RPi
and
64-Bit and 32-Bit Code

NORMAN MCENTIRE

Recommended Release for RPi is now 64-bits

- As of 2023 October (previous recommendation was 32-bits)
- Now, by default, all utils are 64-bit
- And when you build on host, gcc is 64-bit

```
$ file /usr/bin/bash
/usr/bin/bash: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically linked,
interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=198ac6b0cc01bf774c5746a36a6fd31dead9bda5, for GNU/Linux 3.7.0, stripped
```

Our Goal

- Our Goal is to write code that builds for both 32-bit and 64-bit
  - Native Compiler will be 64-bit aarch64
  - Cross Compiler will be 32-bit armv7l

Point of Reference (Brand new Install)

```
$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"

$ arch
aarch64


$ dpkg --print-architecture
arm64

$ dpkg --print-foreign-architectures
armhf
```

KEY POINT! Notice that aarch64 has already installed the foreign A=architecture of armhf

hello.c - aarch64

```
$ cat hello.c
#include <stdio.h>

int main() {
  puts("Hello World");
  return 0;
}
```

```
$ gcc -Wall -o hello64 hello.c

file hello64
hello64: ELF 64-bit LSB pie executable, ARM aarch64,
version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1,
BuildID[sha1]=c342bc4e25cbaed2327d162631fa7b9b4a24c7e3,
for GNU/Linux 3.7.0, not stripped
```

So the native host compiler is **64-bit!**
But how to compile for 32-bit?

Install Cross Compiler - armhf

```
$ sudo apt-get update
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian-security bookworm-security InRelease
Hit:3 http://deb.debian.org/debian bookworm-updates InRelease
Hit:4 http://archive.raspberrypi.com/debian bookworm InRelease
Reading package lists... Done

$ sudo apt-get install gcc-arm-linux-gnueabihf
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils-arm-linux-gnueabihf cpp-12-arm-linux-gnueabihf cpp-arm-linux-gnueabihf
  gcc-12-arm-linux-gnueabihf gcc-12-arm-linux-gnueabihf-base gcc-12-cross-base libasan8-armhf-cross
  libatomic1-armhf-cross libc6-armhf-cross libc6-dev-armhf-cross libgcc-12-dev-armhf-cross
  libgcc-s1-armhf-cross libgomp1-armhf-cross libstdc++6-armhf-cross libubsan1-armhf-cross
  linux-libc-dev-armhf-cross
. . .
After this operation, 122 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Results - Cross Toolchain

```
arm-linux-gnueabihf-
arm-linux-gnueabihf-addr2line
arm-linux-gnueabihf-ar
arm-linux-gnueabihf-as
arm-linux-gnueabihf-c++filt
arm-linux-gnueabihf-cpp
arm-linux-gnueabihf-cpp-12
arm-linux-gnueabihf-dwp
arm-linux-gnueabihf-elfedit
arm-linux-gnueabihf-gcc
arm-linux-gnueabihf-gcc-12
arm-linux-gnueabihf-gcc-ar
arm-linux-gnueabihf-gcc-ar-12
arm-linux-gnueabihf-gcc-nm
arm-linux-gnueabihf-gcc-nm-12
arm-linux-gnueabihf-gcc-ranlib
arm-linux-gnueabihf-gcc-ranlib-12
arm-linux-gnueabihf-gcov
arm-linux-gnueabihf-gcov-12
arm-linux-gnueabihf-gcov-dump
arm-linux-gnueabihf-gcov-dump-12
arm-linux-gnueabihf-gcov-tool
arm-linux-gnueabihf-gcov-tool-12
arm-linux-gnueabihf-gprof
arm-linux-gnueabihf-ld
arm-linux-gnueabihf-ld.bfd
arm-linux-gnueabihf-ld.gold
arm-linux-gnueabihf-lto-dump
arm-linux-gnueabihf-lto-dump-12
arm-linux-gnueabihf-nm
arm-linux-gnueabihf-objcopy
arm-linux-gnueabihf-objdump
arm-linux-gnueabihf-ranlib
arm-linux-gnueabihf-readelf
arm-linux-gnueabihf-size
arm-linux-gnueabihf-strings
arm-linux-gnueabihf-strip
```

Results

```
4 arm-linux-gnueabihf-gcc -Wall -o hello32 hello.c

$ file hello32
hello32: ELF 32-bit LSB pie executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3,
BuildID[sha1]=b95b0a16f8e89f765f7687c1931bc9117cfcfe59, for GNU/Linux 3.2.0, not stripped


$ ldd hello32
  not a dynamic executable

$ ldd hello64
  linux-vdso.so.1 (0x0000007f883e1000)
  libc.so.6 => /lib/aarch64-linux-gnu/libc.so.6 (0x0000007f881d0000)
  /lib/ld-linux-aarch64.so.1 (0x0000007f883a4000)

$ ./hello64
Hello World


$ ./hello32
-bash: ./hello32: cannot execute: required file not found
```

readelf -a hello32 vs readelf -a hello64

```
$ readelf -a hello32
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x409
  Start of program headers:          52 (bytes into file)
  Start of section headers:          6692 (bytes into file)
  Flags:                             0x5000400, Version5 EABI, hard-float ABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         9
  Size of section headers:           40 (bytes)
  Number of section headers:         29
  Section header string table index: 28
```

```
$ readelf -a hello64
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           AArch64
  Version:                           0x1
  Entry point address:               0x640
  Start of program headers:          64 (bytes into file)
  Start of section headers:          68576 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 28
```

readelf -a hello32 vs readelf -a hello64

Interpreter

```
$ readelf -a hello32 | grep interpret
      [Requesting program interpreter: /lib/ld-linux-armhf.so.3]
```

```
$ readelf -a hello64 | grep interpret
      [Requesting program interpreter: /lib/ld-linux-aarch64.so.1]

$ ls /lib/ld-*
/lib/ld-linux-aarch64.so.1
```

apt-file list

```
$ apt-file /lib/ld-linux-armhf.so.3
-bash: apt-file: command not found

$ sudo apt-get update
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian-security bookworm-security InRelease
Hit:3 http://deb.debian.org/debian bookworm-updates InRelease
Hit:4 http://archive.raspberrypi.com/debian bookworm InRelease
Reading package lists... Done
. .

$ sudo apt-get install apt-file
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
. . .

$ sudo apt-file search /lib/ld-linux-armhf.so.3
Finding relevant cache files to search ...E: The cache is empty. You need to run "apt-file update" first.


$ sudo apt-file update
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian-security bookworm-security InRelease

. . .

$ sudo apt-file search /lib/ld-linux-armhf.so.3
libc6-armhf-cross: /usr/arm-linux-gnueabihf/lib/ld-linux-armhf.so.3
```

sudo apt-get instal libc6:armhf

```
$ sudo apt-get install libc6:armhf
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  gcc-12-base:armhf libgcc-s1:armhf libidn2-0:armhf libunistring2:armhf
Suggested packages:
  glibc-doc:armhf locales:armhf libnss-nis:armhf libnss-nisplus:armhf
The following NEW packages will be installed:
  gcc-12-base:armhf libc6:armhf libgcc-s1:armhf libidn2-0:armhf libunistring2:armhf
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,713 kB of archives.
After this operation, 11.4 MB of additional disk space will be used.
. . .
```

We can now run both 64-bit and 32-bit code

```
#include <stdio.h>
#include <inttypes.h> // for PRIuPTR

int main() {
  puts("Hello World");
  char *p = "hello";
  printf("sizeof(p): %" PRIuPTR "\n", sizeof(p));
  return 0;
}
```

```
$ ./hello32
Hello World
sizeof(p): 4
```

```
$ ./hello64
Hello World
sizeof(p): 8
```

Example Project: MicroPython

- Suppose your task is to build MicroPython
- You want to build MicroPython for both your 64-bit Host
- And for your 32-bit Target

About MicroPython

- MicroPython is Python3 for Embedded Systems
- Compatible with Python3 but optimized for embedded systems
  - Does not include ALL the Python3 standard library…but a lot of it!
- This will be covered in another slide deck/video