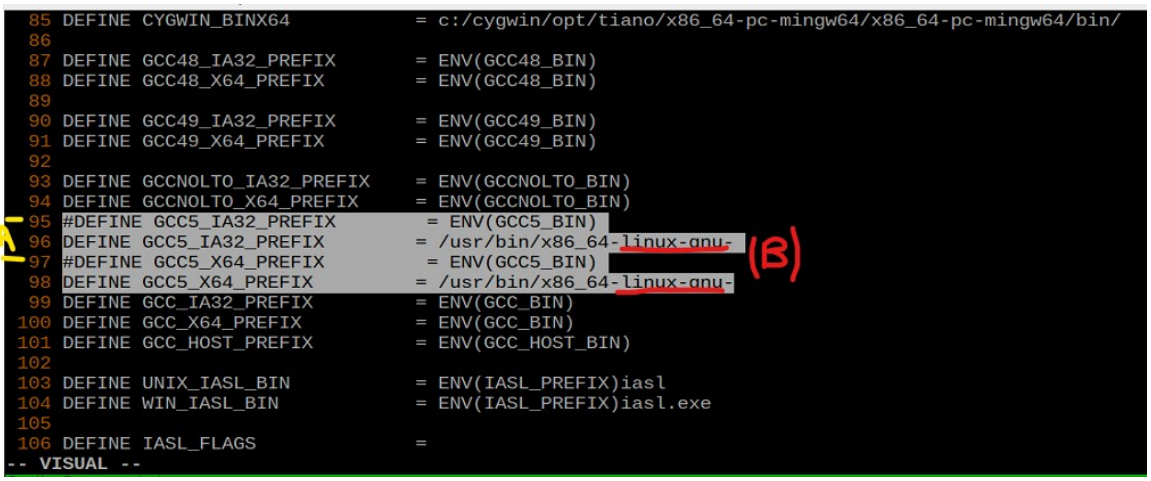
Tras instalar el paquete del compilador, verificamos la ruta en la cual fue instaldo el compilador:



Copiamos esta dirección para modificar el archivo de configuracion ubicado en la carpeta *Conf* de nuestro ambiente de desarollo:



Usamos el editor de nuestra preferencia, (en este caso Vim con la opcion de numeros de linea):



Copie las lineas que se muestran en (A) y las "originales" las puede borrar o comentar, sustituya las nuevas lineas con la informacion mostrada (B), la cual es la dirección donde esta instalalado el compilador.

Cuando compile para arquitectura X64 o IA32, usando el script de python *build* ajuste el parametro -t

GCC5 (tool chain GCC5), ya que fue la variable de ambiente para el compilador que modificamos. Por ejemplo, compilar OVMF.fd (archivo de Qemu) usaremos:

```
rodrigo@raspberrypi:~/src/edk2 $ build -a X64 -t GCC5 -b DEBUG -p MdeModulePkg/MdeModulePkg.dsc
```

Tras termine el proceso de compilacion deberemos ver el mensaje de compilación terminada sin errores: