Using objdump to disassemble an object...

PeterChan

Using objdump to disassemble an object file Dec 23, 2014 3:21 AM

The cross compiler toolchain has an "objdump" utility that can disassemble an object file readily. This page shows you how to disassemble an object file using this utility and its application.

In Android, the cross compiler toolchain is installed to prebuilts/gcc/linux-x86/arm. In LTIB, the cross compiler toolchain is installed to /opt/freescale/usr. Depends on which BSP releases had been installed in your Linux host, several cross compiler toolchains may have installed. Please ensure you are always use the appropriate toolchain for the BSP.

For LTIB release L3.0.35_4.1.0, its cross compile toolchain is located at /opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin. Let's take this toolchain as an example to show the usage of "objdump".

To disassemble an object file to assembly, simply run the command below and you will get the assembly code.

```
/opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/arm-fsl-linux-gnueabi-objdump -S <object file>
```

Because no debug info is added into the object file when compiled, the output is just assembly code when function names and labels only. To get an Intermix source code with disassemble code, it is necessary to add the "-g" option when compile.

The "objdump" is useful to find out where kernel crashes from error log. For example, if we got this error and want to find out where the kernel hangs.

```
Unable to handle kernel NULL pointer dereference at virtual address 00000030

pgd = c0004000

[00000030] *pgd=00000000

Internal error: Oops: 5 [#1] PREEMPT

Modules linked in:

CPU: 0    Not tainted  (3.0.35 #6)

PC is at gckKERNEL_QueryProcessDB+0x24/0x1c4

LR is at viv_gpu_resmem_query+0x28/0x4c

pc : [<c03713b4>]    lr : [<c0367d00>]    psr: a0070113

sp : da397e38  ip : 00000000  fp : c0831ce8

r10: 00000001  r9 : d2a61ce0  r8 : 000009f4

r7 : 00000000  r6 : da397e68  r5 : d2a61c20  r4 : 00000000

r3 : 00000001  r2 : 00000000  r1 : 000009f4  r0 : 00000000
```

First we now that the crash occurred in gckKERNEL_QueryProcessDB() at address offset 0x24 and its caller is viv_gpu_resmem_query() at address offset 0x28. Execute a text search and we know its source file is drivers/mxc/gpu-viv/hal/kernel/gc_hal_kernel_db.c. The compile option for gc_hal_kernel_db.c is provided by drivers/mxc/gpu-viv/Kbuild. So, we enable "-g" compile option here and rebuild the kernel.

```
diff --git a/drivers/mxc/gpu-viv/Kbuild b/drivers/mxc/gpu-viv/Kbuild

index bc5ec02..773d0a0 100644

--- a/drivers/mxc/gpu-viv/Kbuild
```

```
+++ b/drivers/mxc/gpu-viv/Kbuild

@@ -115,9 +115,9 @@ EXTRA_CFLAGS += -DFLAREON

endif



ifeq ($(DEBUG), 1)

-EXTRA_CFLAGS += -DDBG=1 -DDEBUG -D_DEBUG

+EXTRA_CFLAGS += -DDBG=1 -DDEBUG -D_DEBUG -g

else

-EXTRA_CFLAGS += -DDBG=0

+EXTRA_CFLAGS += -DDBG=0 -g

endif



ifeq ($(NO_DMA_COHERENT), 1)
```

Then we execute /opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/arm-fsl-linux-gnueabi-objdump -S ./drivers/mxc/gpu-viv/hal/kernel/gc_hal_kernel_db.o > gc_hal_kernel_db.asm

Inspect the gckKERNEL_QueryProcessDB in the disassemble output:

```
00000c68 <gckKERNEL_QueryProcessDB>:

    IN gctUINT32 ProcessID,

    IN gctBOOL LastProcessID,
    IN gceDATABASE_TYPE Type,

    OUT gcuDATABASE_INFO * Info

    )

{

c68:   e92d45f0    push    {r4, r5, r6, r7, r8, sl, lr}

c6c:   e24dd00c    sub sp, sp, #12

c70:   e1a04000    mov r4, r0

c74:   e1a06001    mov r6, r1

c78:   e59d8028    ldr r8, [sp, #40]   ; 0x28

c7c:   e1a05002    mov r5, r2

c80:   e1a07003    mov r7, r3

    gcmkHEADER_ARG("Kernel=0x%x ProcessID=%d Type=%d Info=0x%x",

                   Kernel, ProcessID, Type, Info);


    /* Verify the arguments. */

    gcmkVERIFY_OBJECT(Kernel, gcvOBJ_KERNEL);

    gcmkVERIFY_ARGUMENT(Info != gcvNULL);

c84:   e3580000    cmp r8, #0

c88:   03e0a000    mvneq   sl, #0

c8c:   0a000030    beq d54 <gckKERNEL_QueryProcessDB+0xec>


    /* Acquire the database mutex. */
    gcmkONERROR(

c90:   e5903030    ldr r3, [r0, #48]   ; 0x30

c94:   e3e02000    mvn r2, #0

c98:   e5900004    ldr r0, [r0, #4]

c9c:   e5931040    ldr r1, [r3, #64]   ; 0x40

ca0:   ebfffffe    bl  0 <gckOS_AcquireMutex>

ca4:   e250a000    subs    sl, r0, #0

ca8:   ba000029    blt d54 <gckKERNEL_QueryProcessDB+0xec>

        gckOS_AcquireMutex(Kernel->os, Kernel->db->dbMutex, gcvINFINITE));

    acquired = gcvTRUE;
```

The error log tells us the error occurs at 0xc68 + 0x24. From the disassemble output, we found that at address 0xc90, the error is caused by r0=0 which is corresponding to the first parameter gckKERNEL Kernel = NULL.

Tags: debug, objdump, disassemble