Embedded Linux Systems Programming Toolchain

2023-08-29

NORMAN MCENTIRE

References

- https://en.wikipedia.org/wiki/Toolchain
- https://en.wikipedia.org/wiki/GNU_toolchain
- https://en.wikipedia.org/wiki/GNU_Project
- https://en.wikipedia.org/wiki/Buildroot

Toolchain Introduction

- A toolchain is a set of programming tools
    - Compiler - compile source code into object code
    - Linker - link object code into executable code
    - Debugger
- Toolchain often installed on a Host and compiles for a Target
    - Host - the machine where you use the toolchain
        - Example: x86_64 PC
    - Target - the machine that runs the code created by the host
        - Example: armv7l embedded system

GNU Toolchain

- Widely used Toolchain produced by the GNU Project
    - GNU = GNU’s Not Unix
    - Free Software - “Free as in Freedom”
- Includes the following
    - GNU make
    - GNU Compiler Collection
    - GNU Library (glibc)
    - Binutils (linker, assembler, etc.)
    - GNU Debugger

Example Toolchain

```
$ ls arm*
armv7l-timesys-linux-uclibcgnueabi-addr2line   armv7l-timesys-linux-uclibcgnueabi-gcov-tool
armv7l-timesys-linux-uclibcgnueabi-ar          armv7l-timesys-linux-uclibcgnueabi-gprof
armv7l-timesys-linux-uclibcgnueabi-as          armv7l-timesys-linux-uclibcgnueabi-ld
armv7l-timesys-linux-uclibcgnueabi-c++         armv7l-timesys-linux-uclibcgnueabi-ld.bfd
armv7l-timesys-linux-uclibcgnueabi-c++filt     armv7l-timesys-linux-uclibcgnueabi-nm
armv7l-timesys-linux-uclibcgnueabi-cpp         armv7l-timesys-linux-uclibcgnueabi-objcopy
armv7l-timesys-linux-uclibcgnueabi-elfedit     armv7l-timesys-linux-uclibcgnueabi-objdump
armv7l-timesys-linux-uclibcgnueabi-g++         armv7l-timesys-linux-uclibcgnueabi-pkg-config
armv7l-timesys-linux-uclibcgnueabi-gcc         armv7l-timesys-linux-uclibcgnueabi-pkg-config.real
armv7l-timesys-linux-uclibcgnueabi-gcc-5.3.0   armv7l-timesys-linux-uclibcgnueabi-ranlib
armv7l-timesys-linux-uclibcgnueabi-gcc-ar      armv7l-timesys-linux-uclibcgnueabi-readelf
armv7l-timesys-linux-uclibcgnueabi-gcc-nm      armv7l-timesys-linux-uclibcgnueabi-size
armv7l-timesys-linux-uclibcgnueabi-gcc-ranlib  armv7l-timesys-linux-uclibcgnueabi-strings
armv7l-timesys-linux-uclibcgnueabi-gcov        armv7l-timesys-linux-uclibcgnueabi-strip
```

Example: Compiling hello.c: Host and Target

Host Native Toolchain on x86_64 Ubuntu

```
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

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

$ file hello
hello: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0,
 BuildID[sha1]=847ece746b56cba30c2cca74ef5fb73245b351c5, not stripped
```

Cross Development Toolchain - armv7l-timesys-linux-uclibcgnueabi

```
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

```
$ armv7l-timesys-linux-uclibcgnueabi-gcc -Wall -o hello hello.c

$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked,
interpreter /lib/ld-uClibc.so.0, with debug_info, not stripped
```

Host Native Toolchain - Raspberry Pi armv7l

```
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

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

$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked,
 interpreter /lib/ld-linux-armhf.so.3, BuildID[sha1]=31cd63ca6cbb0712b1cf6cc4b740d7a9cd44f8de,
for GNU/Linux 3.2.0, not stripped
```

nm - List Symbols from Object File - x86_64

Host Native Toolchain

```
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

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

$ file hello
hello: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0,
 BuildID[sha1]=847ece746b56cba30c2cca74ef5fb73245b351c5, not stripped
```

```
$ nm hello
0000000000200dc8 d _DYNAMIC
0000000000200fb8 d _GLOBAL_OFFSET_TABLE_
00000000000006e0 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000000834 r __FRAME_END__
00000000000006f0 r __GNU_EH_FRAME_HDR
0000000000201010 D __TMC_END__
0000000000201010 B __bss_start
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000201000 D __data_start
00000000000005f0 t __do_global_dtors_aux
0000000000200dc0 t __do_global_dtors_aux_fini_array_entry
0000000000201008 D __dso_handle
0000000000200db8 t __frame_dummy_init_array_entry
                 w __gmon_start__
0000000000200dc0 t __init_array_end
0000000000200db8 t __init_array_start
00000000000006d0 T __libc_csu_fini
0000000000000660 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
0000000000201010 D _edata
0000000000201018 B _end
00000000000006d4 T _fini
00000000000004e8 T _init
0000000000000530 T _start
0000000000201010 b completed.7698
0000000000201000 W data_start
0000000000000560 t deregister_tm_clones
0000000000000630 t frame_dummy
000000000000063a T main
                 U puts@@GLIBC_2.2.5
00000000000005a0 t register_tm_clones
```

T - Global Text Symbol (main is entry point)

U - Global Undefined Text Symbol located in glibc

nm - List Symbols from Object File - armv7l

Host Native Toolchain

```
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

```
$ armv7l-timesys-linux-uclibcgnueabi-gcc -Wall -o hello hello.c

$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked,
interpreter /lib/ld-uClibc.so.0, with debug_info, not stripped
```

```
$ nm hello
00020514 d _DYNAMIC
000205cc d _GLOBAL_OFFSET_TABLE_
         w _ITM_deregisterTMCloneTable
         w _ITM_registerTMCloneTable
         w _Jv_RegisterClasses
00010504 r __EH_FRAME_BEGIN__
. . .
0002050c t __do_global_dtors_aux_fini_array_entry
000205f0 D __dso_handle
00020610 B __end__
00020508 t __frame_dummy_init_array_entry
         w __register_frame_info
         U __uClibc_main
00020610 B _bss_end__
000103b0 W _call_via_fp
000103b4 W _call_via_ip
000103bc W _call_via_lr
00010384 W _call_via_r0
00010388 W _call_via_r1
. . .
000103ac W _call_via_sl
000103b8 W _call_via_sp
000205f4 D _edata
00020610 B _end
000104e8 T _fini
000102e8 T _init
00010348 T _start
         U abort
000205f4 b completed.9157
000205ec W data_start
000103c0 t deregister_tm_clones
0001046c t frame_dummy
000104c8 T main
000205f8 b object.9162
         U puts
000103f0 t register_tm_clones
```

T - Global Text Symbol (main is entry point)

U - Global Undefined Text Symbol located in glibc

nm - List Symbols from Object File - RPi/armv7l

Host Native Toolchain

```
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

```
$ armv7l-timesys-linux-uclibcgnueabi-gcc -Wall -o hello hello.c

$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked,
interpreter /lib/ld-uClibc.so.0, with debug_info, not stripped
```

```
$ nm hello
         U abort@GLIBC_2.4
00010494 r all_implied_fbits
00010530 r all_implied_fbits
0002102c B __bss_end__
0002102c B _bss_end__
00021028 B __bss_start
00021028 B __bss_start__
00010350 t call_weak_fn
00021028 b completed.0
00021020 D __data_start
00021020 W data_start
00010374 t deregister_tm_clones
000103d8 t __do_global_dtors_aux
00020f14 d __do_global_dtors_aux_fini_array_entry
00021024 D __dso_handle
00020f18 d _DYNAMIC
00021028 D _edata
0002102c B __end__
0002102c B _end
00010488 T _fini
00010400 t frame_dummy
00020f10 d __frame_dummy_init_array_entry
000105c8 r __FRAME_END__
00021000 d _GLOBAL_OFFSET_TABLE_
         w __gmon_start__
000102c4 T _init
00020f14 d __init_array_end
00020f10 d __init_array_start
00010490 R _IO_stdin_used
00010484 T __libc_csu_fini
00010424 T __libc_csu_init
         U __libc_start_main@GLIBC_2.4
00010404 T main
         U puts@GLIBC_2.4
000103a0 t register_tm_clones
00010314 T _start
00021028 D __TMC_END__
```

T - Global Text Symbol (main is entry point)

U - Global Undefined Text Symbol located in glibc

Demo: Build on RPi, move to Embedded System

- For RPi code to run on target embedded system
  - Architectures must match, e.g. armv7l
  - C library must match: glibc vs uclibc
- For the next demo:
  - We will build on the RPi armv7l/glibc, then move to code to an embedded system running armv7l/uclibc
    - At first it will not work….but we will make changes to make it work!

Step 1. Build for RPi/armv7l, move to embedded system and try running

```
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
$ gcc -Wall -o hello hello.c

$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked,
 interpreter /lib/ld-linux-armhf.so.3, BuildID[sha1]=31cd63ca6cbb0712b1cf6cc4b740d7a9cd44f8de,
for GNU/Linux 3.2.0, not stripped

$ ./hello
Hello World

$ cp hello hello-rpi

$ scp hello-rpi root@10.176.100.92:.

$ ssh root@10.176.100.92

# uname -a
Linux custom-soc 4.4.106-ts-armv7l #1 PREEMPT Sun Jan 1 00:00:00 EST 2017 armv7l GNU/Linux
#
# ./hello-rpi
-sh: ./hello-rpi: not found

# ls -l hello-rpi
-rwxr-xr-x    1 root     root          8072 Aug 29 17:34 hello-rpi
```

What?
Why does it say not found?

Step 2a. readelf (-a for all info)

RPI Toolchain

```
$ readelf -a hello-rpi
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:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x10314
  Start of program headers:          52 (bytes into file)
  Start of section headers:          6912 (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
```

Timesys Toolchain

```
$ armv7l-timesys-linux-uclibcgnueabi-readelf -a hello-timesys
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:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x10348
  Start of program headers:          52 (bytes into file)
  Start of section headers:          5688 (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:         6
  Size of section headers:           40 (bytes)
  Number of section headers:         29
  Section header string table index: 26
```

NOTE: So far so good, both match!

Step 2b. readelf (-a for all)

```
Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        00010154 000154 000019 00   A  0   0  1
  [ 2] .note.gnu.bu[...] NOTE            00010170 000170 000024 00   A  0   0  4
  [ 3] .note.ABI-tag     NOTE            00010194 000194 000020 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        000101b4 0001b4 00002c 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          000101e0 0001e0 000050 10   A  6   1  4
  [ 6] .dynstr           STRTAB          00010230 000230 000041 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          00010272 000272 00000a 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         0001027c 00027c 000020 00   A  6   1  4
  [ 9] .rel.dyn          REL             0001029c 00029c 000008 08   A  5   0  4
  [10] .rel.plt          REL             000102a4 0002a4 000020 08  AI  5  21  4
  [11] .init             PROGBITS        000102c4 0002c4 00000c 00  AX  0   0  4
  [12] .plt              PROGBITS        000102d0 0002d0 000044 04  AX  0   0  4
  [13] .text             PROGBITS        00010314 000314 000174 00  AX  0   0  4
  [14] .fini             PROGBITS        00010488 000488 000008 00  AX  0   0  4
  [15] .rodata           PROGBITS        00010490 000490 000130 00   A  0   0  4
  [16] .ARM.exidx        ARM_EXIDX       000105c0 0005c0 000008 00  AL 13   0  4
  [17] .eh_frame         PROGBITS        000105c8 0005c8 000004 00   A  0   0  4
  [18] .init_array       INIT_ARRAY      00020f10 000f10 000004 04  WA  0   0  4
  [19] .fini_array       FINI_ARRAY      00020f14 000f14 000004 04  WA  0   0  4
  [20] .dynamic          DYNAMIC         00020f18 000f18 0000e8 08  WA  6   0  4
  [21] .got              PROGBITS        00021000 001000 000020 04  WA  0   0  4
  [22] .data             PROGBITS        00021020 001020 000008 00  WA  0   0  4
  [23] .bss              NOBITS          00021028 001028 000004 00  WA  0   0  1
  [24] .comment          PROGBITS        00000000 001028 00002e 01  MS  0   0  1
  [25] .ARM.attributes   ARM_ATTRIBUTES  00000000 001056 00002f 00      0   0  1
  [26] .symtab           SYMTAB          00000000 001088 0006a0 10     27  84  4
  [27] .strtab           STRTAB          00000000 001728 0002d0 00      0   0  1
  [28] .shstrtab         STRTAB          00000000 0019f8 000105 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  y (purecode), p (processor specific)
```

```
Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        000100f4 0000f4 000014 00   A  0   0  1
  [ 2] .hash             HASH            00010108 000108 00004c 04   A  3   0  4
  [ 3] .dynsym           DYNSYM          00010154 000154 0000e0 10   A  4   1  4
  [ 4] .dynstr           STRTAB          00010234 000234 00008c 00   A  0   0  1
  [ 5] .rel.plt          REL             000102c0 0002c0 000028 08  AI  3  16  4
  [ 6] .init             PROGBITS        000102e8 0002e8 000010 00  AX  0   0  4
  [ 7] .plt              PROGBITS        000102f8 0002f8 000050 04  AX  0   0  4
  [ 8] .text             PROGBITS        00010348 000348 0001a0 00  AX  0   0  4
  [ 9] .fini             PROGBITS        000104e8 0004e8 000010 00  AX  0   0  4
  [10] .rodata           PROGBITS        000104f8 0004f8 00000c 00   A  0   0  4
  [11] .eh_frame         PROGBITS        00010504 000504 000004 00   A  0   0  4
  [12] .init_array       INIT_ARRAY      00020508 000508 000004 00  WA  0   0  4
  [13] .fini_array       FINI_ARRAY      0002050c 00050c 000004 00  WA  0   0  4
  [14] .jcr              PROGBITS        00020510 000510 000004 00  WA  0   0  4
  [15] .dynamic          DYNAMIC         00020514 000514 0000b8 08  WA  4   0  4
  [16] .got              PROGBITS        000205cc 0005cc 000020 04  WA  0   0  4
  [17] .data             PROGBITS        000205ec 0005ec 000008 00  WA  0   0  4
  [18] .bss              NOBITS          000205f4 0005f4 00001c 00  WA  0   0  4
  [19] .comment          PROGBITS        00000000 0005f4 00004c 01  MS  0   0  1
  [20] .ARM.attributes   ARM_ATTRIBUTES  00000000 000640 00002f 00      0   0  1
  [21] .debug_aranges    PROGBITS        00000000 000670 000078 00      0   0  8
  [22] .debug_info       PROGBITS        00000000 0006e8 0002bf 00      0   0  1
  [23] .debug_abbrev     PROGBITS        00000000 0009a7 000038 00      0   0  1
  [24] .debug_line       PROGBITS        00000000 0009df 00013c 00      0   0  1
  [25] .debug_ranges     PROGBITS        00000000 000b20 000048 00      0   0  8
  [26] .shstrtab         STRTAB          00000000 00153d 0000f8 00      0   0  1
  [27] .symtab           SYMTAB          00000000 000b68 0006b0 10     28  69  4
  [28] .strtab           STRTAB          00000000 001218 000325 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
```

Step 2c. readelf (-a for all)

RPI Toolchain

```
There are no section groups in this file.

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  EXIDX          0x0005c0 0x000105c0 0x000105c0 0x00008 0x00008 R   0x4
  PHDR           0x000034 0x00010034 0x00010034 0x00120 0x00120 R   0x4
  INTERP         0x000154 0x00010154 0x00010154 0x00019 0x00019 R   0x1
      [Requesting program interpreter: /lib/ld-linux-armhf.so.3]
  LOAD           0x000000 0x00010000 0x00010000 0x005cc 0x005cc R E 0x10000
  LOAD           0x000f10 0x00020f10 0x00020f10 0x00118 0x0011c RW  0x10000
  DYNAMIC        0x000f18 0x00020f18 0x00020f18 0x000e8 0x000e8 RW  0x4
  NOTE           0x000170 0x00010170 0x00010170 0x00044 0x00044 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x10
  GNU_RELRO      0x000f10 0x00020f10 0x00020f10 0x000f0 0x000f0 R   0x1
```

Timesys Toolchain

```
There are no section groups in this file.

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x00010034 0x00010034 0x000c0 0x000c0 R E 0x4
  INTERP         0x0000f4 0x000100f4 0x000100f4 0x00014 0x00014 R   0x1
      [Requesting program interpreter: /lib/ld-uClibc.so.0]
  LOAD           0x000000 0x00010000 0x00010000 0x00508 0x00508 R E 0x10000
  LOAD           0x000508 0x00020508 0x00020508 0x000ec 0x00108 RW  0x10000
  DYNAMIC        0x000514 0x00020514 0x00020514 0x000b8 0x000b8 RW  0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x10
```

NOTE! Program Interpreters are different!

Step 2d. readelf (-a for all)

```
Dynamic section at offset 0xf18 contains 24 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x102c4
 0x0000000d (FINI)                       0x10488
 0x00000019 (INIT_ARRAY)                 0x20f10
 0x0000001b (INIT_ARRAYSZ)               4 (bytes)
 0x0000001a (FINI_ARRAY)                 0x20f14
 0x0000001c (FINI_ARRAYSZ)               4 (bytes)
 0x6ffffef5 (GNU_HASH)                   0x101b4
 0x00000005 (STRTAB)                     0x10230
 0x00000006 (SYMTAB)                     0x101e0
 0x0000000a (STRSZ)                      65 (bytes)
 0x0000000b (SYMENT)                     16 (bytes)
 0x00000015 (DEBUG)                      0x0
 0x00000003 (PLTGOT)                     0x21000
 0x00000002 (PLTRELSZ)                   32 (bytes)
 0x00000014 (PLTREL)                     REL
 0x00000017 (JMPREL)                     0x102a4
 0x00000011 (REL)                        0x1029c
 0x00000012 (RELSZ)                      8 (bytes)
 0x00000013 (RELENT)                     8 (bytes)
 0x6ffffffe (VERNEED)                    0x1027c
 0x6fffffff (VERNEEDNUM)                 1
 0x6ffffff0 (VERSYM)                     0x10272
 0x00000000 (NULL)                       0x0
```

```
Dynamic section at offset 0x514 contains 18 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libc.so.0]
 0x0000000c (INIT)                       0x102e8
 0x0000000d (FINI)                       0x104e8
 0x00000019 (INIT_ARRAY)                 0x20508
 0x0000001b (INIT_ARRAYSZ)               4 (bytes)
 0x0000001a (FINI_ARRAY)                 0x2050c
 0x0000001c (FINI_ARRAYSZ)               4 (bytes)
 0x00000004 (HASH)                       0x10108
 0x00000005 (STRTAB)                     0x10234
 0x00000006 (SYMTAB)                     0x10154
 0x0000000a (STRSZ)                      140 (bytes)
 0x0000000b (SYMENT)                     16 (bytes)
 0x00000015 (DEBUG)                      0x0
 0x00000003 (PLTGOT)                     0x205cc
 0x00000002 (PLTRELSZ)                   40 (bytes)
 0x00000014 (PLTREL)                     REL
 0x00000017 (JMPREL)                     0x102c0
 0x00000000 (NULL)                       0x0
```

Step 2e. readelf (-a for all)

```
Version symbols section '.gnu.version' contains 5 entries:
 Addr: 0x0000000000010272  Offset: 0x000272  Link: 5 (.dynsym)
  000:   0 (*local*)       0 (*local*)       2 (GLIBC_2.4)     2 (GLIBC_2.4)
  004:   2 (GLIBC_2.4)

Version needs section '.gnu.version_r' contains 1 entry:
 Addr: 0x000000000001027c  Offset: 0x00027c  Link: 6 (.dynstr)
  000000: Version: 1  File: libc.so.6  Cnt: 1
  0x0010:   Name: GLIBC_2.4  Flags: none  Version: 2

Displaying notes found in: .note.gnu.build-id
  Owner                 Data size Description
  GNU                   0x00000014NT_GNU_BUILD_ID (unique build ID bitstring)
    Build ID: 31cd63ca6cbb0712b1cf6cc4b740d7a9cd44f8de

Displaying notes found in: .note.ABI-tag
  Owner                 Data size Description
  GNU                   0x00000010NT_GNU_ABI_TAG (ABI version tag)
    OS: Linux, ABI: 3.2.0
Attribute Section: aeabi
File Attributes
  Tag_CPU_name: "6"
  Tag_CPU_arch: v6
  Tag_ARM_ISA_use: Yes
  Tag_THUMB_ISA_use: Thumb-1
  Tag_FP_arch: VFPv2
  Tag_ABI_PCS_wchar_t: 4
  Tag_ABI_FP_rounding: Needed
  Tag_ABI_FP_denormal: Needed
  Tag_ABI_FP_exceptions: Needed
  Tag_ABI_FP_number_model: IEEE 754
  Tag_ABI_align_needed: 8-byte
  Tag_ABI_align_preserved: 8-byte, except leaf SP
  Tag_ABI_enum_size: int
  Tag_ABI_VFP_args: VFP registers
  Tag_CPU_unaligned_access: v6
```

```
No version information found in this file.
Attribute Section: aeabi
File Attributes
  Tag_CPU_name: "7-A"
  Tag_CPU_arch: v7
  Tag_CPU_arch_profile: Application
  Tag_ARM_ISA_use: Yes
  Tag_THUMB_ISA_use: Thumb-2
  Tag_FP_arch: VFPv3-D16
  Tag_ABI_PCS_wchar_t: 4
  Tag_ABI_FP_denormal: Needed
  Tag_ABI_FP_exceptions: Needed
  Tag_ABI_FP_number_model: IEEE 754
  Tag_ABI_align_needed: 8-byte
  Tag_ABI_enum_size: int
  Tag_ABI_VFP_args: VFP registers
  Tag_CPU_unaligned_access: v6
```

Summary so far

```
$ ls -l /lib/ld-linux-armhf.so.3

lrwxrwxrwx 1 root root 30 Oct 18  2022 /lib/ld-linux-armhf.so.3 -> arm-linux-gnueabihf/ld-2.31.so

ls -l /lib/arm-linux-gnueabihf/ld-2.31.so
-rwxr-xr-x 1 root root 146888 Oct 18  2022 /lib/arm-linux-gnueabihf/ld-2.31.so
```

About 150K

```
ls -l /lib/arm-linux-gnueabihf/libc.so.6
lrwxrwxrwx 1 root root 12 Oct 18  2022 /lib/arm-linux-gnueabihf/libc.so.6 -> libc-2.31.so

$ ls -l /lib/arm-linux-gnueabihf/libc-2.31.so
-rwxr-xr-x 1 root root 1319784 Oct 18  2022 /lib/arm-linux-gnueabihf/libc-2.31.so
```

About 1.4M

Adding RPi files to custom embedded system

```
# mkdir /lib/arm-linux-gnueabihf
# cp /media/mmcblk0p1/ld-2.31.so /lib/arm-linux-gnueabihf/.
# ln -s /lib/arm-linux-gnueabihf/ld-2.31.so /lib/ld-linux-armhf.so.3
# ./hello-rpi
./hello-rpi: error while loading shared libraries: libc.so.6: cannot open shared object file: No such file or directory
#
```

```
# cp /media/mmcblk0p1/libc-2.31.so /lib/arm-linux-gnueabihf/.
# ln -s /lib/arm-linux-gnueabihf/libc-2.31.so /lib/arm-linux-gnueabihf/libc.so.6

# ./hello-rpi
Hello World
```

It works!

Summary

- We have gained experience with Embedded Linux Toolchains
  - Host Compiler: gcc
  - Target Compiler (example): armv7l-gnulinuxeabi-gcc
- Typical Workflow
  - Build on Host with Cross-Development Toolchain
  - Copy code to Target to Run