Linux userspace空间进程挂掉调试

作者： 匡道珍

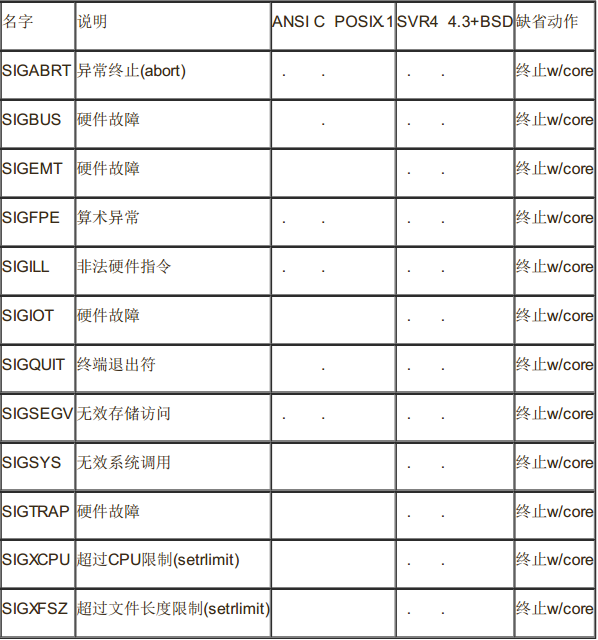
最近在做项目时，总是遇到进程运行着就挂掉了，而且出现的概率非常之低，出现一次之后就再也无法复现。每次遇到这样的问题，每次的解决方法都是进行大量的测试加大复现概率和代码走读，到最后都是不了了之。解决这类问题有几个难点：

1、由于系统在运行中出现某个进程挂掉，根本不知道其触发条件是什么；

2、由于是系统的常驻进程，代码量比较大，并不是简单的进行代码走读就可以查出问题说在，虽然有coverity等相关的代码检测工具，但是由于其是静态检测，有些问题还是无法检测出来；

3、出现的概率小，即使通过代码走读解决了一些问题也不知道这些问题是否是导致进程挂掉的本质原因；

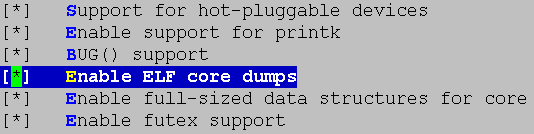
对于解决这些问题，其实linux内核已经提供了一个core dump机制，该机制在linux userspace空间进程挂掉时会转存挂掉进程相关的内存信息并且保存在一个文件中，通过gdb调试该core dump信息就可以确定进程挂在那个函数那一行。当然并不是所有情况进程挂掉都会生产core dump文件，产生core dump文件只有在进程收到以下信号时才会产生。



# 一、开启内核宏和设置相关配置

1、为了产生core dump文件必须开启内核CONFIG\_ELF\_CORE宏，打开make menuconfig查找General setup ---> Configure standard kernel features (for small systems) --->

[\*] Enable ELF core dumps



2、userspace空间需要通过命令ulimit进行相关的配置，并且ulimit命令一般是shell的内建命令，通ulimit –a查看

time(seconds) unlimited

file(blocks) unlimited

data(kb) unlimited

stack(kb) 8192

coredump(blocks) unlimited

memory(kb) unlimited

locked memory(kb) 64

process 466

nofiles 1024

vmemory(kb) unlimited

locks unlimited

coredump(blocks)如果后面数字是0则表示不会生产coredump文件，为了产生core dump文件可以通过ulimit –c unlimited设置生成core dump文件的大小，当设置unlimited时不限制core dump文件大小，否则core dump文件最大不能超过设置的大小。因为在当前用户shell设置ulimit只会对当前shell生效不会对其他用户shell生效，为了在系统起来和对所有shell生效，可以在/etc/profile(没profile文件手动创建)中添加ulimit –c unlimited，这样在系统起来时就会生效。

3、产生core dump文件存放路径，默认情况下core dump文件是在执行目录生产一个core文件，由于所有的core dump文件名字都一样。这样不利于管理core dump文件，于是可以通过设置/proc/sys/kernel/core\_pattern中的值来改变生产core dump文件的路径和文件名，core\_pattern文件参数如下：

%p - insert pid into filename 添加pid

     %u - insert current uid into filename 添加当前uid

     %g - insert current gid into filename 添加当前gid

     %s - insert signal that caused the coredump into the filename 添加导致产生core的信号

     %t - insert UNIX time that the coredump occurred into filename 添加core文件生成时的unix时间

     %h - insert hostname where the coredump happened into filename 添加主机名

     %e - insert coredumping executable name into filename 添加命令名

设置core\_pattern参数echo "/tmp/core-%e-%p-%t-%s" > /proc/sys/kernel/core\_pattern

这样就会在/tmp下面产生相关的core dump文件。

4、当没有该命令时，可以通过系统函数getrlimit、setrlimit设置使其产生core dump文件，但是这样设置只会在当前进程中生效,以下代码可以参考

static void init\_core\_dump()

{

struct rlimit limit;

unsigned char core\_pattern[] = "/tmp/core-%e-%p-%t-%s";

char buf[256] = {0};

memset(&limit, 0x0, sizeof(struct rlimit));

if (0 == getrlimit(RLIMIT\_CORE, &limit)){

printf("%s %d: rlim\_cur = %d, rlim\_max = %d\n", \_\_func\_\_, \_\_LINE\_\_,

limit.rlim\_cur, limit.rlim\_max);

}

memset(&limit, 0x0, sizeof(struct rlimit));

limit.rlim\_cur = -1;

limit.rlim\_max = -1;

if (0 == setrlimit(RLIMIT\_CORE, &limit)){

printf("%s %d: open core dump success\n", \_\_func\_\_, \_\_LINE\_\_);

sprintf(buf, “%s > /proc/sys/kernel/core\_pattern”, core\_pattern);

system("echo %s > /proc/sys/kernel/core\_pattern", core\_pattern);

}else {

perror("setrlimit");

}

memset(&limit, 0x0, sizeof(struct rlimit));

if (0 == getrlimit(RLIMIT\_CORE, &limit)){

printf("%s %d: rlim\_cur = %d, rlim\_max = %d\n", \_\_func\_\_, \_\_LINE\_\_,

limit.rlim\_cur, limit.rlim\_max);

}

}

# 二、编译gdb

由于嵌入式是使用的arm架构，需要编译可以在x86上运行而又可以调试arm架构的gdb，x86自带的gdb调试无法调试arm架构。

1、从<http://www.gnu.org/software/gdb/> 下载 **gdb-7.7.tar.gz**

2、*tar -zxvf gdb-7.7.tar.gz* 进行解压后

3、*cd gdb-7.7/*

4、*./configure --host=i386 --target=arm-brcm-linux-uclibcgnueabi*

*CC=arm-brcm-linux-uclibcgnueabi-gcc*

由于我们编译的gdb需要在x86上运行，需要带上—host=i386，如果在x86上运行部带也可以，执行configure时会从编译环境中获取该值，如果编译的gdb要在arm架构的板子上运行就要加上—host= arm-brcm-linux-uclibcgnueabi，而--target=arm-brcm-linux-uclibcgnueabi则表示编译出来的gdb可以解析arm架构的相关指令。CC是交叉编译工具链中的gcc编译器，编译时可能会出错libtermcap.a无法找到，可以下载termcap-1.3.1.tar.gz通过arm的交叉编译工具链生产libtermcap.a，并将其拷贝到gdb源码下的libtermcap(没该文件夹手动创建)文件下,找到gdb/configure

cygwin\*)

if test -d $srcdir/libtermcap; then

LIBS="../libtermcap/libtermcap.a $LIBS"

ac\_cv\_search\_tgetent="../libtermcap/libtermcap.a"

fi ;;

go32\* | \*djgpp\*)

ac\_cv\_search\_tgetent="none required"

;;

\*mingw32\*)

ac\_cv\_search\_tgetent="none required"

CONFIG\_OBS="$CONFIG\_OBS windows-termcap.o"

;;

esac

并且在其后添加以下语句

ac\_cv\_search\_tgetent="/home/suse/Project/Program/termcap-1.3.1/libtermcap.a"

在执行make就可以编译通过了。

# 三、补充setrlimit函数和getrlimit

通过ulimit设置产生core dump文件，该值只有对在该shell启动的进程才会生效。如果有A、B、C三个进程，而A进程通过shell启动，B、C通过A进程fork出来，则B、C进程无法继承A的ulimit属性，如果B、C需要产生core dump文件，可以通过setrlimit函数进行设置

# 四、实践

## 例题1：

源代码如下：

1 #include <stdio.h>

2 #include <sys/time.h>

3 #include <sys/resource.h>

4 #include <string.h>

5

6

7 int func1()

8 {

9 // char \*ptr = "123";

10 // \*ptr = 0;

11

12 printf("%s %d: --------%s------\n", \_\_func\_\_, \_\_LINE\_\_, 1);

13 }

14

15 int main()

16 {

17 func1();

18 return 0;

19 }

Gdb在调试时需要在编译时加入-g选项，不过由于嵌入式程序资源紧张编译好之后都会执行strip去除相关的符号，即使加入-g编译执行strip之后二进制文件中也不会有符号。因此在程序挂掉并且产生了core dump文件可以对源码进行编译并且加入-g选项，当然这时千万不要执行strip。

可执行文件a.out, core dump文件core-a.out-659-1304208581-11

linux-z13b:Program # arm-brcm-linux-uclibcgnueabi-gdb a.out core-a.out-659-1304208581-11

GNU gdb (GDB) 7.7

Copyright (C) 2014 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law. Type "show copying"

and "show warranty" for details.

This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-brcm-linux-uclibcgnueabi".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online at:

<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".

Type "apropos word" to search for commands related to "word"...

Reading symbols from a.out...done.

[New LWP 659]

warning: Could not load shared library symbols for 2 libraries, e.g. /lib/libc.so.0.

Use the "info sharedlibrary" command to see the complete listing.

Do you need "set solib-search-path" or "set sysroot"?

Core was generated by `./a.out'.

Program terminated with signal SIGSEGV, Segmentation fault.

#0 0x402897a8 in ?? () //由于加载的库有问题函数名显示为??

(gdb) set solib-search-path /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib

Reading symbols from /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/libc.so.0...(no debugging symbols found)...done.

Loaded symbols for /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/libc.so.0

Reading symbols from /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/ld-uClibc.so.0...(no debugging symbols found)...done.

Loaded symbols for /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/ld-uClibc.so.0

(gdb) where

#0 0x402897a8 in strnlen () from /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/libc.so.0

#1 0x40284050 in \_vfprintf\_internal ()

from /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/libc.so.0

#2 0x40283b28 in vfprintf () from /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/libc.so.0

#3 0x4028110c in printf () from /projects/hnd/tools/linux/hndtools-arm-linux-2.6.36-uclibc-4.5.3/lib/libc.so.0

#4 0x00008448 in func1 () at segment\_fault.c:12

#5 0x0000845c in main () at segment\_fault.c:17

(gdb)

说明：

1、由于运行gdb加载的库默认是从x86下面的lib中加载的，而我们需要的库在交叉工具链中，因此需要使用set solib-search-path设置加载库的路径，当然如果出错的地方在我们的程序中不加载库也可以；

2、运行where命令就会出现调用栈信息，最上面是出错的函数；

3、上面程序由于挂掉的地方在系统库函数中，所以挂掉的点是strnlen，其实是由于在func1函数中调用了printf函数传递的参数有问题导致的；

4、由于编译时加载了-g参数会显示相关的行号。

5、根据core dump文件core-a.out-659-1304208581-11(生成core dump文件名需要按照之前讲的方法设置)就知道出错的进程名是a.out，进程id是659，而1304208581表示时间，通过time计算的，11表示是什么原因导致进程挂掉，11号信号为段错误。

## 例题2：

1 #include <stdio.h>

2 #include <sys/time.h>

3 #include <sys/resource.h>

4 #include <string.h>

5

6

7 int func1()

8 {

9 char \*ptr = "123";

10 \*ptr = 0;

11

12 // printf("%s %d: --------%s------\n", \_\_func\_\_, \_\_LINE\_\_, 1);

13 }

14

15 int main()

16 {

17 func1();

18 return 0;

19 }

linux-z13b:Program # /home/suse/Project/Program/arm-gdb/bin/arm-brcm-linux-uclibcgnueabi-gdb a.out core-a.out-1083-1304215052-11

/\*删除了gdb运行时打印的相关信息，信息和上一个例题一样\*/

warning: Could not load shared library symbols for 2 libraries, e.g. /lib/libc.so.0.

Use the "info sharedlibrary" command to see the complete listing.

Do you need "set solib-search-path" or "set sysroot"?

Core was generated by `./a.out'.

Program terminated with signal SIGSEGV, Segmentation fault.

#0 0x00008410 in func1 () at segment\_fault.c:10

10 \*ptr = 0;

(gdb) where

#0 0x00008410 in func1 () at segment\_fault.c:10

#1 0x00008430 in main () at segment\_fault.c:17

(gdb)

说明：

1、由于导致程序挂掉地方在源代码中，因此不需要设置库路径也可以找到栈信息；

2、通过gdb调试可以很清楚的知道在那个函数中出问题了。

## 例题3

下面例题出现的段错误是项目中出现的，出问题的函数如下：

TPI\_RET tpi\_do\_scan(char \*wl\_ifname, int \*len,TPI\_AP\_SCAN\_LIST ap\_list[],int arry\_len)

{

int i = 0,retry\_times = 0;

wds\_ap\_list\_info\_t \*wds\_aplist = NULL;

wl\_scan\_params\_t \*params;

int params\_size = WL\_SCAN\_PARAMS\_FIXED\_SIZE + NUMCHANS \* sizeof(uint16);

int org\_scan\_time = 20, scan\_time = 40;

// TPI\_RET ret =TPI\_RET\_OK;

int ret =TPI\_RET\_OK;

char \*ptr = "1234";

\*ptr = 0;

……..

return ret;

}

linux-z13b:Program # /home/suse/Project/Program/arm-gdb/bin/arm-brcm-linux-uclibcgnueabi-gdb ../5.0\_UGW/85ap\_ugw5.0/develop/prod/httpd/httpd/src/httpd core-httpd-1193-1405605950-11

warning: Could not load shared library symbols for 11 libraries, e.g. /lib/libCfm.so.

Use the "info sharedlibrary" command to see the complete listing.

Do you need "set solib-search-path" or "set sysroot"?

Core was generated by `httpd'.

Program terminated with signal SIGSEGV, Segmentation fault.

#0 0x401d8264 in ?? ()

(gdb) where //没有加载出错库之前的栈信息

#0 0x401d8264 in ?? () //由于该函数在库libtpi中没有加载就显示??

#1 0x0004c730 in formwrlWDS (wp=0x0, path=0x0, query=0x0)

at /home/suse/Project/5.0\_UGW/85ap\_ugw5.0/develop/prod/httpd/ap\_web/cgi/wifi.c:4183

#2 0x0004c730 in formwrlWDS (wp=0x979f8, path=0xbe8d42f0 "wifiWDS", query=0x97e30 "wl\_radio=0&0.908410849282518")

at /home/suse/Project/5.0\_UGW/85ap\_ugw5.0/develop/prod/httpd/ap\_web/cgi/wifi.c:4183

#3 0x00012fd4 in websFormHandler (wp=0x979f8, urlPrefix=0x90128 "/goform", webDir=0x90140 "", arg=0,

url=0x97ee0 "/goform/wifiWDS?wl\_radio=0&0.908410849282518", path=0x97d88 "/goform/wifiWDS",

query=0x97e30 "wl\_radio=0&0.908410849282518") at form.c:74

#4 0x00013e80 in websUrlHandlerRequest (wp=0x979f8) at handler.c:285

#5 0x0002642c in websGetInput (wp=0x979f8, ptext=0xbe8d4c90, pnbytes=0xbe8d4c8c) at webs.c:802

#6 0x00025c0c in websReadEvent (wp=0x979f8) at webs.c:452

#7 0x00025b3c in websSocketEvent (sid=1, mask=2, iwp=621048) at webs.c:406

#8 0x00018844 in socketDoEvent (sp=0x97c08) at sockGen.c:1337

#9 0x0001865c in socketProcess (sid=1) at sockGen.c:1257

#10 0x0002a104 in main (argc=1, argv=0xbe8d4eb4) at main.c:411

(gdb) set solib-search-path /home/suse/Project/5.0\_UGW/85ap\_ugw5.0/develop/cbb/cbb\_tpi/lib/

warning: `/lib/libpthread.so.0': Shared library architecture unknown is not compatible with target architecture arm.

Reading symbols from /home/suse/Project/5.0\_UGW/85ap\_ugw5.0/develop/cbb/cbb\_tpi/lib/libtpi.so...done.

Loaded symbols for /home/suse/Project/5.0\_UGW/85ap\_ugw5.0/develop/cbb/cbb\_tpi/lib/libtpi.so

warning: `/lib/libgcc\_s.so.1': Shared library architecture unknown is not compatible with target architecture arm.

(gdb) where //加载库后的栈信息

#0 0x401d8264 in tpi\_do\_scan (wl\_ifname=0x401b35ac <error: Cannot access memory at address 0x401b35ac>,

len=0xbe8d32a0, ap\_list=0xbe8ce520, arry\_len=128) at wifi/tpi\_wifi.c:1108

#1 0x0004c730 in formwrlWDS (wp=0x979f8, path=0xbe8d42f0 "wifiWDS", query=0x97e30 "wl\_radio=0&0.908410849282518")

at /home/suse/Project/5.0\_UGW/85ap\_ugw5.0/develop/prod/httpd/ap\_web/cgi/wifi.c:4183

#2 0x00012fd4 in websFormHandler (wp=0x979f8, urlPrefix=0x90128 "/goform", webDir=0x90140 "", arg=0,

url=0x97ee0 "/goform/wifiWDS?wl\_radio=0&0.908410849282518", path=0x97d88 "/goform/wifiWDS",

query=0x97e30 "wl\_radio=0&0.908410849282518") at form.c:74

#3 0x00013e80 in websUrlHandlerRequest (wp=0x979f8) at handler.c:285

#4 0x0002642c in websGetInput (wp=0x979f8, ptext=0xbe8d4c90, pnbytes=0xbe8d4c8c) at webs.c:802

#5 0x00025c0c in websReadEvent (wp=0x979f8) at webs.c:452

#6 0x00025b3c in websSocketEvent (sid=1, mask=2, iwp=621048) at webs.c:406

#7 0x00018844 in socketDoEvent (sp=0x97c08) at sockGen.c:1337

#8 0x0001865c in socketProcess (sid=1) at sockGen.c:1257

#9 0x0002a104 in main (argc=1, argv=0xbe8d4eb4) at main.c:411

(gdb)