LINUX 交叉编译器的常用调试命令总结：nm,addr2line,objdump,readelf,objcopy,gdb

一、符号列表输出命令：nm

- **命令格式：nm [options] [objfile...]** ---- list symbols from object files

```
options:  [-A|-o|--print-file-name] [-a|--debug-syms]
          [-B|--format=bsd] [-C|--demangle[=style]]
          [-D|--dynamic] [-fformat|--format=format]
          [-g|--extern-only] [-h|--help]
          [-l|--line-numbers] [--inlines]
          [-n|-v|--numeric-sort]
          [-P|--portability] [-p|--no-sort]
          [-r|--reverse-sort] [-S|--print-size]
          [-s|--print-armap] [-t radix|--radix=radix]
          [-u|--undefined-only] [-V|--version]
          [-X 32_64] [--defined-only] [--no-demangle]
          [--plugin name]
          [--no-recurse-limit|--recurse-limit]]
          [--size-sort] [--special-syms]
          [--synthetic] [--with-symbol-versions] [--target=bfdname]
```

- **典型用法：**

```
# arm-oe-linux-gnueabi-nm  -n  test_nw.o  test_api_all
test_nw.o:
         U QL_MCM_NW_AddRxMsgHandler
         U QL_MCM_NW_Client_Deinit
....
test_api_all:
         U _IO_getc
         w _ITM_deregisterTMCloneTable
         w _ITM_registerTMCloneTable
         w _Jv_RegisterClasses
....
```

二、地址转源码行命令：addr2line

- **命令格式：addr2line [options] [addr addr ...]** --- convert addresses into file names and line numbers

```
options:   [-a|--addresses]
           [-b bfdname|--target=bfdname]
           [-C|--demangle[=style]]
           [-r|--no-recurse-limit]
           [-R|--recurse-limit]
           [-e filename|--exe=filename]
           [-f|--functions] [-s|--basename]
           [-i|--inlines]
           [-p|--pretty-print]
           [-j|--section=name]
           [-H|--help] [-V|--version]
```

- **典型用法：**

```
# arm-oe-linux-gnueabi-addr2line -f -e test_api_all 0x00018b20
ql_mcm_voice_set_ecall_config
/home/q/lib_src/common_api/ql_mcm_voice.c:1681

#  arm-oe-linux-gnueabi-addr2line -fs -e test_api_all 0x00018b20
ql_mcm_voice_set_ecall_config
ql_mcm_voice.c:1681

# arm-oe-linux-gnueabi-addr2line -s -e test_api_all 0x00018b20
ql_mcm_voice.c:1681
```

三、反汇编命令：objdump

- **命令格式：objdump [options] objfile...** --- display information from object files

```
options:  [-a|--archive-headers]
          [-b bfdname|--target=bfdname]
          [-C|--demangle[=style] ]
          [-d|--disassemble[=symbol]]
          [-D|--disassemble-all]
          [-z|--disassemble-zeroes]
          [-EB|-EL|--endian={big | little }]
          [-f|--file-headers]
          [-F|--file-offsets]
          [--file-start-context]
```

```
        [-g|--debugging]
        [-e|--debugging-tags]
        [-h|--section-headers|--headers]
        [-i|--info]
        [-j section|--section=section]
        [-l|--line-numbers]
        [-S|--source]
        [--source-comment[=text]]
        [-m machine|--architecture=machine]
        [-M options|--disassembler-options=options]
        [-p|--private-headers]
        [-P options|--private=options]
        [-r|--reloc]
        [-R|--dynamic-reloc]
        [-s|--full-contents]
        [-W[lLiaprmfFsoRtUuTgAckK]|
--dwarf[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,=frames-interp,=str,=loc,=Ranges,=pubtypes,=trace_info,=trace_abbrev,=trace_aranges,=gdb_index,=addr,=cu_index,=links,=follow-links]]
        [--ctf=section]
        [-G|--stabs]
        [-t|--syms]
        [-T|--dynamic-syms]
        [-x|--all-headers]
        [-w|--wide]
        [--start-address=address]
        [--stop-address=address]
        [--prefix-addresses]
        [--[no-]show-raw-insn]
        [--adjust-vma=offset]
        [--dwarf-depth=n]
        [--dwarf-start=n]
        [--ctf-parent=section]
        [--ctf-symbols=section]
        [--ctf-strings=section]
        [--no-recurse-limit|--recurse-limit]
        [--special-syms]
        [--prefix=prefix]
        [--prefix-strip=level]
        [--insn-width=width]
        [-V|--version]
        [-H|--help]
```

- **典型用法：**

```
# arm-oe-linux-gnueabi-objdump  -S -D test_api_all
```

---把程序转成汇编命令输出

```
test_api_all:       file format elf32-littlearm
Disassembly of section .interp:
00008174 <.interp>:
    8174:    62696c2f      rsbvs     r6, r9, #12032     ; 0x2f00
    8178:    2d646c2f      stclcs    12, cr6, [r4, #-188]!     ; 0xffffff44
    817c:    756e696c      strbvc     r6, [lr, #-2412]!     ; 0xfffff694
.....
```

```
# arm-oe-linux-gnueabi-objdump -S -D -j .init test_api_all
```

---与前者的差别在于只对程序段 .init 进行反汇编

```
test_api_all:        file format elf32-littlearm
Disassembly of section .init:
00008dd4 <_init>:
        8dd4:     e92d4008      push      {r3, lr}
        8dd8:     eb0001c0      bl    94e0 <call_weak_fn>
        8ddc:     e8bd8008      pop {r3, pc}
```

```
# arm-oe-linux-gnueabi-objdump -h   test_api_all
```

---显示程序所有段及其大小等信息

```
test_api_all:        file format elf32-littlearm
Sections:
Idx Name                  Size          VMA           LMA           File off    Algn
    0 .interp           00000013   00008174   00008174   00000174    2**0
                              CONTENTS, ALLOC, LOAD, READONLY, DATA
    1 .note.ABI-tag 00000020   00008188   00008188   00000188    2**2
                              CONTENTS, ALLOC, LOAD, READONLY, DATA
    2 .note.gnu.build-id 00000024    000081a8    000081a8    000001a8    2**2
….
```

```
# arm-oe-linux-gnueabi-objdump -x test_api_all
```

---与前者比较，将包含所有段的信息以及程序入口地址等细节信息

```
test_api_all:        file format elf32-littlearm
test_api_all
architecture: arm, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x000094a4
Program Header:
        PHDR off      0x00000034 vaddr 0x00008034 paddr 0x00008034 align 2**2
                 filesz 0x00000140 memsz 0x00000140 flags r--
    INTERP off       0x00000174 vaddr 0x00008174 paddr 0x00008174 align 2**0
                 filesz 0x00000013 memsz 0x00000013 flags r--
    LOAD off    0x00000000 vaddr 0x00008000 paddr 0x00008000 align 2**12
         filesz 0x0001f28c memsz 0x0001f28c flags r-x
........
```

四、ELF 文件解析命令：readelf

- **命令格式：readelf [options] elffile …**　---- display information about ELF files

```
 options:  [-a|--all]
           [-h|--file-header]
           [-l|--program-headers|--segments]
           [-S|--section-headers|--sections]
           [-g|--section-groups]
           [-t|--section-details]
           [-e|--headers]
           [-s|--syms|--symbols]
           [--dyn-syms]
           [-n|--notes]
           [-r|--relocs]
           [-u|--unwind]
           [-d|--dynamic]
           [-V|--version-info]
           [-A|--arch-specific]
           [-D|--use-dynamic]
           [-x <number or name>|--hex-dump=<number or name>]
           [-p <number or name>|--string-dump=<number or name>]
           [-R <number or name>|--relocated-dump=<number or name>]
           [-z|--decompress]
           [-c|--archive-index]
           [-w[lLiaprmfFsoRtUuTgAckK]|
            --debug-
dump[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,=frames-interp,=str,=loc,=Ranges,=pubtypes,=trace_info,=trace_abbrev,=trace_aranges,=gdb_index,
=addr,=cu_index,=links,=follow-links]]
           [--dwarf-depth=n]
           [--dwarf-start=n]
           [--ctf=section]
           [--ctf-parent=section]
           [--ctf-symbols=section]
           [--ctf-strings=section]
           [-I|--histogram]
           [-v|--version]
           [-W|--wide]
```

```
                   [-H|--help]
```

- **典型用法：**

```
# arm-oe-linux-gnueabi-readelf -d test_api_all
```

---查看共享库的依赖库（NEEDED）和搜索名（SONAME）

```
Dynamic section at offset 0x1fd8c contains 32 entries:
  Tag        Type                         Name/Value
 0x00000003 (PLTGOT)                     0x28f04
 0x00000002 (PLTRELSZ)                   480 (bytes)
 0x00000017 (JMPREL)                     0x8bf4
 0x00000014 (PLTREL)                     REL
 0x00000011 (REL)                        0x8bcc
 0x00000012 (RELSZ)                      40 (bytes)
 0x00000013 (RELENT)                     8 (bytes)
 0x00000015 (DEBUG)                      0x0
 0x00000006 (SYMTAB)                     0x81cc
 0x0000000b (SYMENT)                     16 (bytes)
 0x00000005 (STRTAB)                     0x865c
 0x0000000a (STRSZ)                      1064 (bytes)
 0x6ffffef5 (GNU_HASH)                   0x8a84
 0x00000001 (NEEDED)                     Shared library: [libql_sys_log.so.1]
 0x00000001 (NEEDED)                     Shared library: [libpthread.so.0]
 0x00000001 (NEEDED)                     Shared library: [libqmiservices.so.1]
 0x00000001 (NEEDED)                     Shared library: [libqmi_cci.so.1]
 0x00000001 (NEEDED)                     Shared library: [libmcm.so.0]
```

```
# arm-oe-linux-gnueabi-readelf -s test_api_all
```

---列出 ELF 文件的所有符号表

```
Symbol table '.dynsym' contains 73 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000     0 FUNC    GLOBAL DEFAULT  UND strncpy@GLIBC_2.4 (2)
     2: 00000000     0 FUNC    GLOBAL DEFAULT  UND abort@GLIBC_2.4 (2)
     3: 00000000     0 FUNC    GLOBAL DEFAULT  UND malloc@GLIBC_2.4 (2)
     4: 00000000     0 FUNC    GLOBAL DEFAULT  UND strlen@GLIBC_2.4 (2)
     5: 00000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.4
(2)
     6: 00000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
……
```

```
# arm-oe-linux-gnueabi-readelf -h test_api_all
```

---读取 ELF 文件的头部结构信息

```
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:               0x94a4
  Start of program headers:          52 (bytes into file)
  Start of section headers:          519328 (bytes into file)
  Flags:                             0x5000000, Version5 EABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         10
  Size of section headers:           40 (bytes)
  Number of section headers:         43
  Section header string table index: 42
```

# arm-oe-linux-gnueabi-readelf -e test_api_all
---“-e”等效于”–hlS” 读取 ELF 文件头部信息，以及程序、段的头部信息

```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
......
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        00008174 000174 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            00008188 000188 000020 00   A  0   0  4
......
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x00008034 0x00008034 0x00140 0x00140 R   0x4
  INTERP         0x000174 0x00008174 0x00008174 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.3]
  LOAD           0x000000 0x00008000 0x00008000 0x1f28c 0x1f28c R E 0x1000
  LOAD           0x01f690 0x00028690 0x00028690 0x00e44 0x03f44 RW  0x1000
......
Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .note.gnu.build-id .dynsym .dynstr .gnu.hash .gnu.version .gnu.version_r .rel.dyn .rel.plt .init .plt .text .fini .rod
```

```
ata .ARM.extab .ARM.exidx .eh_frame .eh_frame_hdr
   03     .data.rel.ro.local .init_array .fini_array .jcr .dynamic .got .data .bss
   04     .dynamic
   05     .note.ABI-tag .note.gnu.build-id
```

五、内容复制和格式转换命令：objcopy

- **命令格式：objcopy [options] infile [outfile]**   ---- copy and translate object files

```
options:  [-F bfdname|--target=bfdname]
            [-I bfdname|--input-target=bfdname]
            [-O bfdname|--output-target=bfdname]
            [-B bfdarch|--binary-architecture=bfdarch]
            [-S|--strip-all]
            [-g|--strip-debug]
            [--strip-unneeded]
            [-K symbolname|--keep-symbol=symbolname]
            [-N symbolname|--strip-symbol=symbolname]
            [--strip-unneeded-symbol=symbolname]
            [-G symbolname|--keep-global-symbol=symbolname]
            [--localize-hidden]
            [-L symbolname|--localize-symbol=symbolname]
            [--globalize-symbol=symbolname]
            [--globalize-symbols=filename]
            [-W symbolname|--weaken-symbol=symbolname]
            [-w|--wildcard]
            [-x|--discard-all]
            [-X|--discard-locals]
            [-b byte|--byte=byte]
            [-i [breadth]|--interleave[=breadth]]
            [--interleave-width=width]
            [-j sectionpattern|--only-section=sectionpattern]
            [-R sectionpattern|--remove-section=sectionpattern]
            [--remove-relocations=sectionpattern]
            [-p|--preserve-dates]
            [-D|--enable-deterministic-archives]
            [-U|--disable-deterministic-archives]
            [--debugging]
            [--gap-fill=val]
            [--pad-to=address]
            [--set-start=val]
            [--adjust-start=incr]
            [--change-addresses=incr]
            [--change-section-address sectionpattern{=,+,-}val]
```

```
[--change-section-lma sectionpattern{=,+,-}val]
[--change-section-vma sectionpattern{=,+,-}val]
[--change-warnings] [--no-change-warnings]
[--set-section-flags sectionpattern=flags]
[--set-section-alignment sectionpattern=align]
[--add-section sectionname=filename]
[--dump-section sectionname=filename]
[--update-section sectionname=filename]
[--rename-section oldname=newname[,flags]]
[--long-section-names {enable,disable,keep}]
[--change-leading-char] [--remove-leading-char]
[--reverse-bytes=num]
[--srec-len=ival] [--srec-forceS3]
[--redefine-sym old=new]
[--redefine-syms=filename]
[--weaken]
[--keep-symbols=filename]
[--strip-symbols=filename]
[--strip-unneeded-symbols=filename]
[--keep-global-symbols=filename]
[--localize-symbols=filename]
[--weaken-symbols=filename]
[--add-symbol name=[section:]value[,flags]]
[--alt-machine-code=index]
[--prefix-symbols=string]
[--prefix-sections=string]
[--prefix-alloc-sections=string]
[--add-gnu-debuglink=path-to-file]
[--keep-file-symbols]
[--only-keep-debug]
[--strip-dwo]
[--extract-dwo]
[--extract-symbol]
[--writable-text]
[--readonly-text]
[--pure]
[--impure]
[--file-alignment=num]
[--heap=size]
[--image-base=address]
[--section-alignment=num]
[--stack=size]
[--subsystem=which:major.minor]
[--compress-debug-sections]
```

```
                [--decompress-debug-sections]
                [--elf-stt-common=val]
                [--merge-notes]
                [--no-merge-notes]
                [--verilog-data-width=val]
                [-v|--verbose]
                [-V|--version]
                [--help] [--info]
```

- **典型用法：**

```
# arm-oe-linux-gnueabi-objcopy -O srec   test_api_all   test_api_all.srec
```

---- 将 ELF 可执行文件转换成 s-record 格式的文件

```
# arm-oe-linux-gnueabi-objcopy -O binary -R .note -R .comment vmlinux zImage
```

---- 删除 vmlinux 的.note /.comment 段并指定以 rawbinary 格式生成 zImage 文件

```
# arm-oe-linux-gnueabi-objcopy --only-keep-debug   vmlinux   debuginfo
```

---- 提取 ELF 文件中的 debug 信息到 debuginfo

```
# arm-oe-linux-gnueabi-objcopy   --strip-debug   vmlinux   vmlinux.withoutdebug
```

---- 生成不含 debug 信息的可执行文件 vmlinux.withoutdebug.

六、GNU 调试命令： gdb

- **命令格式：gdb [options]   app [app_PID|coredumpfile]**   ---- debugger

```
 options:  [-help] [-nh] [-nx] [-q] [-batch] [-cd=dir] [-f] [-b bps]
                [-tty=dev] [-s symfile] [-e prog] [-se prog] [-c core] [-p procID]
                [-x cmds] [-d dir]
```

- **典型用法：**

```
# arm-oe-linux-gnueabi-gdb app
```

--- app 是可执行程序

```
root@ubuntu:~/# arm-oe-linux-gnueabi-gdb build/vmlinux
GNU gdb (GDB) 7.9.1
...
Reading symbols from build/vmlinux...done.
(gdb) list *(mdm9607_init+0x10)
0xc0a5d830 is in coredump_filter_setup (../kernel/fork.c:545).
540  static unsigned long default_dump_filter = MMF_DUMP_FILTER_DEFAULT;
541
542  static int __init coredump_filter_setup(char *s)
543  {
```

```
544         default_dump_filter =
545                 (simple_strtoul(s, NULL, 0) << MMF_DUMP_FILTER_SHIFT) &
546                 MMF_DUMP_FILTER_MASK;
547         return 1;
548 }
549
(gdb) list *0xc0a5d830
0xc0a5d830 is in coredump_filter_setup (../kernel/fork.c:545).
540 static unsigned long default_dump_filter = MMF_DUMP_FILTER_DEFAULT;
541
542 static int __init coredump_filter_setup(char *s)
543 {
544         default_dump_filter =
545                 (simple_strtoul(s, NULL, 0) << MMF_DUMP_FILTER_SHIFT) &
546                 MMF_DUMP_FILTER_MASK;
547         return 1;
548 }
549
(gdb) disassemble mdm9607_init
Dump of assembler code for function mdm9607_init:
   0xc0a5d820 <+0>: andeq    r0, r0, r0
   0xc0a5d824 <+4>: andeq    r0, r0, r0
End of assembler dump.
```

# arm-oe-linux-gnueabi-gdb app   coredump

--- coredump 是程序异常后生成的 dump 文件，一般用于异常发生后的复查调试

```
~/# arm-oe-linux-gnueabi-gdb example_pthread core.example_pthread.3196.11.4228
GNU gdb (GDB) 7.9.1
...
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?
Core was generated by `./example_pthread'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0   0x00008684 in main ()
(gdb) bt
#0   0x00008684 in main ()
(gdb) info frame
Stack level 0, frame at 0xbec62c60:
 pc = 0x8684 in main; saved pc = 0x423381b4
 Arglist at 0xbec62c48, args:
 Locals at 0xbec62c48, Previous frame's sp is 0xbec62c60
 Saved registers:
  r4 at 0xbec62c58, lr at 0xbec62c5c
```

```
# arm-oe-linux-gnueabi-gdb app   procPID
```

--- procPID 是 app 运行时的进程 PID，用于程序运行时的在线调试

```
/data # ps | grep pthread
 3247 root         0:00 ./example_pthread
 3253 root         0:00 {grep} /bin/busybox /bin/grep pthread
/data # gdb example_pthread   3247
GNU gdb (GDB) 7.9.1
Copyright (C) 2015 Free Software Foundation, Inc.
...
---Type <return> to continue, or q <return> to quit---
0x424877b0 in pthread_join () from /lib/libpthread.so.0
(gdb) info frame
Stack level 0, frame at 0xbee6cc10:
 pc = 0x424877b0 in pthread_join; saved pc = 0x0
 called by frame at 0xbee6cc10
 Arglist at 0xbee6cc10, args:
 Locals at 0xbee6cc10, Previous frame&apos;s sp is 0xbee6cc10
(gdb) bt
#0   0x424877b0 in pthread_join () from /lib/libpthread.so.0
#1   0x00000000 in ?? ()
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
(gdb) info functions
All defined functions:
Non-debugging symbols:
0x000084e0   _init
0x00008500   __libc_start_main@plt
0x0000850c   abort@plt
0x00008518   __gmon_start__@plt
0x00008524   puts@plt
0x00008530   getpid@plt
0x0000853c   pthread_self@plt
0x00008548   printf@plt
0x00008554   sleep@plt
...
```