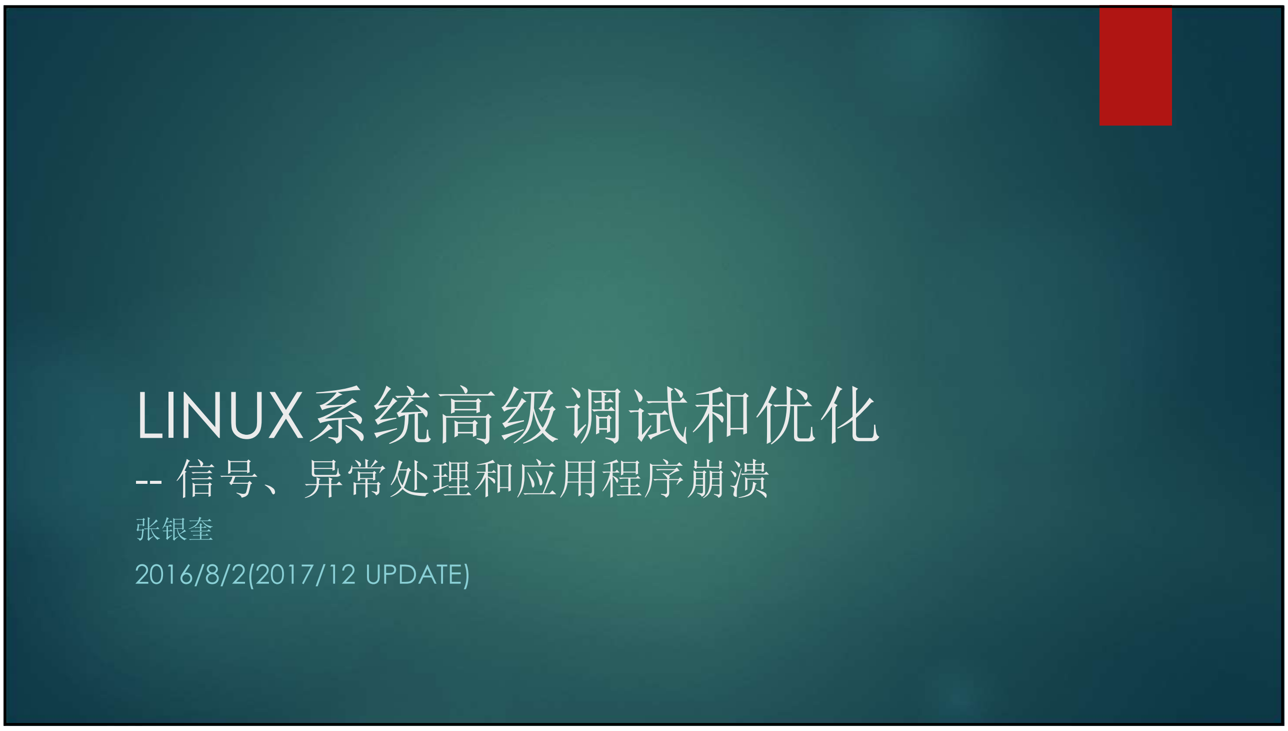
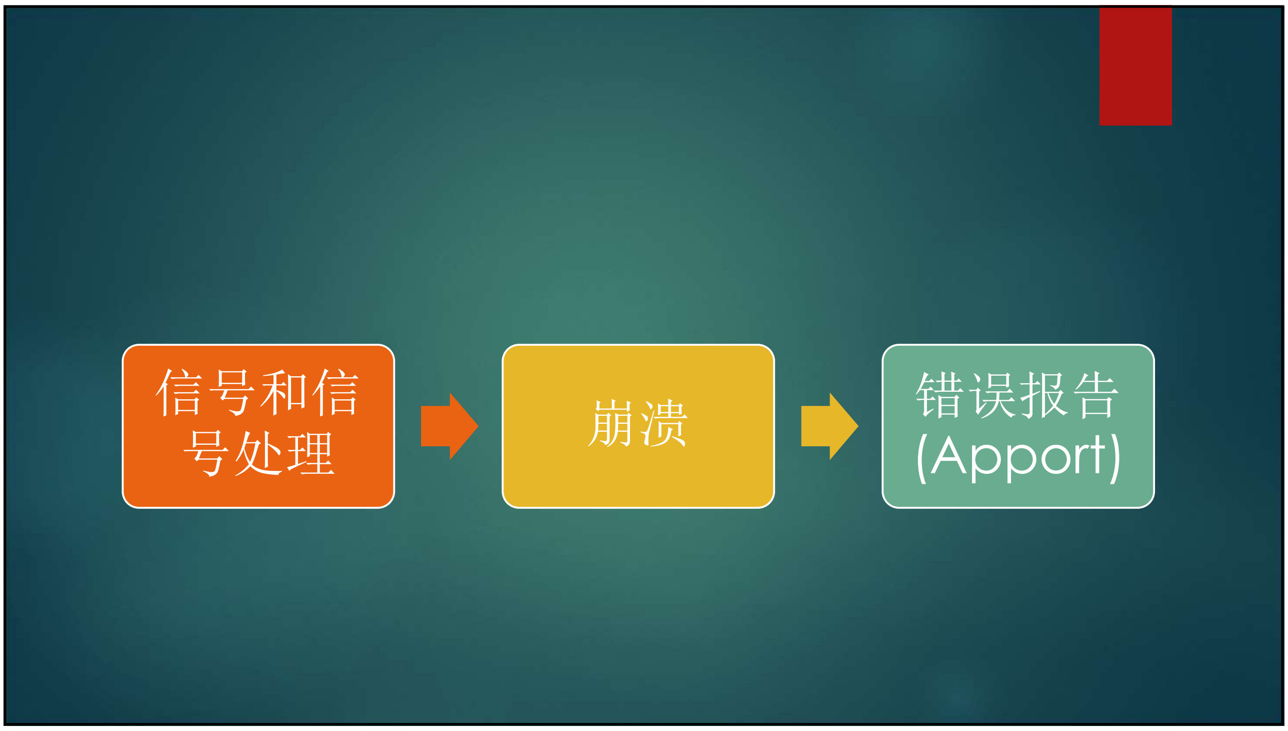


LINUX系统高级调试和优化
-- 信号、异常处理和应用程序崩溃

张银奎

2016/8/2(2017/12 UPDATE)

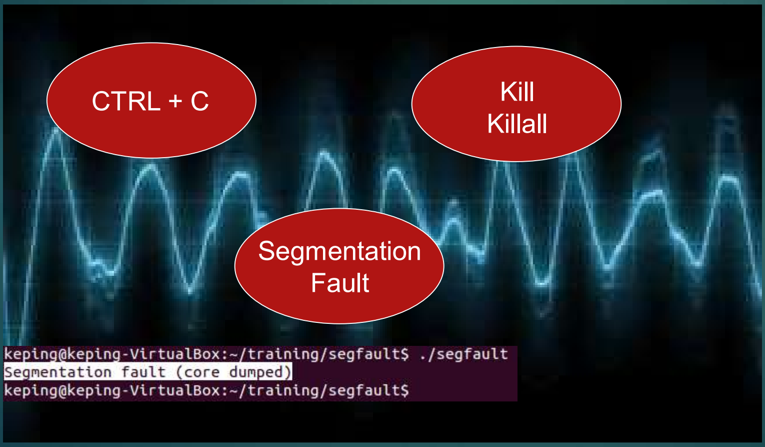
1



2

无处不在的SIGNAL

Linux系统每一秒中都有大量的信号在产生、传递、接收和处理，不管你有没有意识到它的存在

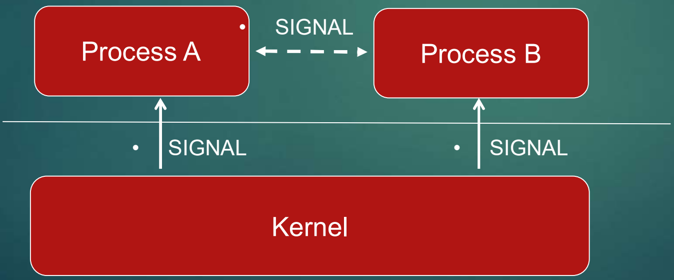


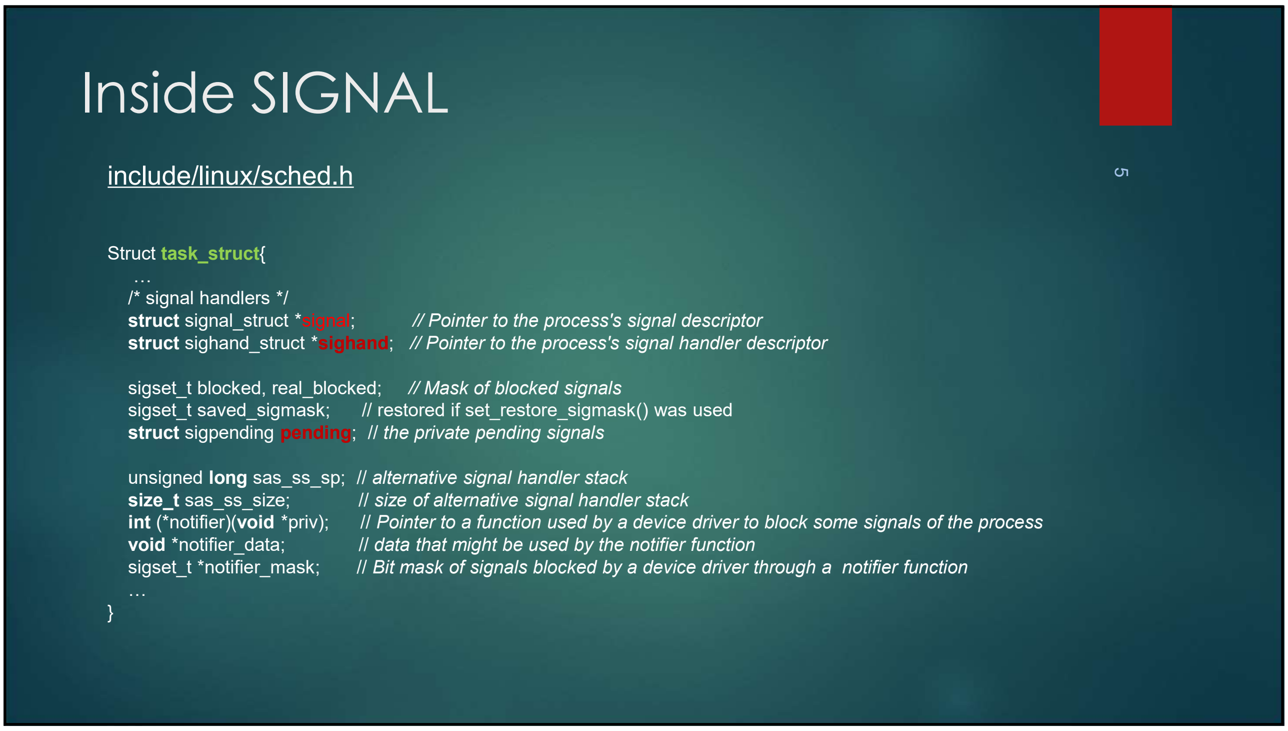
```
keping@keping-VirtualBox:~/training/segfault$ ./segfault
Segmentation fault (core dumped)
keping@keping-VirtualBox:~/training/segfault$
```

从本页开始的信号部分，由王科平为第一届庐山研习班而编写

什么是SIGNAL

- 在软件层次上是对中断机制的一种模拟
- 一种进程间的异步通信机制
- 内核也可以因为内部事件而给进程发送信号，通知进程发生了某个事件 (如著名的segmentation fault)
- 从UNIX继承过来，Linux进行了扩展和改进



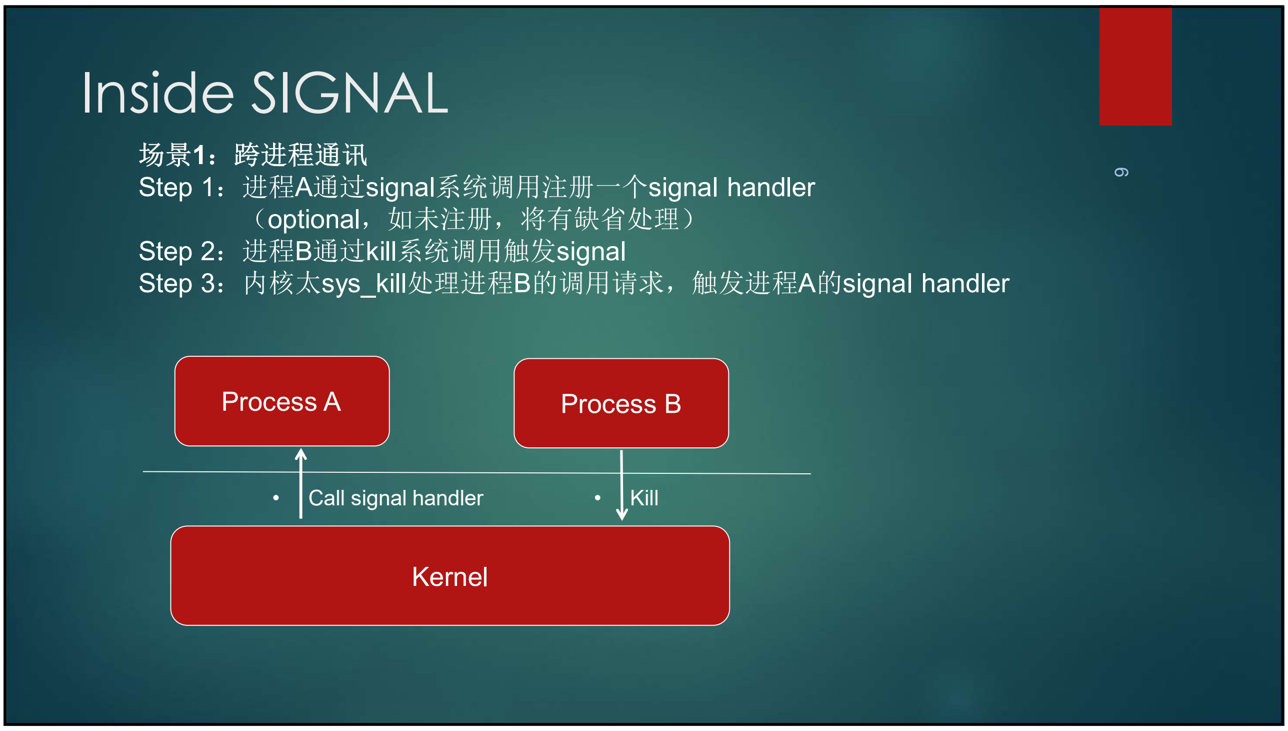


Inside SIGNAL

include/linux/sched.h

```
Struct task_struct{
   …
   /* signal handlers */
   struct signal_struct *signal;          // Pointer to the process's signal descriptor
   struct sighand_struct *sighand;   // Pointer to the process's signal handler descriptor

   sigset_t blocked, real_blocked;     // Mask of blocked signals
   sigset_t saved_sigmask;      // restored if set_restore_sigmask() was used
   struct sigpending pending;  // the private pending signals

   unsigned long sas_ss_sp;  // alternative signal handler stack
   size_t sas_ss_size;            // size of alternative signal handler stack
   int (*notifier)(void *priv);      // Pointer to a function used by a device driver to block some signals of the process
   void *notifier_data;            // data that might be used by the notifier function
   sigset_t *notifier_mask;       // Bit mask of signals blocked by a device driver through a  notifier function
   …
}
```

5



Inside SIGNAL

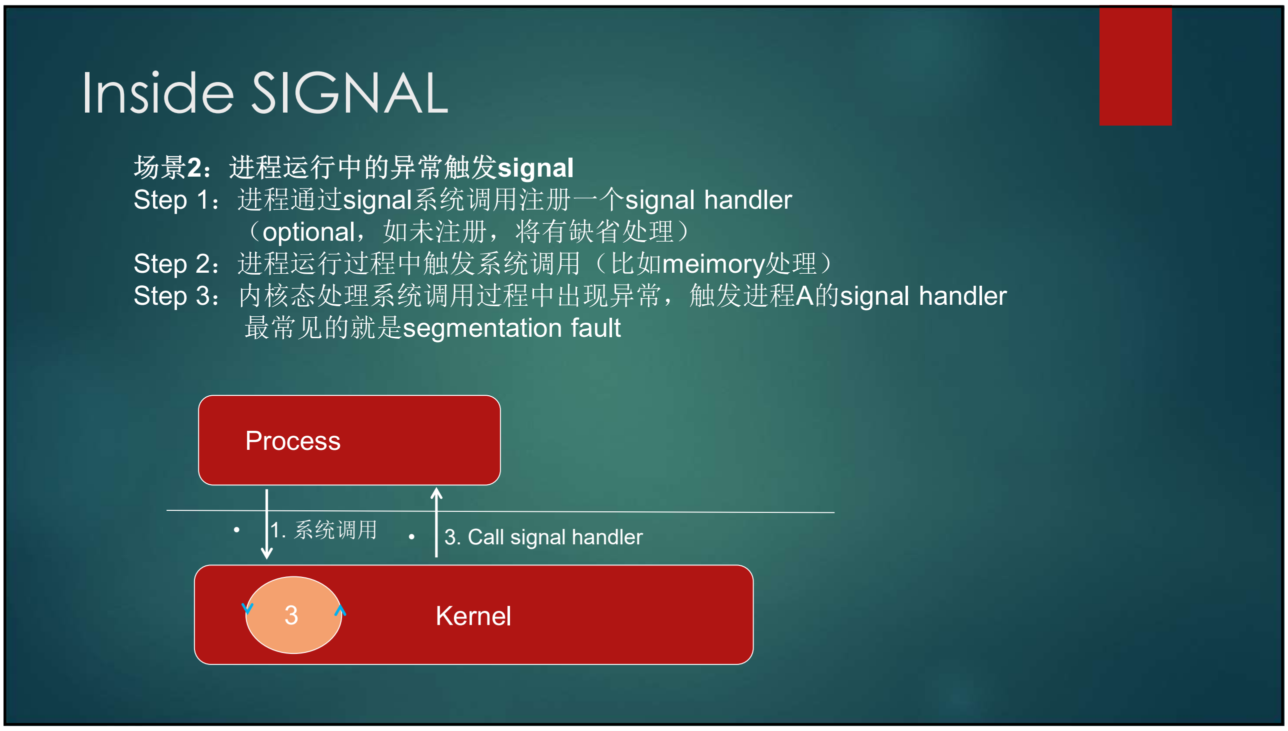
场景1：跨进程通讯

Step 1：进程A通过signal系统调用注册一个signal handler
（optional，如未注册，将有缺省处理）

Step 2：进程B通过kill系统调用触发signal

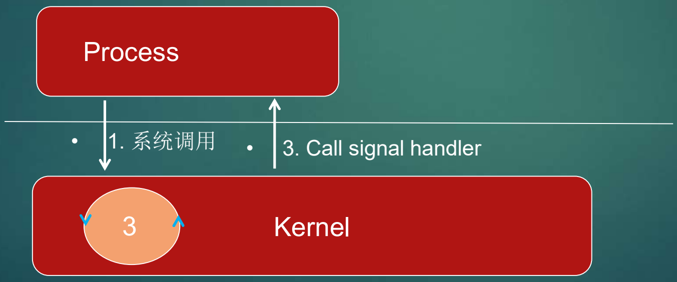
Step 3：内核太sys_kill处理进程B的调用请求，触发进程A的signal handler

6

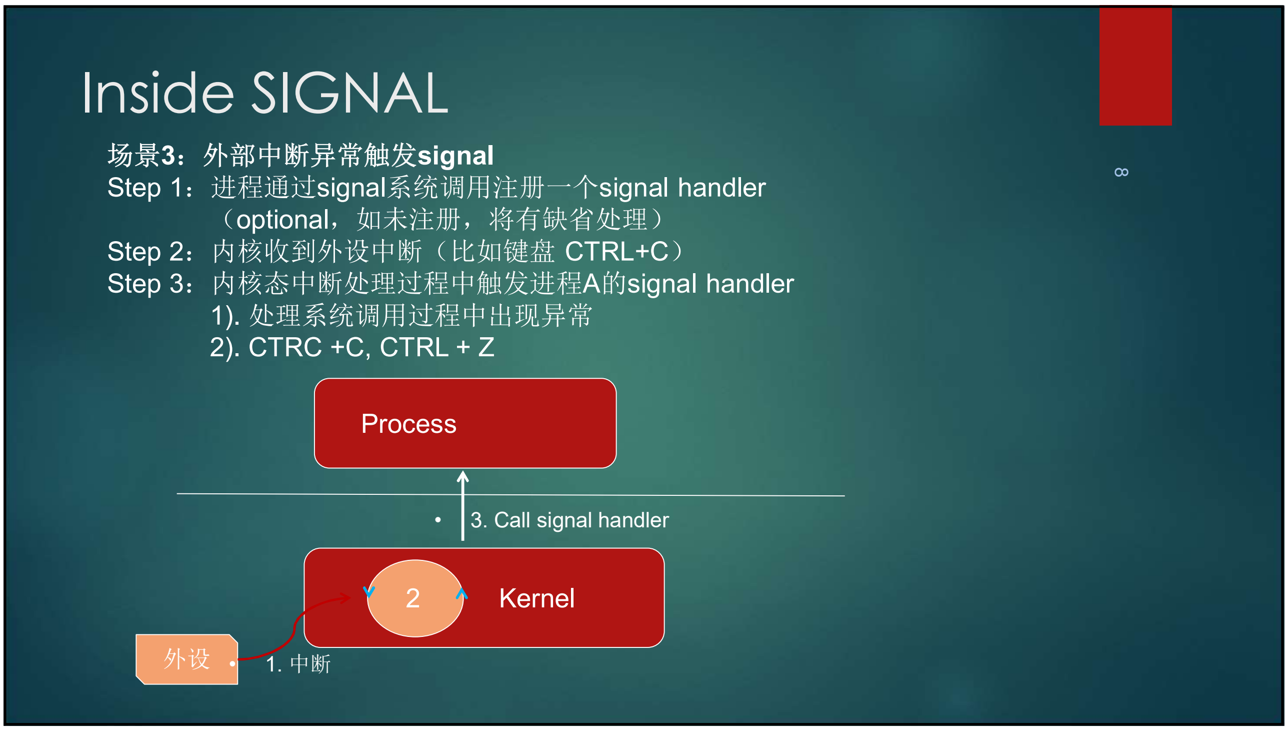


Inside SIGNAL

场景2：进程运行中的异常触发signal
Step 1：进程通过signal系统调用注册一个signal handler
（optional，如未注册，将有缺省处理）
Step 2：进程运行过程中触发系统调用（比如meimory处理）
Step 3：内核态处理系统调用过程中出现异常，触发进程A的signal handler
最常见的就是segmentation fault

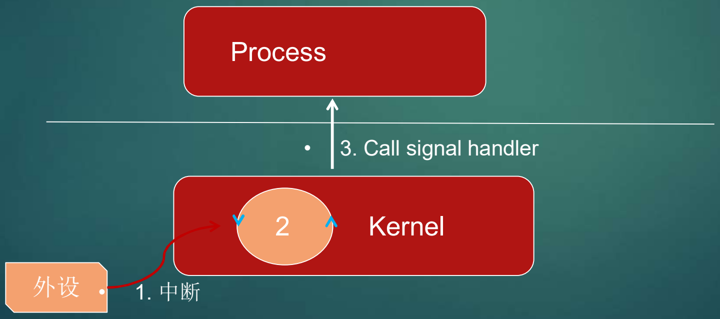


7

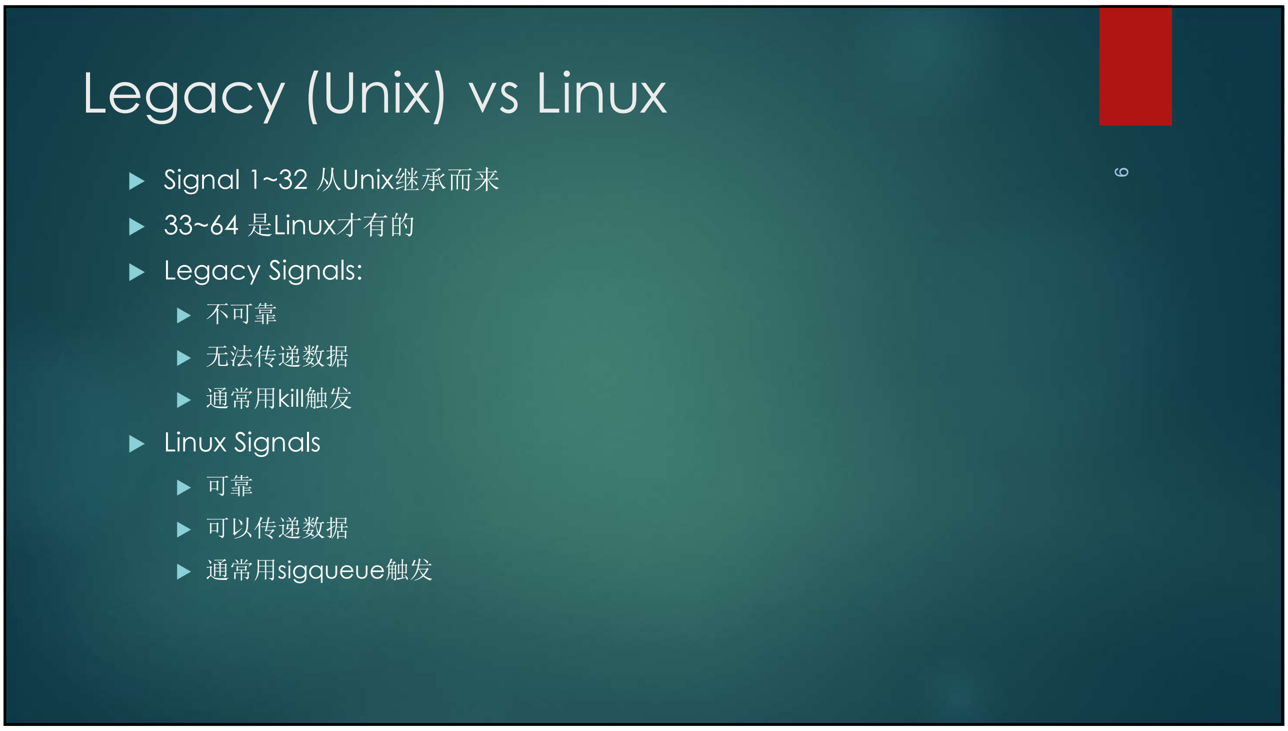


Inside SIGNAL

场景3：外部中断异常触发signal
Step 1：进程通过signal系统调用注册一个signal handler
（optional，如未注册，将有缺省处理）
Step 2：内核收到外设中断（比如键盘 CTRL+C）
Step 3：内核态中断处理过程中触发进程A的signal handler
1). 处理系统调用过程中出现异常
2). CTRC +C, CTRL + Z

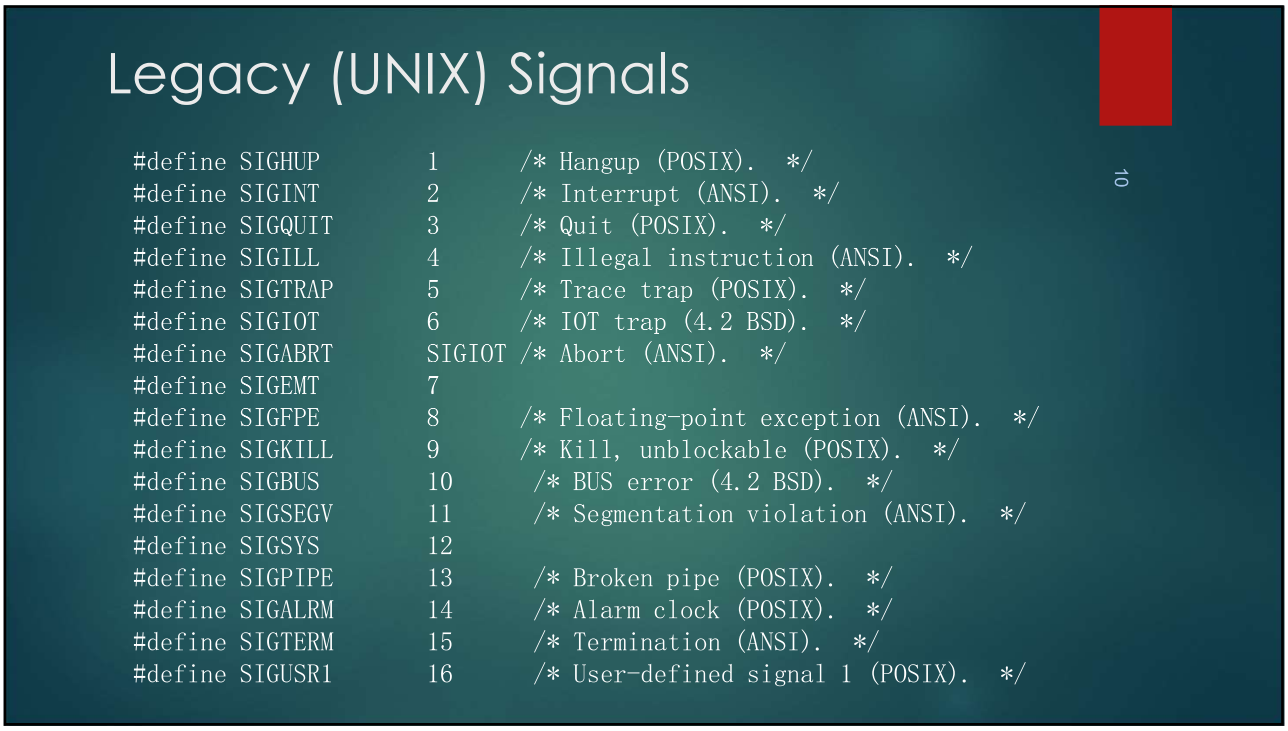


8



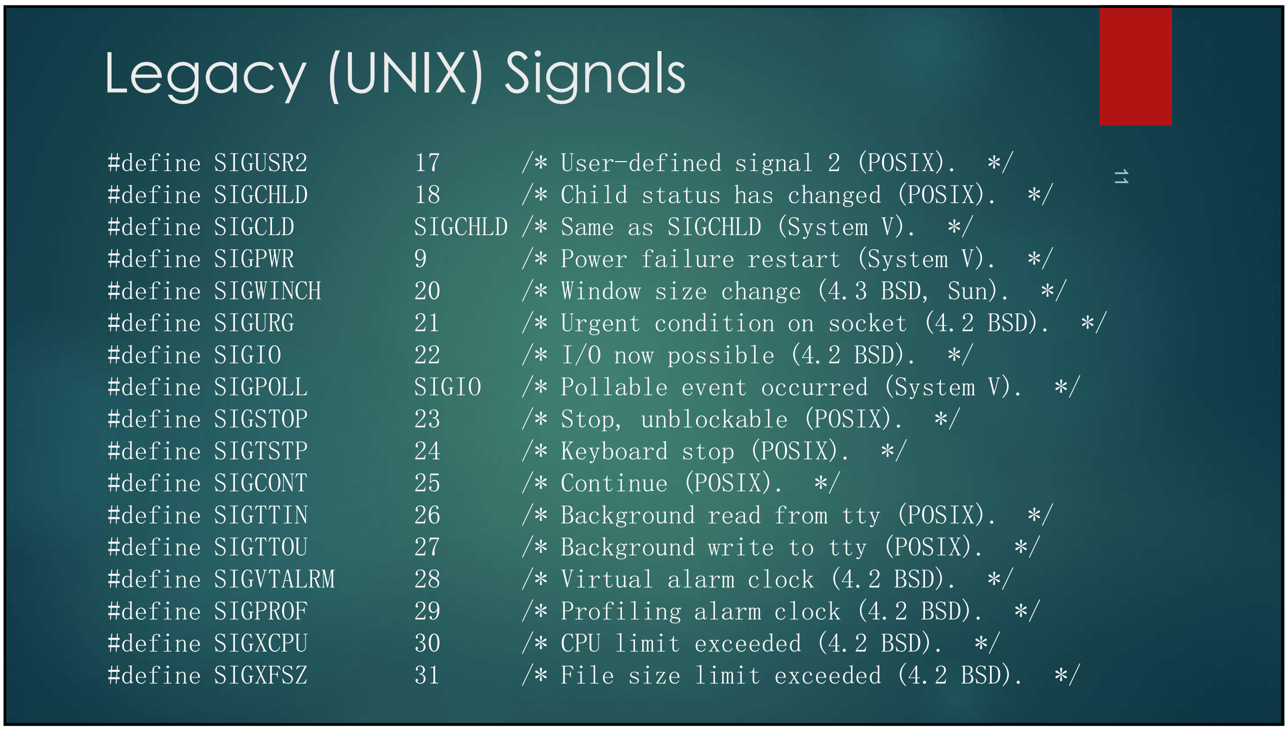
Legacy (Unix) vs Linux

- Signal 1~32 从Unix继承而来
- 33~64 是Linux才有的
- Legacy Signals:
  - 不可靠
  - 无法传递数据
  - 通常用kill触发
- Linux Signals
  - 可靠
  - 可以传递数据
  - 通常用sigqueue触发



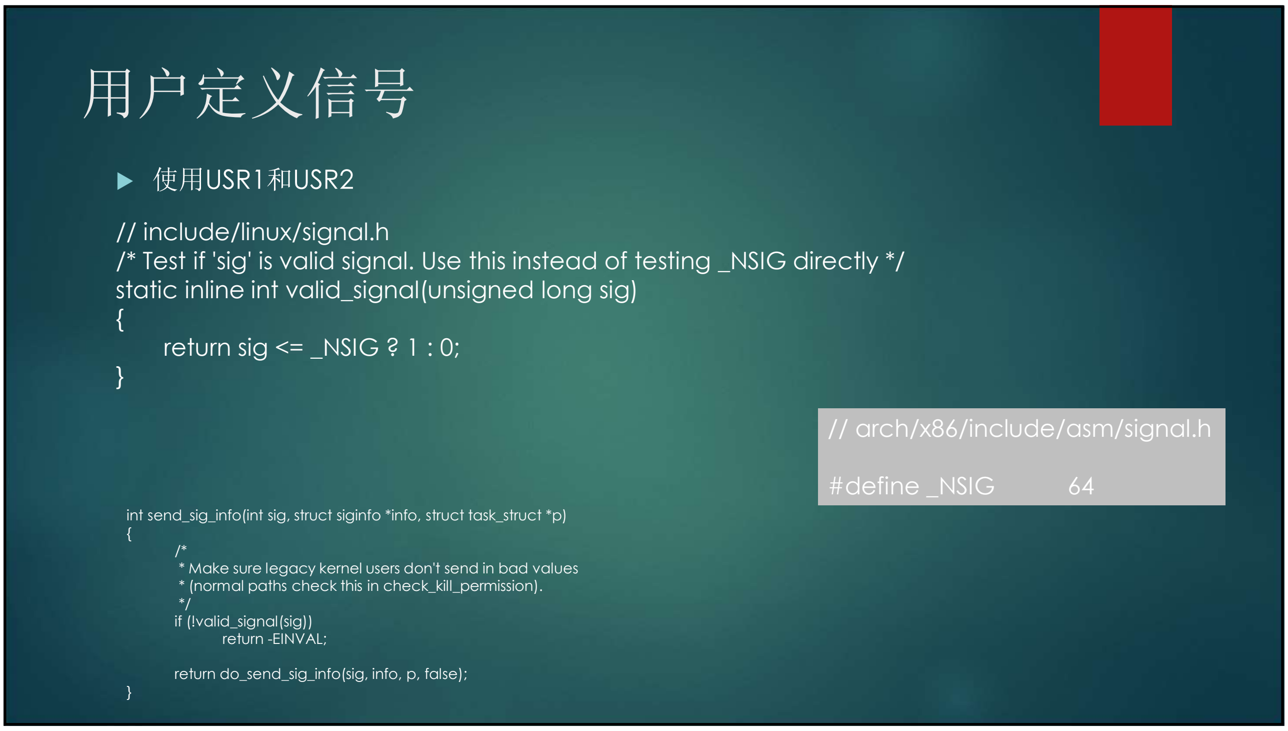
Legacy (UNIX) Signals

```
#define SIGHUP      1      /* Hangup (POSIX).  */
#define SIGINT      2      /* Interrupt (ANSI).  */
#define SIGQUIT     3      /* Quit (POSIX).  */
#define SIGILL      4      /* Illegal instruction (ANSI).  */
#define SIGTRAP     5      /* Trace trap (POSIX).  */
#define SIGIOT      6      /* IOT trap (4.2 BSD).  */
#define SIGABRT     SIGIOT /* Abort (ANSI).  */
#define SIGEMT      7
#define SIGFPE      8      /* Floating-point exception (ANSI).  */
#define SIGKILL     9      /* Kill, unblockable (POSIX).  */
#define SIGBUS      10      /* BUS error (4.2 BSD).  */
#define SIGSEGV     11      /* Segmentation violation (ANSI).  */
#define SIGSYS      12
#define SIGPIPE     13      /* Broken pipe (POSIX).  */
#define SIGALRM     14      /* Alarm clock (POSIX).  */
#define SIGTERM     15      /* Termination (ANSI).  */
#define SIGUSR1     16      /* User-defined signal 1 (POSIX).  */
```



Legacy (UNIX) Signals

```
#define SIGUSR2         17      /* User-defined signal 2 (POSIX).  */
#define SIGCHLD         18      /* Child status has changed (POSIX).  */
#define SIGCLD          SIGCHLD /* Same as SIGCHLD (System V).  */
#define SIGPWR          9       /* Power failure restart (System V).  */
#define SIGWINCH        20      /* Window size change (4.3 BSD, Sun).  */
#define SIGURG          21      /* Urgent condition on socket (4.2 BSD).  */
#define SIGIO           22      /* I/O now possible (4.2 BSD).  */
#define SIGPOLL         SIGIO   /* Pollable event occurred (System V).  */
#define SIGSTOP         23      /* Stop, unblockable (POSIX).  */
#define SIGTSTP         24      /* Keyboard stop (POSIX).  */
#define SIGCONT         25      /* Continue (POSIX).  */
#define SIGTTIN         26      /* Background read from tty (POSIX).  */
#define SIGTTOU         27      /* Background write to tty (POSIX).  */
#define SIGVTALRM       28      /* Virtual alarm clock (4.2 BSD).  */
#define SIGPROF         29      /* Profiling alarm clock (4.2 BSD).  */
#define SIGXCPU         30      /* CPU limit exceeded (4.2 BSD).  */
#define SIGXFSZ         31      /* File size limit exceeded (4.2 BSD).  */
```



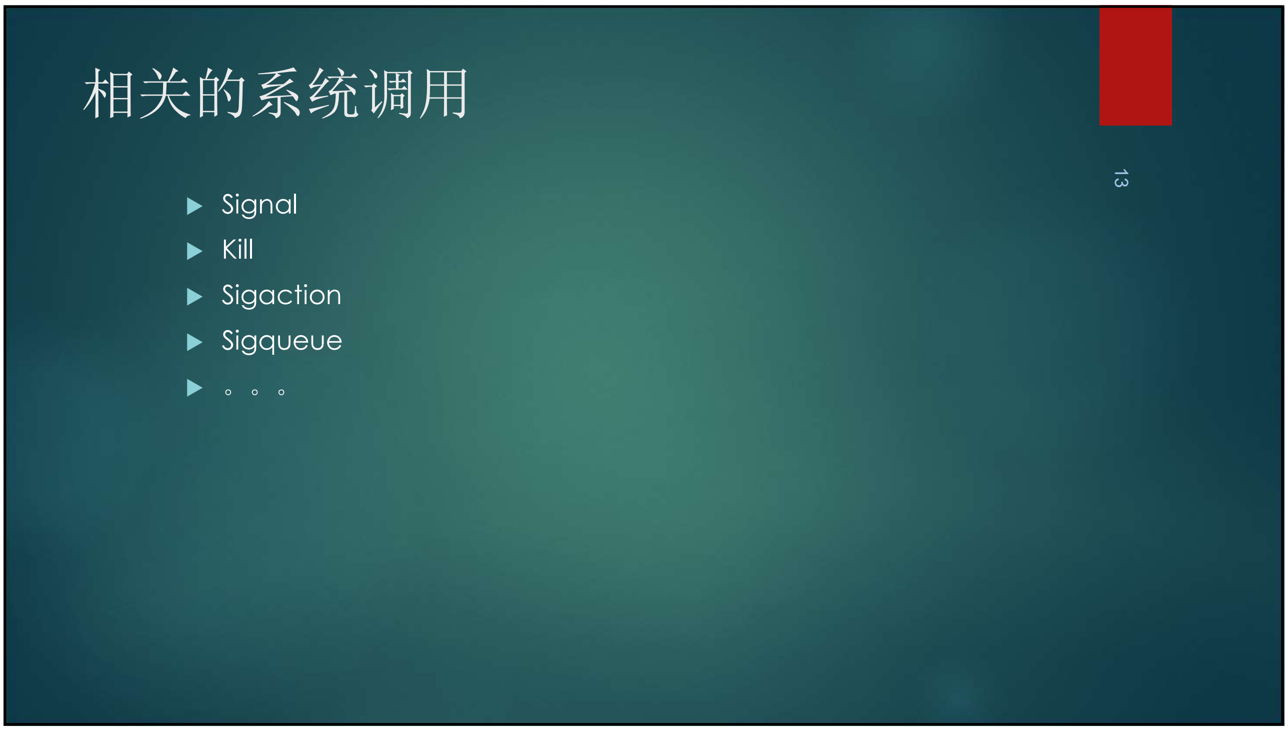
用户定义信号

- 使用USR1和USR2

```
// include/linux/signal.h
/* Test if 'sig' is valid signal. Use this instead of testing _NSIG directly */
static inline int valid_signal(unsigned long sig)
{
    return sig <= _NSIG ? 1 : 0;
}
```

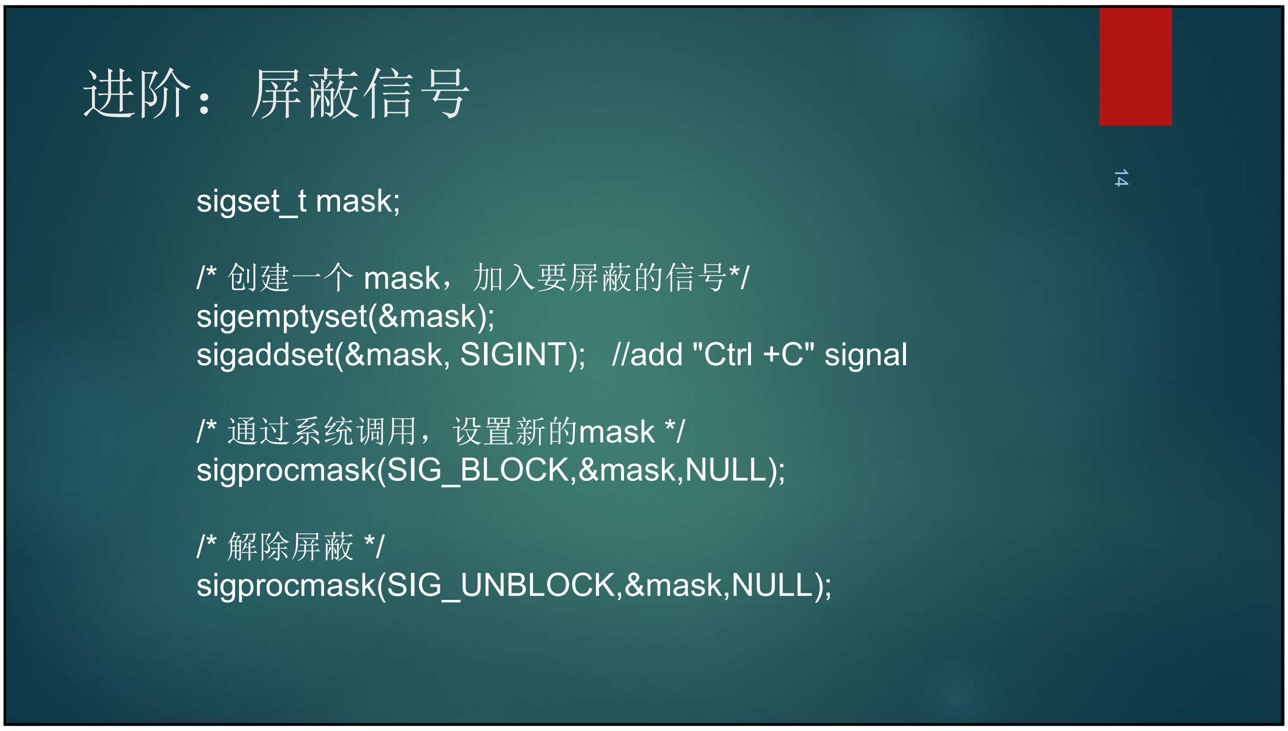
```
// arch/x86/include/asm/signal.h

#define _NSIG       64
```

```
int send_sig_info(int sig, struct siginfo *info, struct task_struct *p)
{
        /*
         * Make sure legacy kernel users don't send in bad values
         * (normal paths check this in check_kill_permission).
         */
        if (!valid_signal(sig))
                return -EINVAL;

        return do_send_sig_info(sig, info, p, false);
}
```



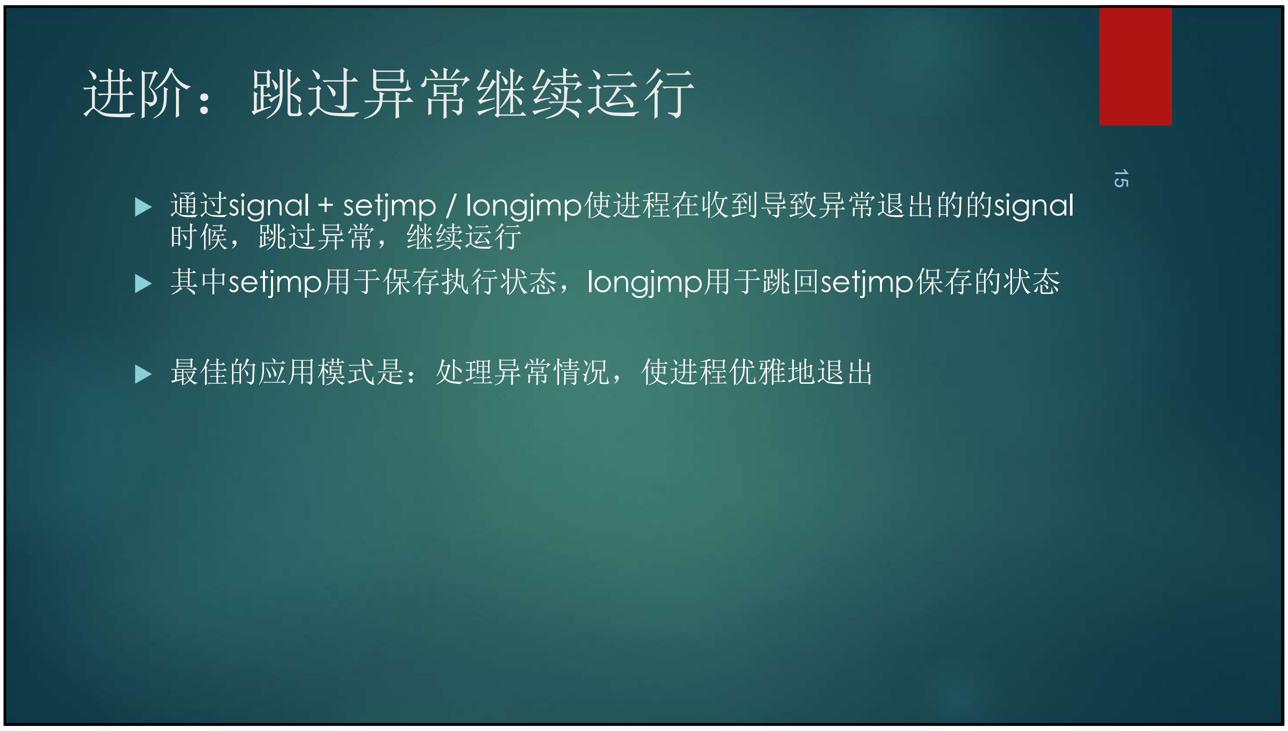
相关的系统调用

- Signal
- Kill
- Sigaction
- Sigqueue
- 。。。



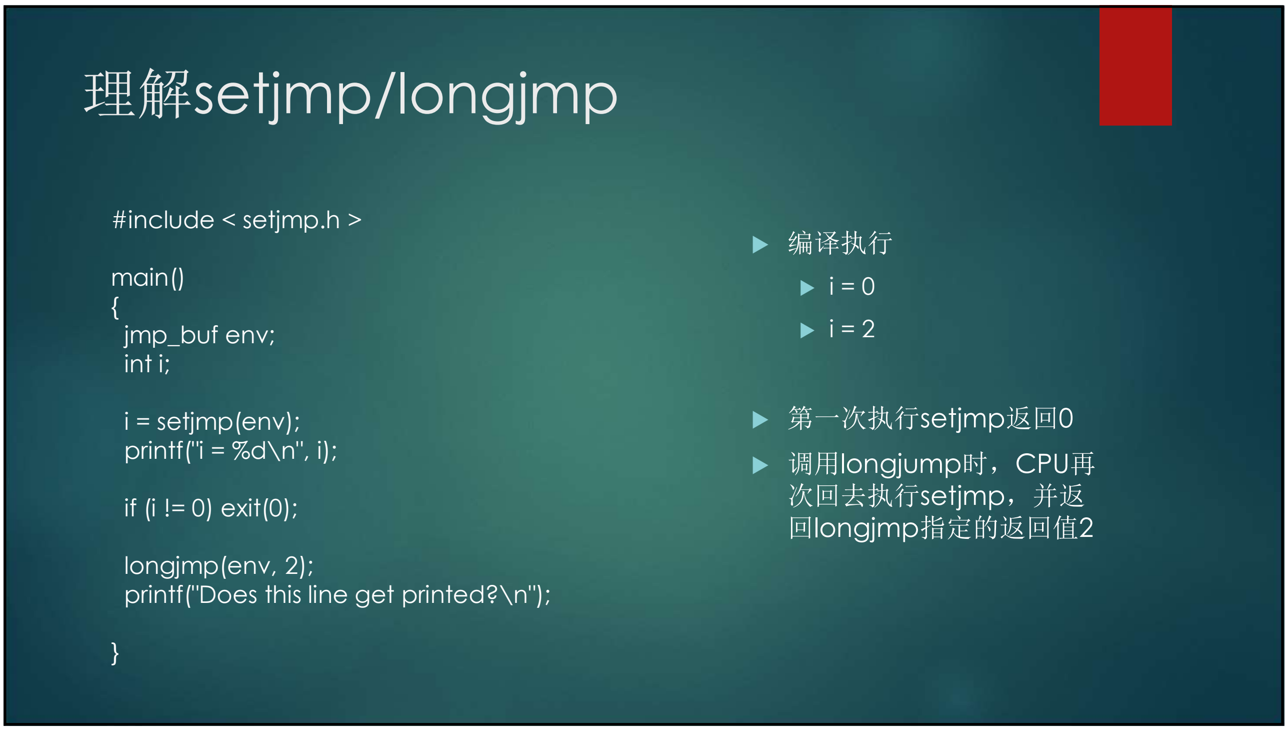
进阶：屏蔽信号

```
sigset_t mask;

/* 创建一个 mask，加入要屏蔽的信号*/
sigemptyset(&mask);
sigaddset(&mask, SIGINT);   //add "Ctrl +C" signal

/* 通过系统调用，设置新的mask */
sigprocmask(SIG_BLOCK,&mask,NULL);

/* 解除屏蔽 */
sigprocmask(SIG_UNBLOCK,&mask,NULL);
```



进阶：跳过异常继续运行

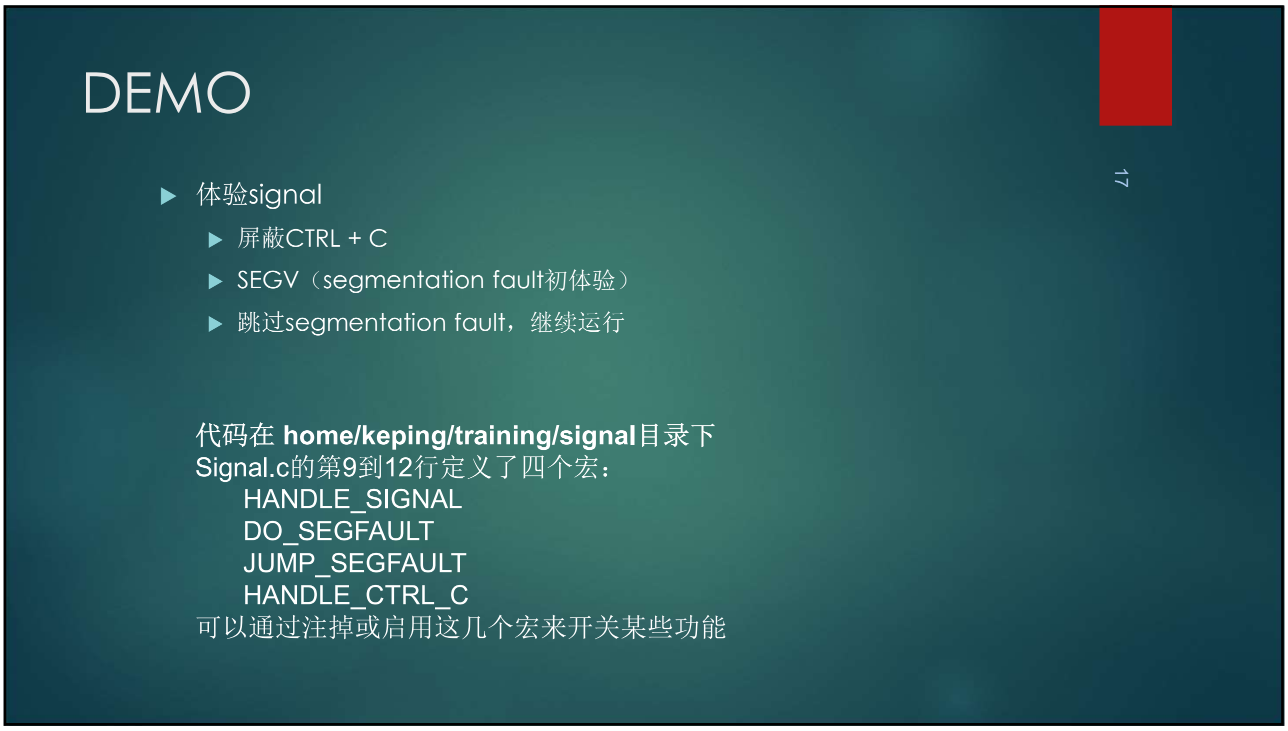
- 通过signal + setjmp / longjmp使进程在收到导致异常退出的的signal时候，跳过异常，继续运行
- 其中setjmp用于保存执行状态，longjmp用于跳回setjmp保存的状态
- 最佳的应用模式是：处理异常情况，使进程优雅地退出



理解setjmp/longjmp

```
#include < setjmp.h >

main()
{
  jmp_buf env;
  int i;

  i = setjmp(env);
  printf("i = %d\n", i);

  if (i != 0) exit(0);

  longjmp(env, 2);
  printf("Does this line get printed?\n");

}
```

- 编译执行
  - i = 0
  - i = 2
- 第一次执行setjmp返回0
- 调用longjump时，CPU再次回去执行setjmp，并返回longjmp指定的返回值2



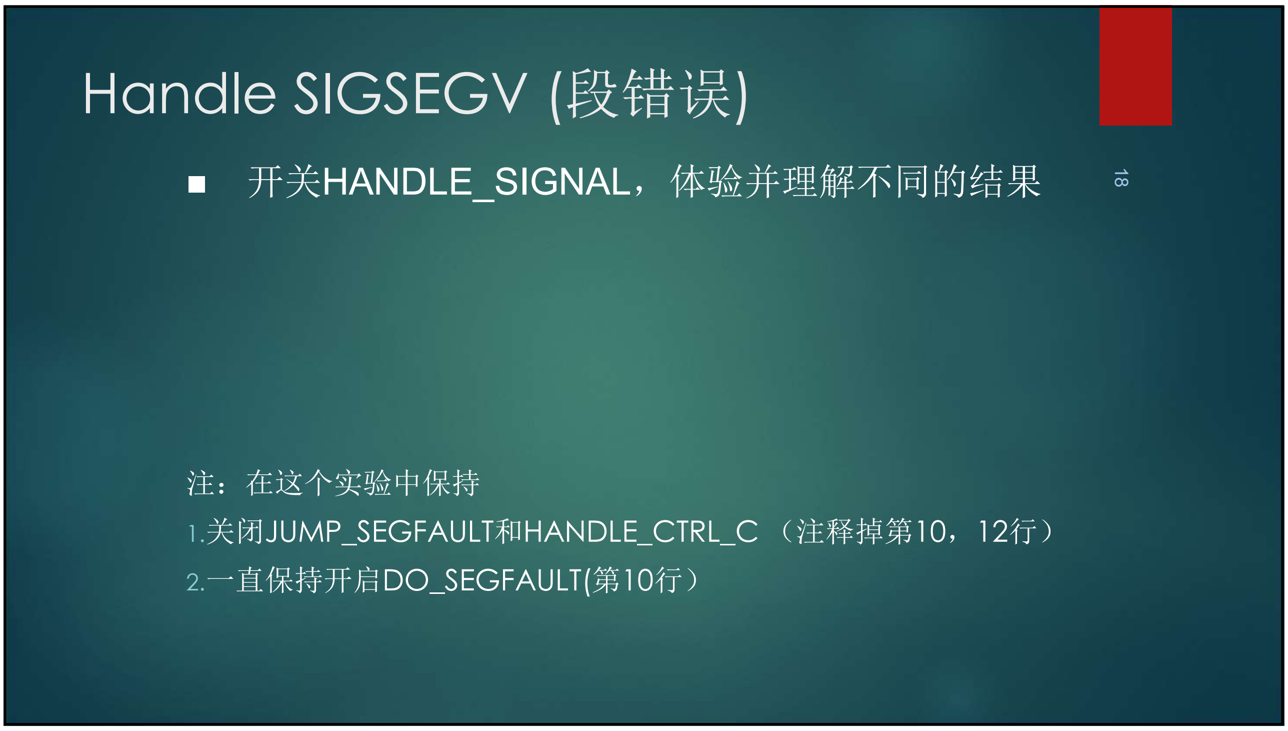
DEMO

- 体验signal
  - 屏蔽CTRL + C
  - SEGV（segmentation fault初体验）
  - 跳过segmentation fault，继续运行

代码在 **home/keping/training/signal**目录下
Signal.c的第9到12行定义了四个宏：
　　HANDLE_SIGNAL
　　DO_SEGFAULT
　　JUMP_SEGFAULT
　　HANDLE_CTRL_C
可以通过注掉或启用这几个宏来开关某些功能

17

17



Handle SIGSEGV (段错误)

- 开关HANDLE_SIGNAL，体验并理解不同的结果

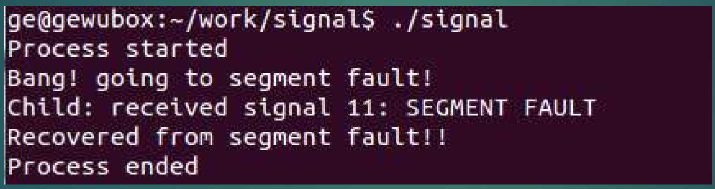
注：在这个实验中保持
1.关闭JUMP_SEGFAULT和HANDLE_CTRL_C （注释掉第10，12行）
2.一直保持开启DO_SEGFAULT(第10行）

18

18

跳过段错误

- 开启HANDLE_SIGNAL，DO_SEGFAULT和JUMP_SEGFAULT，关闭HANDLE_CTRL_C，体验通过setjump / longjmp处理段错误的效果

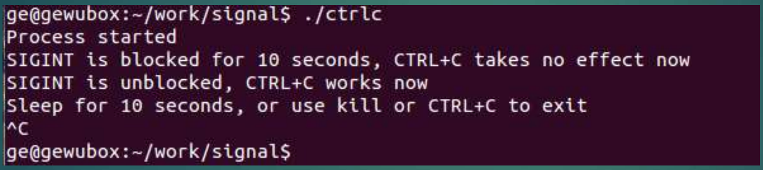


```
ge@gewubox:~/work/signal$ ./signal
Process started
Bang! going to segment fault!
Child: received signal 11: SEGMENT FAULT
Recovered from segment fault!!
Process ended
```

源代码位于setjmp/longjmp.c
X64的核心实现在sysdep/x86_64/__longjmp.S

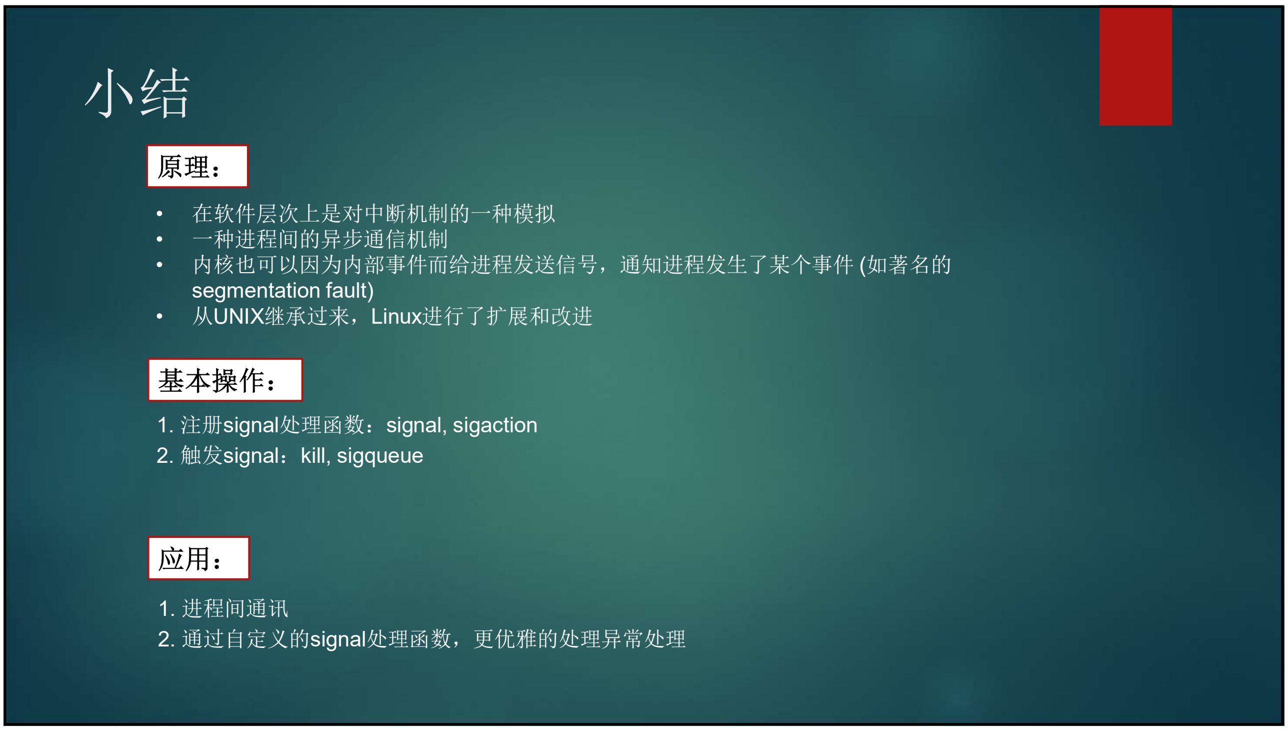
屏蔽CTRL + C

- 开关HANDLE_SIGNAL和HANDLE_CTRL_C，重新编译并运行 ./signal
- Signal运行起来后，按CTRL+C



```
ge@gewubox:~/work/signal$ ./ctrlc
Process started
SIGINT is blocked for 10 seconds, CTRL+C takes no effect now
SIGINT is unblocked, CTRL+C works now
Sleep for 10 seconds, or use kill or CTRL+C to exit
^C
ge@gewubox:~/work/signal$
```

注：在这个实验中关闭DO_SEGFAULT 和JUMP_SEGFAULT （注释掉第9，10行）



小结

原理：

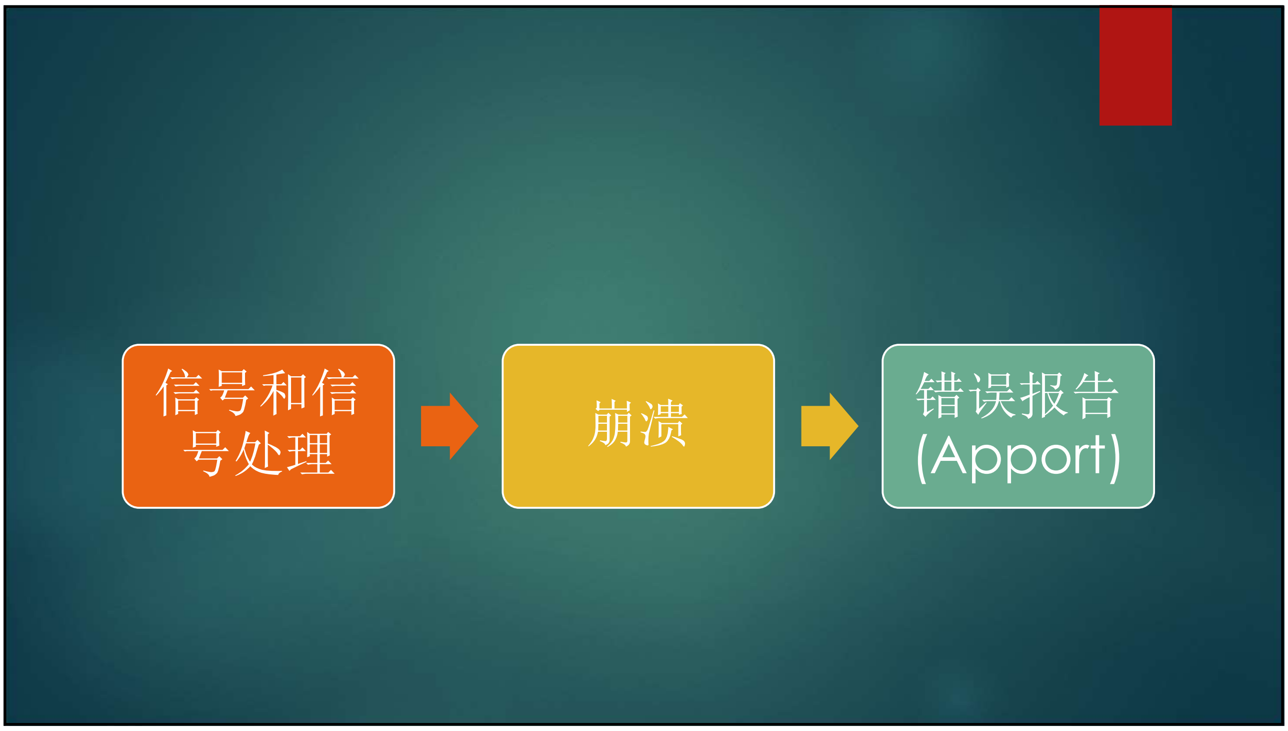
- 在软件层次上是对中断机制的一种模拟
- 一种进程间的异步通信机制
- 内核也可以因为内部事件而给进程发送信号，通知进程发生了某个事件 (如著名的segmentation fault)
- 从UNIX继承过来，Linux进行了扩展和改进

基本操作：

1. 注册signal处理函数：signal, sigaction
2. 触发signal：kill, sigqueue

应用：

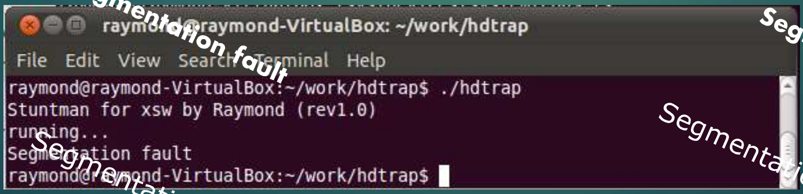
1. 进程间通讯
2. 通过自定义的signal处理函数，更优雅的处理异常处理



$$\begin{tikzcd}\text{信号和信号处理} \arrow[r] & \text{崩溃} \arrow[r] & \text{错误报告(Apport)}\end{tikzcd}$$

背景

- 重要软件的后台服务随机崩溃
- 著名的段错误
  - 在控制台窗口运行
  - 留下一句话Segmentation fault
- “替身”小程序，模拟故障


raymond@raymond-VirtualBox: ~/work/hdtrap
File Edit View Search Terminal Help

```
raymond@raymond-VirtualBox:~/work/hdtrap$ ./hdtrap
Stuntman for xsw by Raymond (rev1.0)
running...
Segmentation fault
raymond@raymond-VirtualBox:~/work/hdtrap$
```

Segmentation fault Segmentation fault Segmentation fault Segmentation fault Segmentation fault Segmentation fault Segmentation fault Segmentation fault Segmentation fault Segmentation fault

23

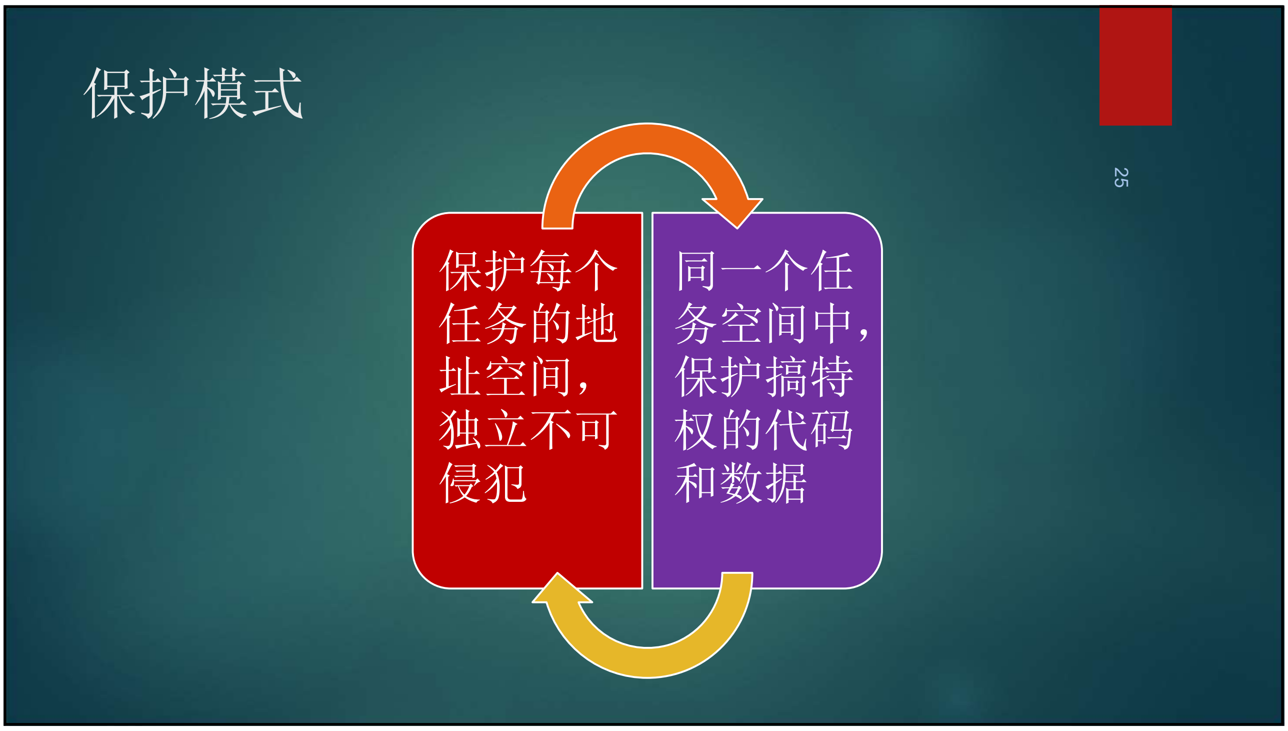
23

24

demo

感受段错误

24

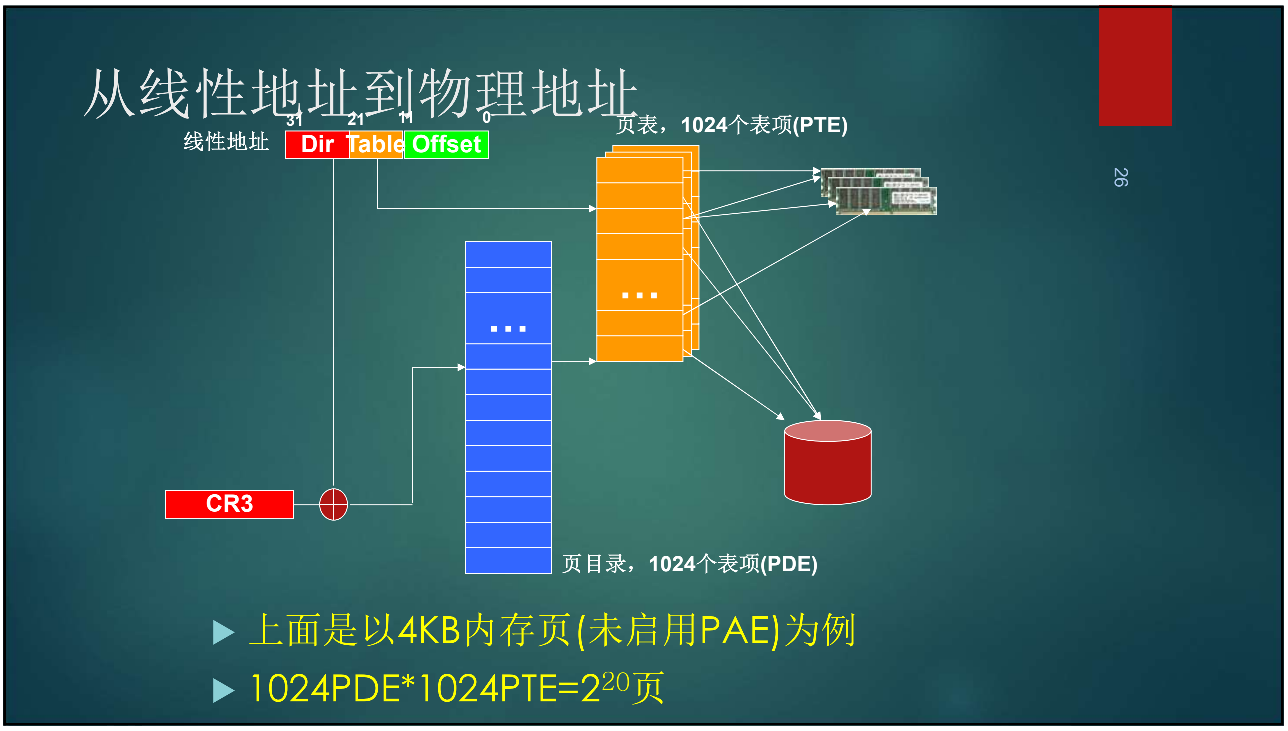


保护模式

保护每个任务的地址空间，独立不可侵犯

同一个任务空间中，保护搞特权的代码和数据

25

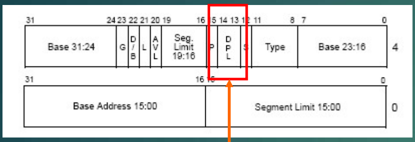


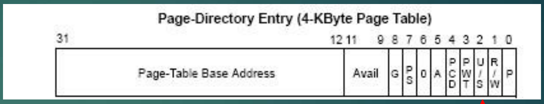
从线性地址到物理地址

- 上面是以4KB内存页(未启用PAE)为例
- 1024PDE*1024PTE=2^{20}页

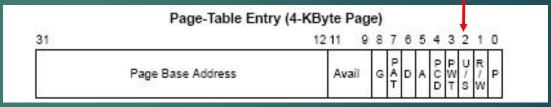
26

特权和秩序





U/S – User(1)/Supervisor(0)



DPL – Descriptor Privilege Level

- 正在执行的代码所在代码段的特权级别就是该代码的特权级别
- 当一段代码调用位于其它段的函数或访问其它段的数据时，CPU会检查发起访问者是否有足够的权限。如果没有通过检查，则产生异常

空指针

- NULL Pointer
- In computing, a null pointer has a value reserved for indicating that the pointer does not refer to a valid object.

I call it my billion-dollar mistake. It was the invention of the null reference in 1965. At that time, I was designing the first comprehensive type system for references in an object oriented language (ALGOL W). My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement. This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.

Charles Antony Richard Hoare

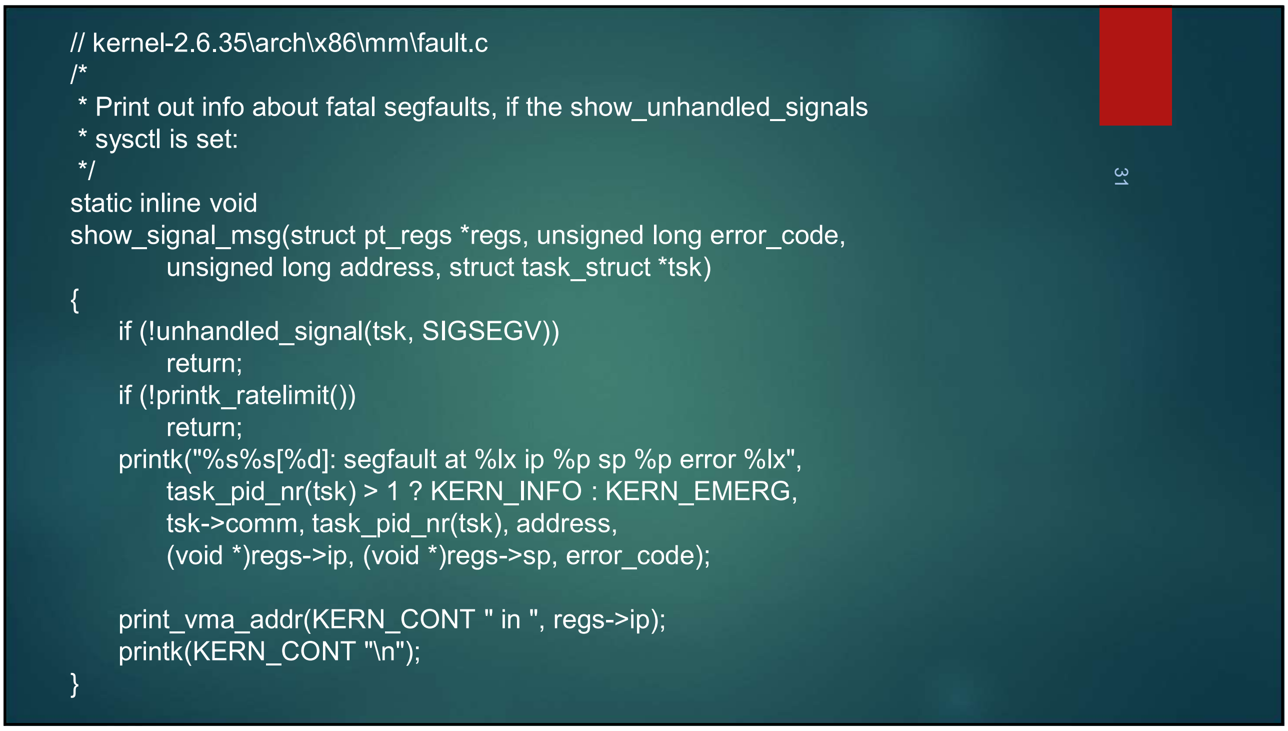
违反规则

- 异常
- Linux
  - 段错误
  - 名字比实际范围小
- Windows
  - 访问违例
  - Access violation
  - 0xC0000005

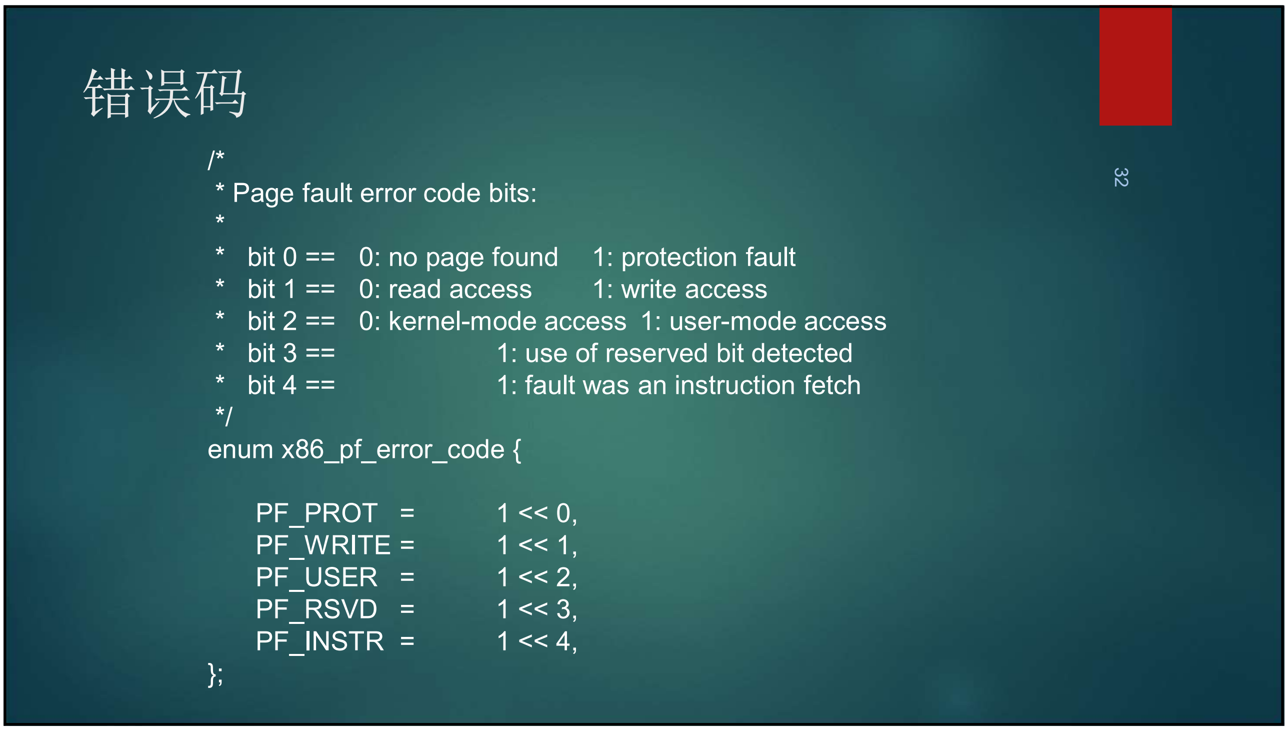
dmesg

```
[25402.126121] hdtrap[1960]: segfault at 0 ip 080484b8 sp bfee0c28 error 6 in
hdtrap[8048000+1000]
```

- IP地址为0x080484b8的指令访问地址0导致了段错误，当时的栈指针是0xbfee0c28，错误码是6（用户态的写操作，参见下文），指令所属的内存块的基地址为8048000，长度为0x1000



```
// kernel-2.6.35\arch\x86\mm\fault.c
/*
 * Print out info about fatal segfaults, if the show_unhandled_signals
 * sysctl is set:
 */
static inline void
show_signal_msg(struct pt_regs *regs, unsigned long error_code,
        unsigned long address, struct task_struct *tsk)
{
    if (!unhandled_signal(tsk, SIGSEGV))
        return;
    if (!printk_ratelimit())
        return;
    printk("%s%s[%d]: segfault at %lx ip %p sp %p error %lx",
        task_pid_nr(tsk) > 1 ? KERN_INFO : KERN_EMERG,
        tsk->comm, task_pid_nr(tsk), address,
        (void *)regs->ip, (void *)regs->sp, error_code);

    print_vma_addr(KERN_CONT " in ", regs->ip);
    printk(KERN_CONT "\n");
}
```

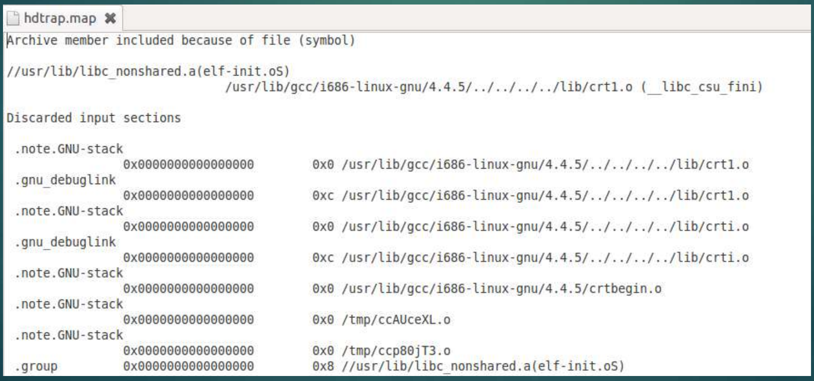


错误码

```
/*
 * Page fault error code bits:
 *
 *   bit 0 ==   0: no page found      1: protection fault
 *   bit 1 ==   0: read access        1: write access
 *   bit 2 ==   0: kernel-mode access  1: user-mode access
 *   bit 3 ==                         1: use of reserved bit detected
 *   bit 4 ==                         1: fault was an instruction fetch
 */
enum x86_pf_error_code {

    PF_PROT  =         1 << 0,
    PF_WRITE =         1 << 1,
    PF_USER  =         1 << 2,
    PF_RSVD  =         1 << 3,
    PF_INSTR =         1 << 4,
};
```

Map文件

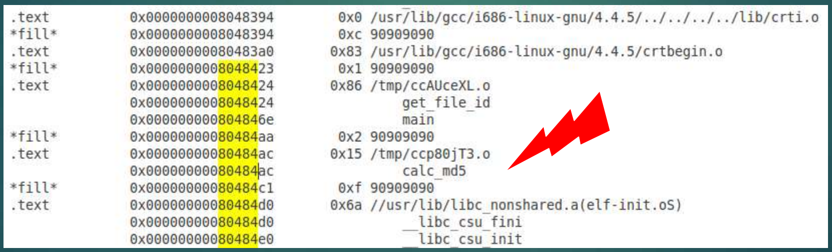
- 产生Map文件
  - gcc -o hdtrap hdtrap.c md5.c -Wl,-Map=hdtrap.map
  - 注意W后是小写L，代表链接选项



```
hdtrap.map
Archive member included because of file (symbol)

//usr/lib/libc_nonshared.a(elf-init.oS)
                              /usr/lib/gcc/i686-linux-gnu/4.4.5/../../../../lib/crt1.o (__libc_csu_fini)

Discarded input sections

 .note.GNU-stack
                0x0000000000000000        0x0 /usr/lib/gcc/i686-linux-gnu/4.4.5/../../../../lib/crt1.o
 .gnu_debuglink
                0x0000000000000000        0xc /usr/lib/gcc/i686-linux-gnu/4.4.5/../../../../lib/crt1.o
 .note.GNU-stack
                0x0000000000000000        0x0 /usr/lib/gcc/i686-linux-gnu/4.4.5/../../../../lib/crti.o
 .gnu_debuglink
                0x0000000000000000        0xc /usr/lib/gcc/i686-linux-gnu/4.4.5/../../../../lib/crti.o
 .note.GNU-stack
                0x0000000000000000        0x0 /usr/lib/gcc/i686-linux-gnu/4.4.5/crtbegin.o
 .note.GNU-stack
                0x0000000000000000        0x0 /tmp/ccAUceXL.o
 .note.GNU-stack
                0x0000000000000000        0x0 /tmp/ccp80jT3.o
 .group         0x0000000000000000        0x8 //usr/lib/libc_nonshared.a(elf-init.oS)
```

查找

- [27566.531498] hdtrap[2504]: segfault at 0 ip 080484b8 sp bff209d8 error 6 in hdtrap[8048000+1000]



```
.text          0x0000000008048394       0x0 /usr/lib/gcc/i686-linux-gnu/4.4.5/../../../../lib/crti.o
*fill*         0x0000000008048394       0xc 90909090
.text          0x00000000080483a0      0x83 /usr/lib/gcc/i686-linux-gnu/4.4.5/crtbegin.o
*fill*         0x0000000008048423       0x1 90909090
.text          0x0000000008048424      0x86 /tmp/ccAUceXL.o
               0x0000000008048424               get_file_id
               0x000000000804846e               main
*fill*         0x00000000080484aa       0x2 90909090
.text          0x00000000080484ac      0x15 /tmp/ccp80jT3.o
               0x00000000080484ac               calc_md5
*fill*         0x00000000080484c1       0xf 90909090
.text          0x00000000080484d0      0x6a //usr/lib/libc_nonshared.a(elf-init.oS)
               0x00000000080484d0               __libc_csu_fini
               0x00000000080484e0               __libc_csu_init
```

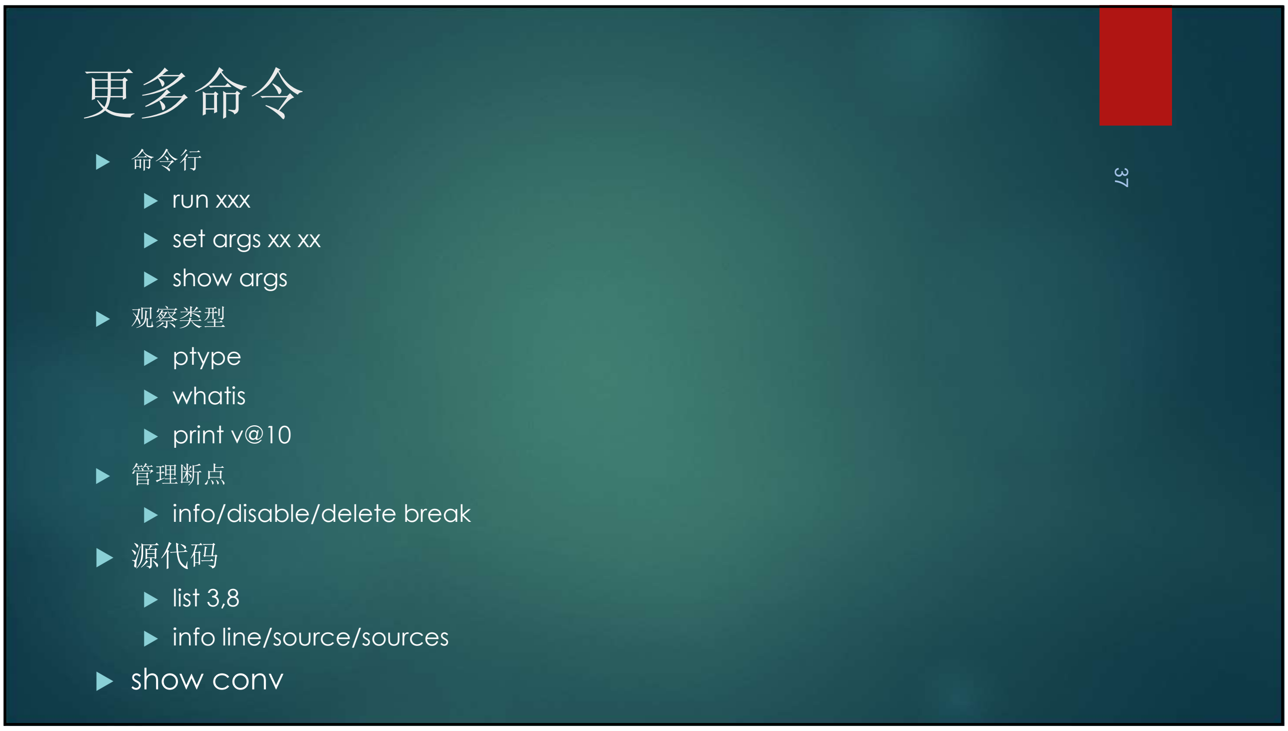
有请GDB

- GNU Debugger
  - 1986年，Richard Stallman创建
  - 1900-1993，John Gilmore维护
  - 目前在GDB Steering Committee
  - 最新版本7.3 (2011.7.26)
  - 支持很多种CPU architecture
  - A29K, ARC, ETRAX CRIS, D10V, D30V, FR-30, FR-V, Intel i960, M32R, 68HC11, Motorola 88000, MCORE, MN10200, MN10300, NS32K, Stormy16, V850, Z8000 and many more
  - 默认为命令行界面,有很多GUI的前端(Add-on)

35

常用调试命令对照表

WinDBG命令	GDB命令	功能
bp	break或b	设置软件断点
ba	watch	设置硬件断点
k	backtrace或bt	显示函数调用序列（栈回溯）
g	continue或c	恢复执行
p/t	next/step或n/s	单步跟踪
d	x	观察内存
dv	info locals	观察局部变量
dt	pt	观察数据类型（结构）
gu	finish	执行到函数返回
.frame	frame	切换当前栈帧
lm	i shared	列模块

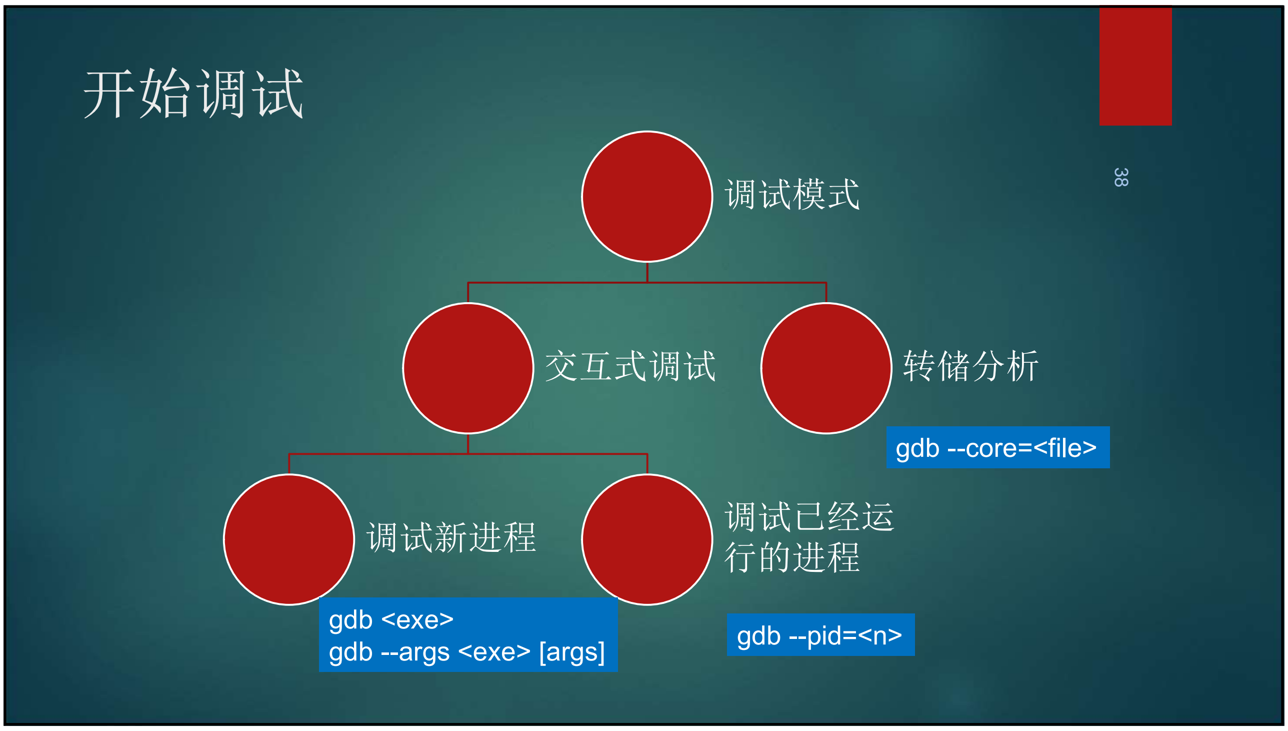


更多命令

- 命令行
  - run xxx
  - set args xx xx
  - show args
- 观察类型
  - ptype
  - whatis
  - print v@10
- 管理断点
  - info/disable/delete break
- 源代码
  - list 3,8
  - info line/source/sources
- show conv

37

37



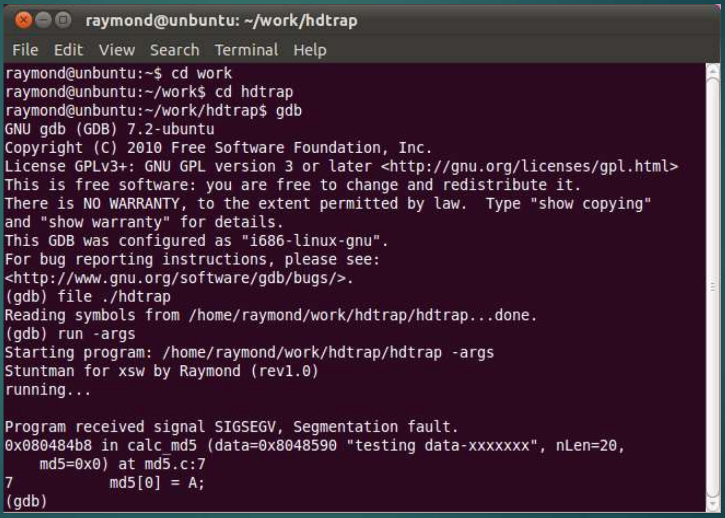
开始调试

38

38

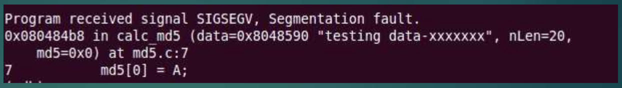
或者

- gdb
- file <exe>
- run [args]


raymond@unbuntu: ~/work/hdtrap
File Edit View Search Terminal Help

```
raymond@unbuntu:~$ cd work
raymond@unbuntu:~/work$ cd hdtrap
raymond@unbuntu:~/work/hdtrap$ gdb
GNU gdb (GDB) 7.2-ubuntu
Copyright (C) 2010 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 "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) file ./hdtrap
Reading symbols from /home/raymond/work/hdtrap/hdtrap...done.
(gdb) run -args
Starting program: /home/raymond/work/hdtrap/hdtrap -args
Stuntman for xsw by Raymond (rev1.0)
running...

Program received signal SIGSEGV, Segmentation fault.
0x080484b8 in calc_md5 (data=0x8048590 "testing data-xxxxxxx", nLen=20,
    md5=0x0) at md5.c:7
7           md5[0] = A;
(gdb)
```

案发函数

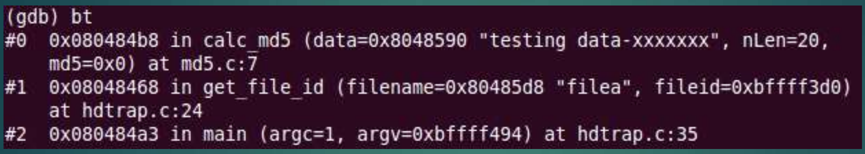


```
Program received signal SIGSEGV, Segmentation fault.
0x080484b8 in calc_md5 (data=0x8048590 "testing data-xxxxxxx", nLen=20,
    md5=0x0) at md5.c:7
7           md5[0] = A;
```

```
int calc_md5(char * data,
            int nLen,
            unsigned int md5[4])
{
    int A, B, C, D;

    // calc md5 for data array now

    // assign the result now
    md5[0] = A;
    // ...

    return 0;
}
```

md5等于0，明显访问空指针

追查调用者

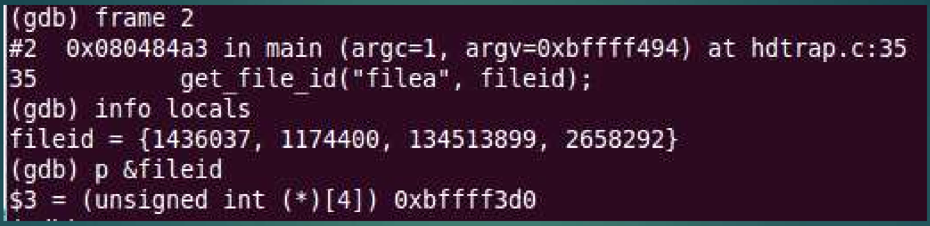


```
(gdb) bt
#0  0x080484b8 in calc_md5 (data=0x8048590 "testing data-xxxxxxx", nLen=20,
    md5=0x0) at md5.c:7
#1  0x08048468 in get_file_id (filename=0x80485d8 "filea", fileid=0xbffff3d0)
    at hdtrap.c:24
#2  0x080484a3 in main (argc=1, argv=0xbffff494) at hdtrap.c:35
```

```
int get_file_id(char * filename, unsigned int * fileid)
{
    SECTION_HEAD section;
    section.data = "testing data-xxxxxxx";
    section.length = strlen(section.data);
    return calc_md5(section.data,
        section.length, fileid);
}
```

也是来自参数

继续上溯



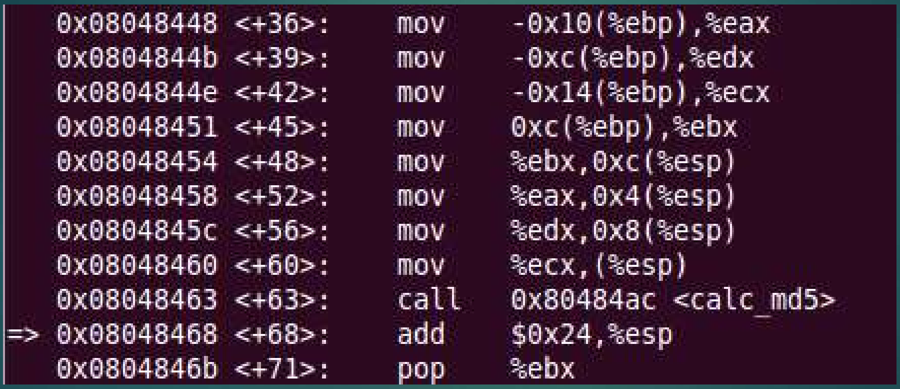
```
(gdb) frame 2
#2  0x080484a3 in main (argc=1, argv=0xbffff494) at hdtrap.c:35
35          get_file_id("filea", fileid);
(gdb) info locals
fileid = {1436037, 1174400, 134513899, 2658292}
(gdb) p &fileid
$3 = (unsigned int (*)[4]) 0xbffff3d0
```

- 在main这一级，传递下去的是局部数组变量fileid，显然不为空
- 传了两次就出问题了？？

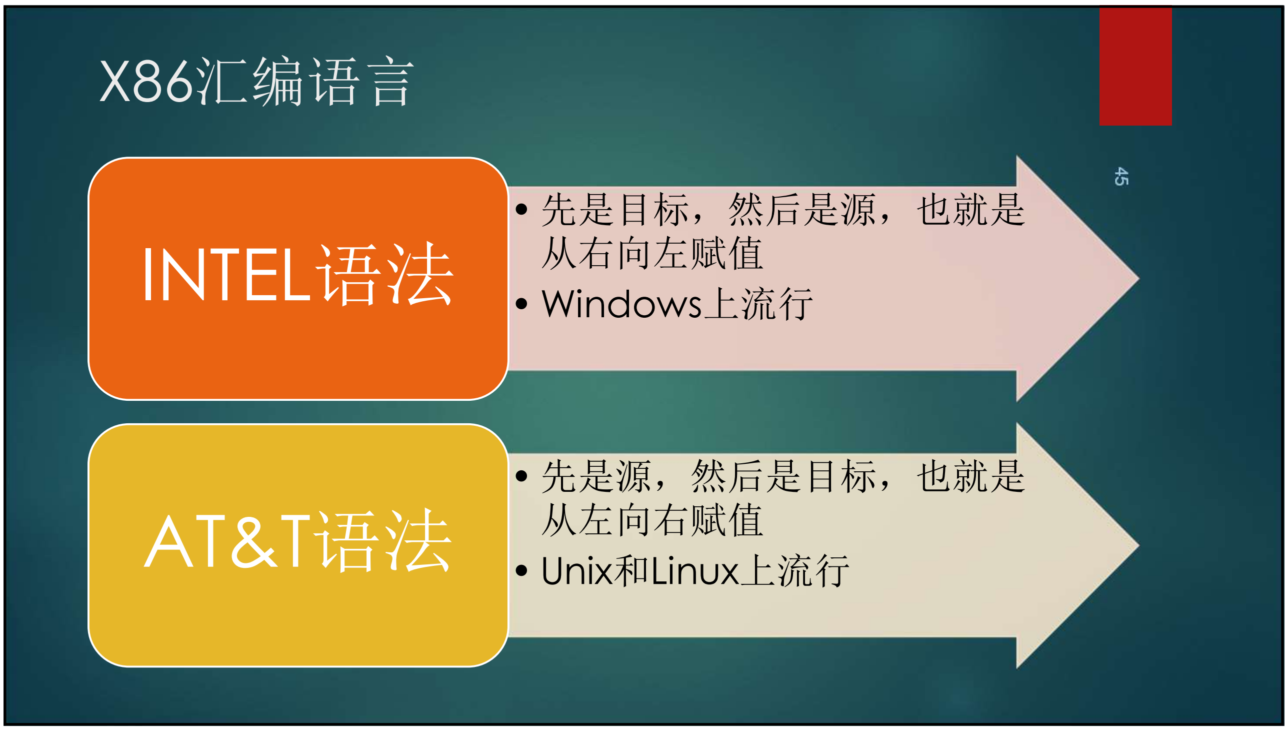
主函数

```
int main(int argc, char * argv[])
{
    unsigned int fileid[4];

    printf("Stuntman for xsw by Raymond (rev1.0)\n");
    printf("running...\n");

    get_file_id("filea", fileid);

    return getchar();
}
```

审查传递过程



```
   0x08048448 <+36>:    mov    -0x10(%ebp),%eax
   0x0804844b <+39>:    mov    -0xc(%ebp),%edx
   0x0804844e <+42>:    mov    -0x14(%ebp),%ecx
   0x08048451 <+45>:    mov    0xc(%ebp),%ebx
   0x08048454 <+48>:    mov    %ebx,0xc(%esp)
   0x08048458 <+52>:    mov    %eax,0x4(%esp)
   0x0804845c <+56>:    mov    %edx,0x8(%esp)
   0x08048460 <+60>:    mov    %ecx,(%esp)
   0x08048463 <+63>:    call   0x80484ac <calc_md5>
=> 0x08048468 <+68>:    add    $0x24,%esp
   0x0804846b <+71>:    pop    %ebx
```



X86汇编语言

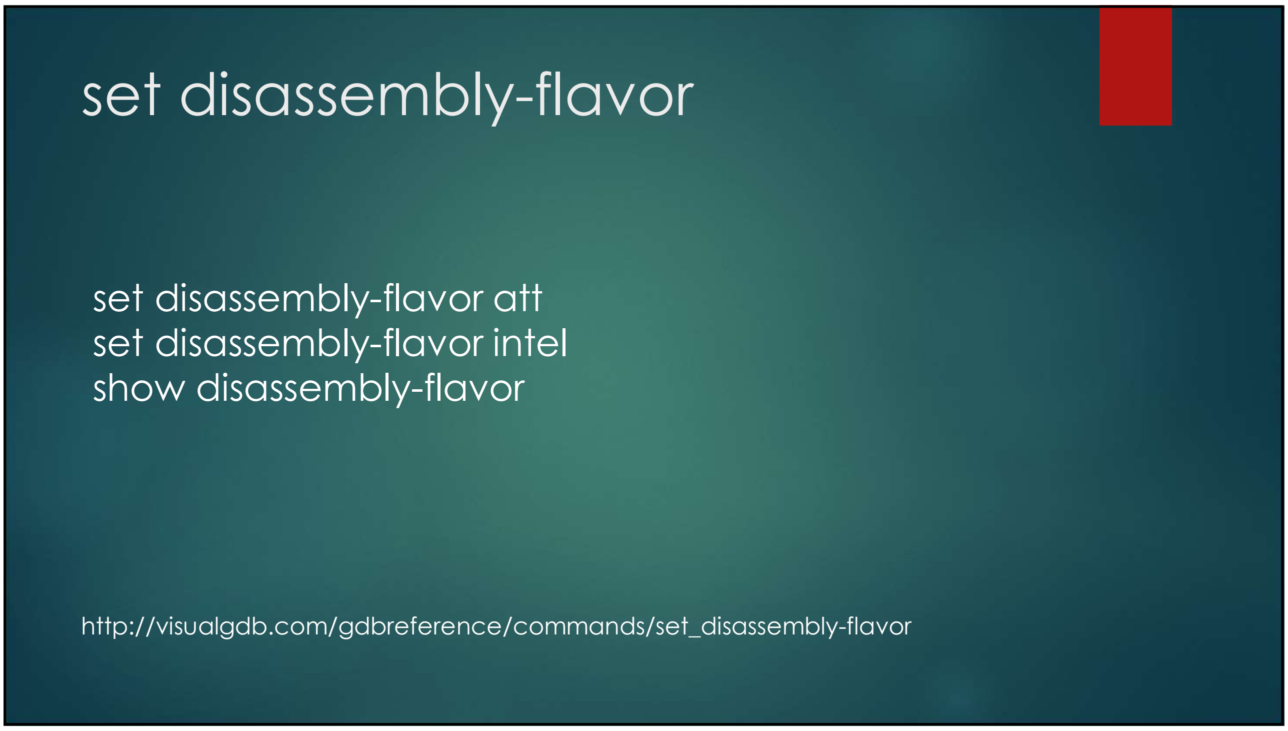
INTEL语法

- 先是目标，然后是源，也就是从右向左赋值
- Windows上流行

AT&T语法

- 先是源，然后是目标，也就是从左向右赋值
- Unix和Linux上流行

45

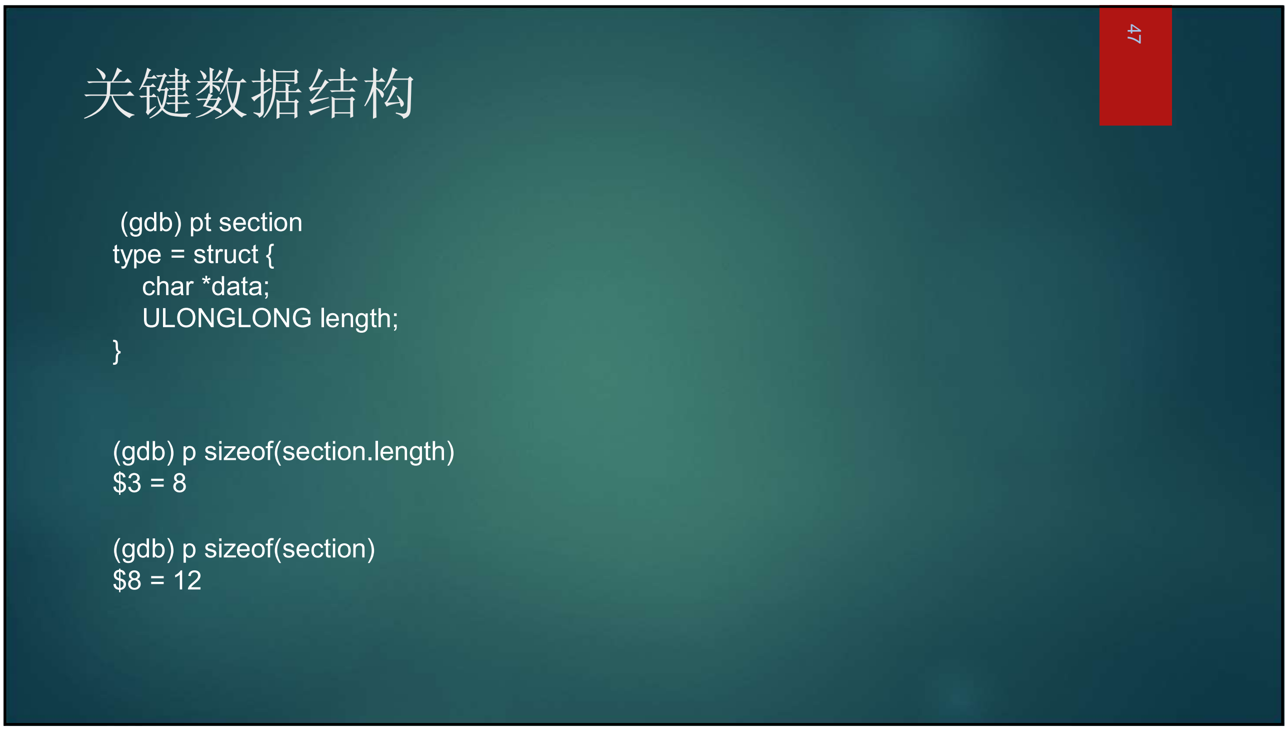


set disassembly-flavor

```
set disassembly-flavor att
set disassembly-flavor intel
show disassembly-flavor
```

http://visualgdb.com/gdbreference/commands/set_disassembly-flavor

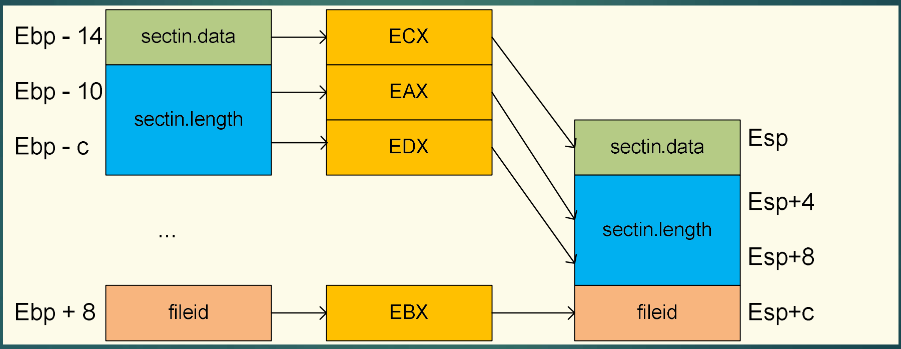
46

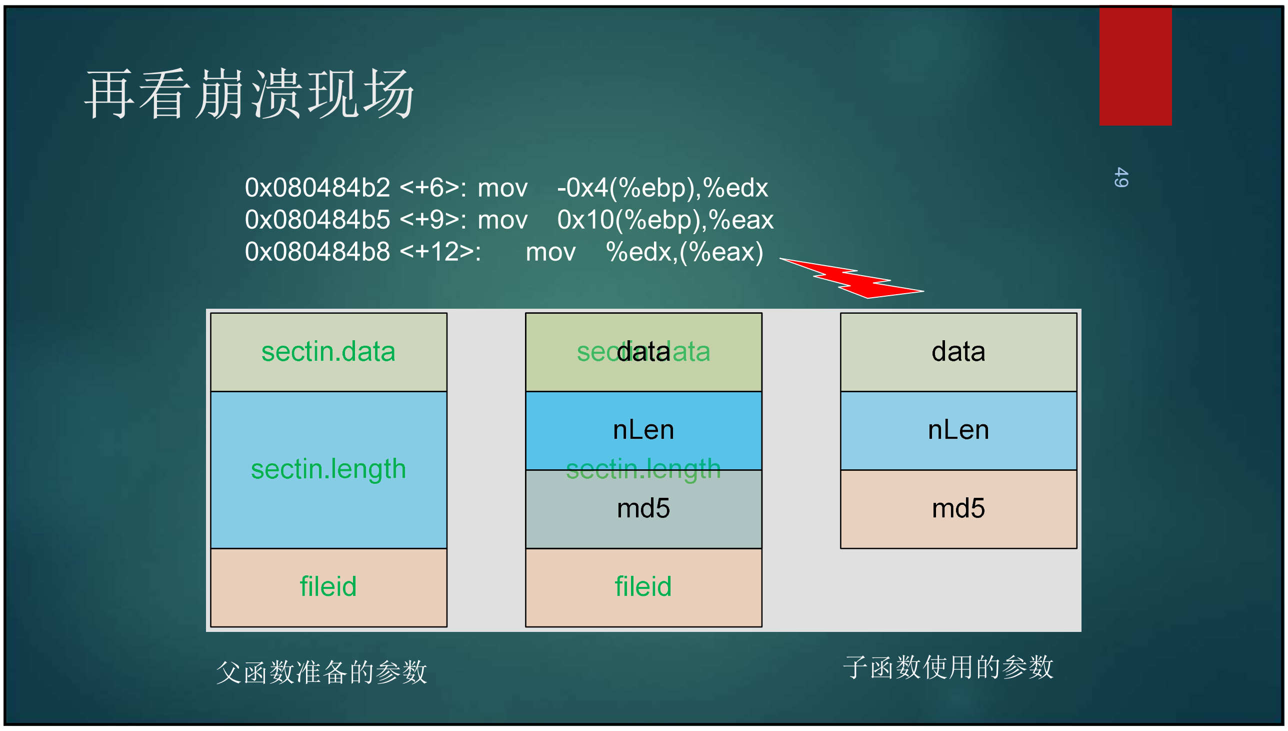


关键数据结构

```
 (gdb) pt section
type = struct {
    char *data;
    ULONGLONG length;
}


(gdb) p sizeof(section.length)
$3 = 8

(gdb) p sizeof(section)
$8 = 12
```

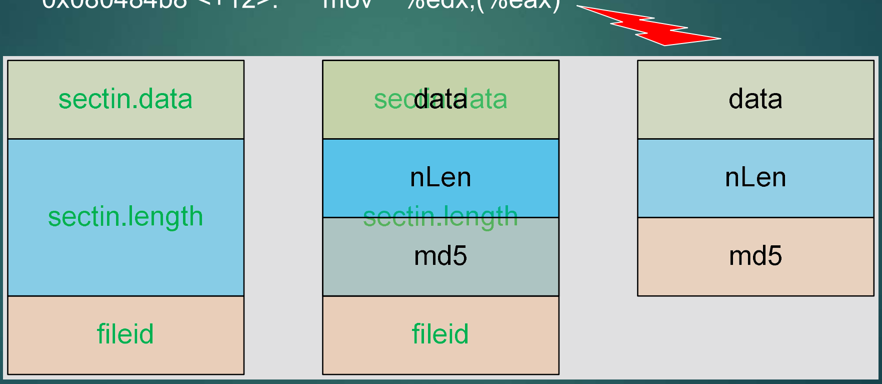
准备参数





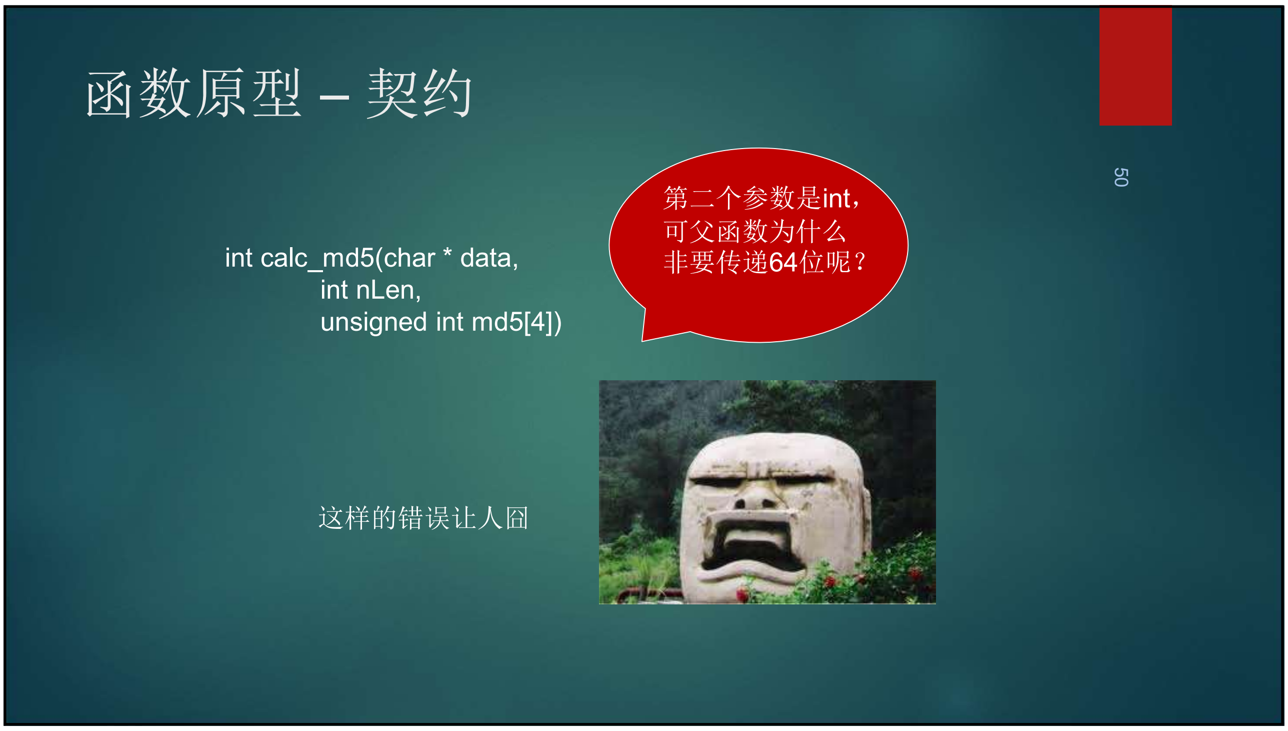
再看崩溃现场

```
0x080484b2 <+6>: mov    -0x4(%ebp),%edx
0x080484b5 <+9>: mov    0x10(%ebp),%eax
0x080484b8 <+12>:      mov    %edx,(%eax)
```



父函数准备的参数

子函数使用的参数

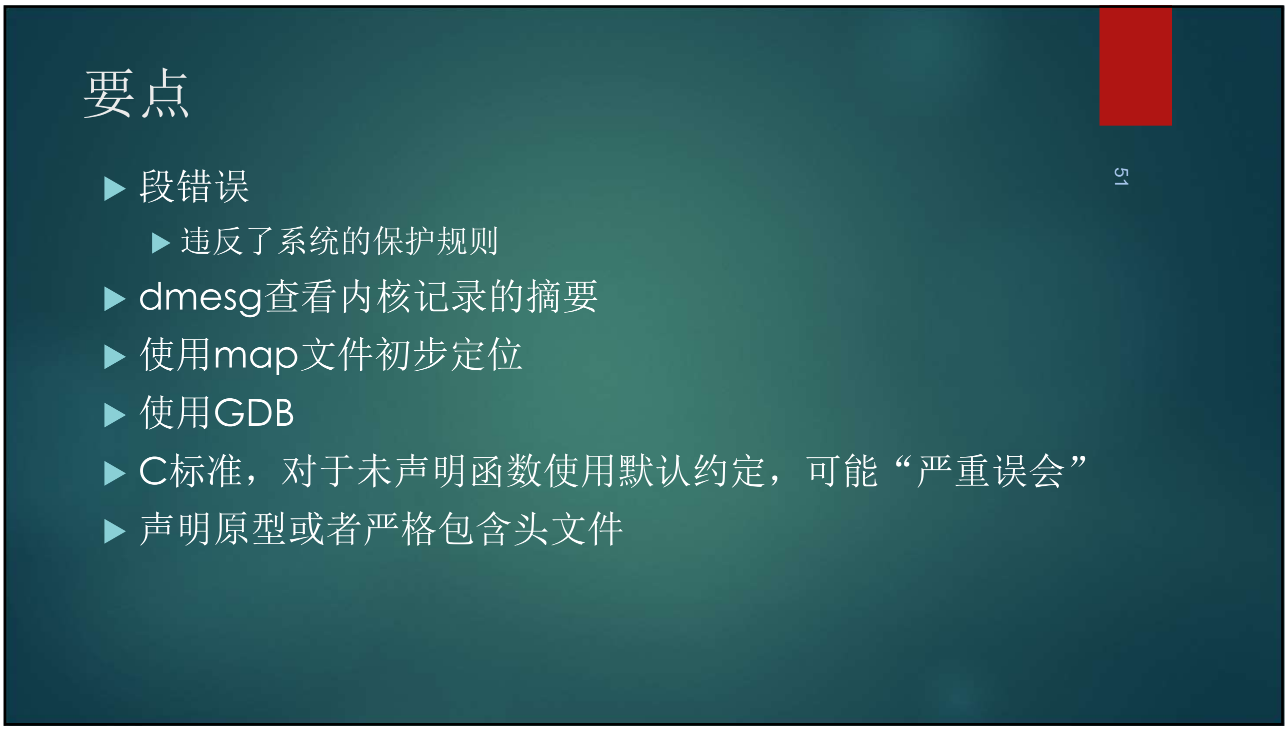


函数原型 – 契约

```
int calc_md5(char * data,
        int nLen,
        unsigned int md5[4])
```

第二个参数是int，
可父函数为什么
非要传递64位呢？

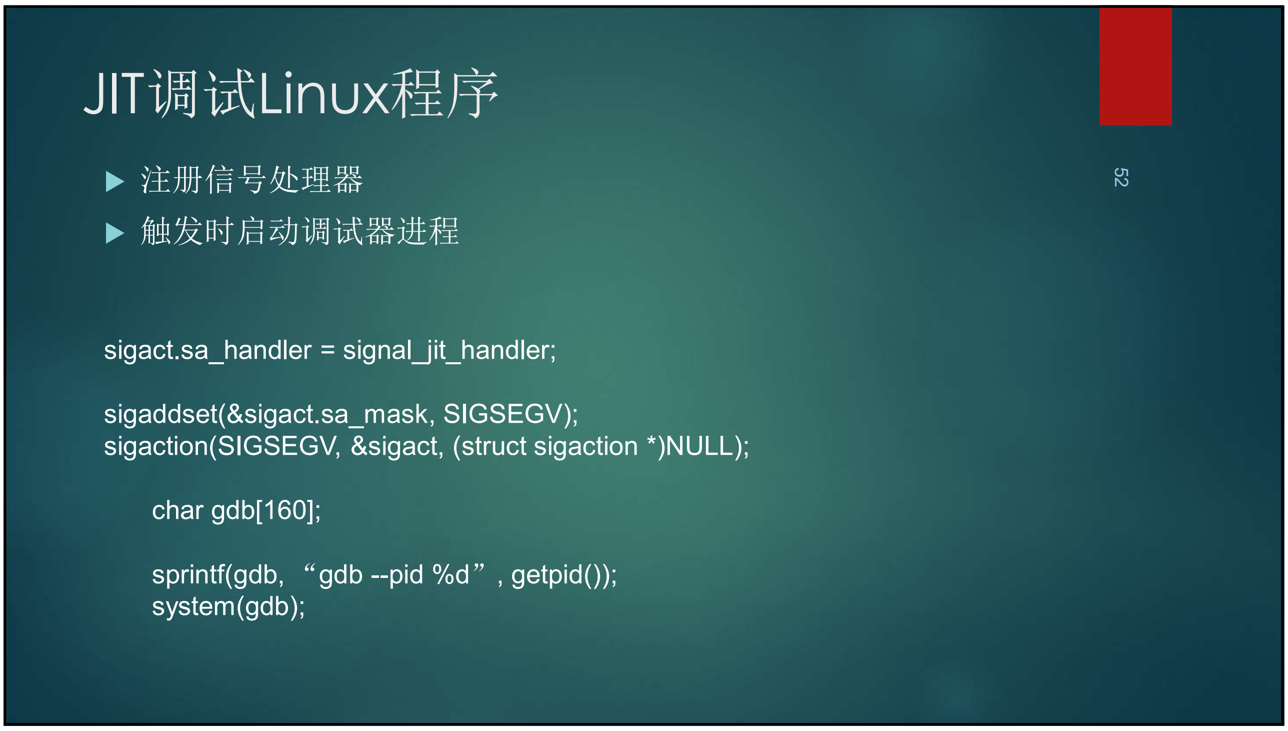
这样的错误让人囧



要点

- 段错误
  - 违反了系统的保护规则
- dmesg查看内核记录的摘要
- 使用map文件初步定位
- 使用GDB
- C标准，对于未声明函数使用默认约定，可能“严重误会”
- 声明原型或者严格包含头文件

51



JIT调试Linux程序

- 注册信号处理器
- 触发时启动调试器进程

```
sigact.sa_handler = signal_jit_handler;

sigaddset(&sigact.sa_mask, SIGSEGV);
sigaction(SIGSEGV, &sigact, (struct sigaction *)NULL);

    char gdb[160];

    sprintf(gdb, “gdb --pid %d” , getpid());
    system(gdb);
```

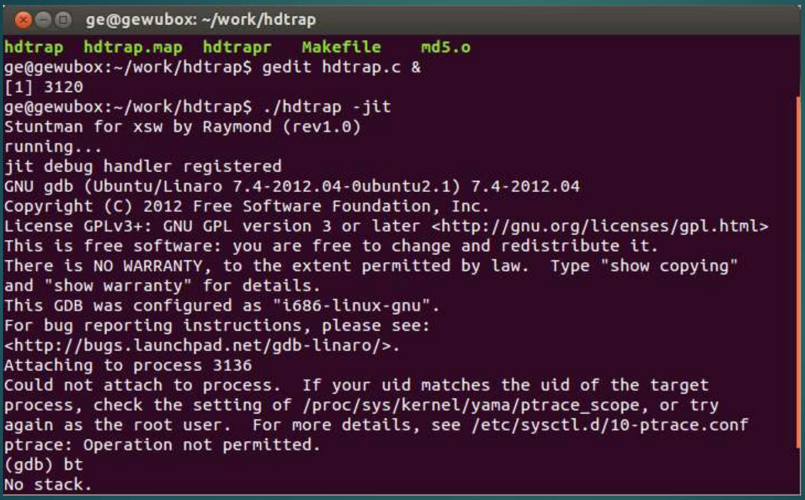
52

```
static void crash_handler(int sig)
{
   int status=0;
   int pid;
   char * gdb_array[]={"gdb", "", NULL};
   char pid_str[40];

   sprintf(pid_str, "--pid=%d%c", getpid(), '\0');
   gdb_array[1]= pid_str;

   pid= fork();

   if (pid < 0) /* error */
      abort();
   else if (pid) /* parent */
   {
      while (1)
         sleep(6000); /* Give GDB time to attach */
      //_exit(1); /* you can skip this line by telling gdb to "return" */
   }
   else /* child */
      execvp("gdb", gdb_array);
}
```

注册信号处理器

```
void register_jit_gdb()
{
   signal(SIGQUIT, crash_handler); /* Normally got from Ctrl-\ */
   signal(SIGILL, crash_handler);
   signal(SIGTRAP, crash_handler);
   signal(SIGABRT, crash_handler);
   signal(SIGFPE, crash_handler);
   signal(SIGBUS, crash_handler);
   signal(SIGSEGV, crash_handler); /* This is the most common crash */
   signal(SIGSYS, crash_handler);
}
```

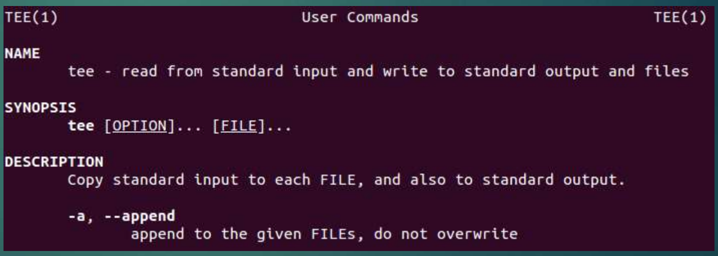
失败



```
hdtrap  hdtrap.map  hdtrapr  Makefile  md5.o
ge@gewubox:~/work/hdtrap$ gedit hdtrap.c &
[1] 3120
ge@gewubox:~/work/hdtrap$ ./hdtrap -jit
Stuntman for xsw by Raymond (rev1.0)
running...
jit debug handler registered
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 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 "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>.
Attaching to process 3136
Could not attach to process.  If your uid matches the uid of the target
process, check the setting of /proc/sys/kernel/yama/ptrace_scope, or try
again as the root user.  For more details, see /etc/sysctl.d/10-ptrace.conf
ptrace: Operation not permitted.
(gdb) bt
No stack.
```

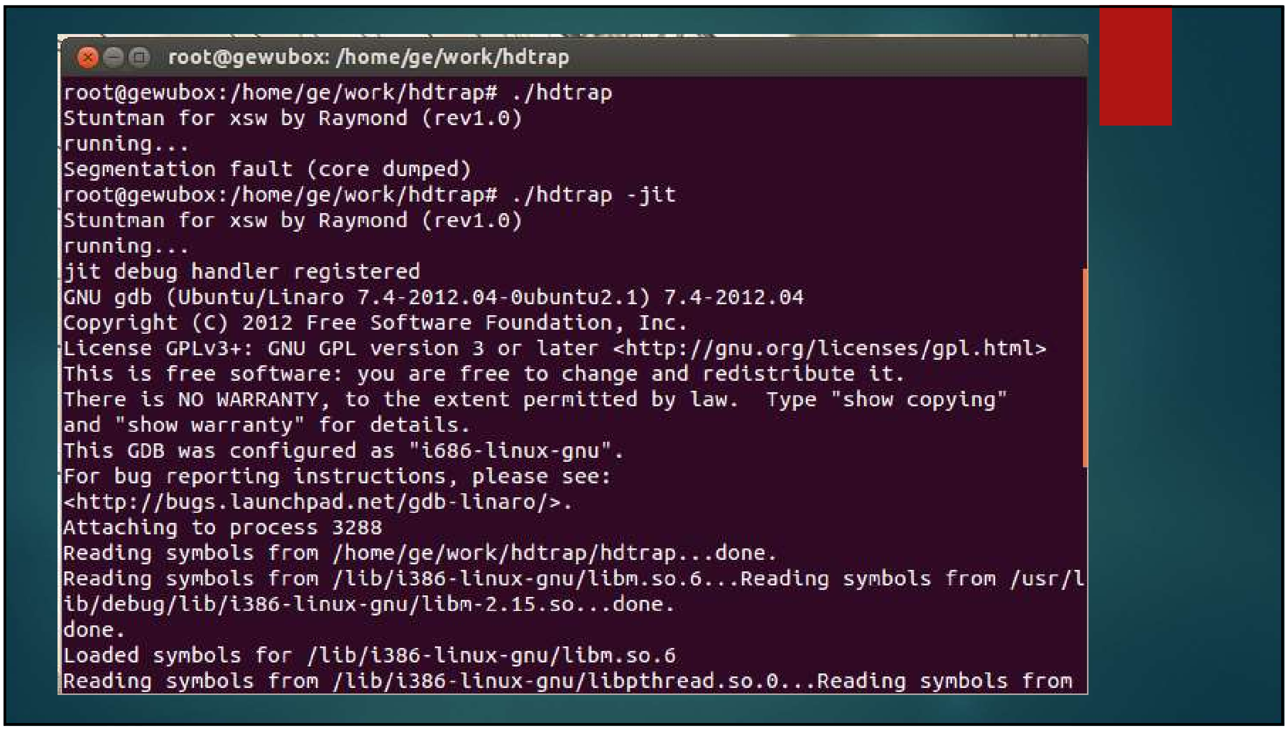
ptrace_scope

- Ubuntu 10.10引入的安全机制
- 默认值为1，防止一个进程分析和修改另一个进程，除非后者是前者的子进程

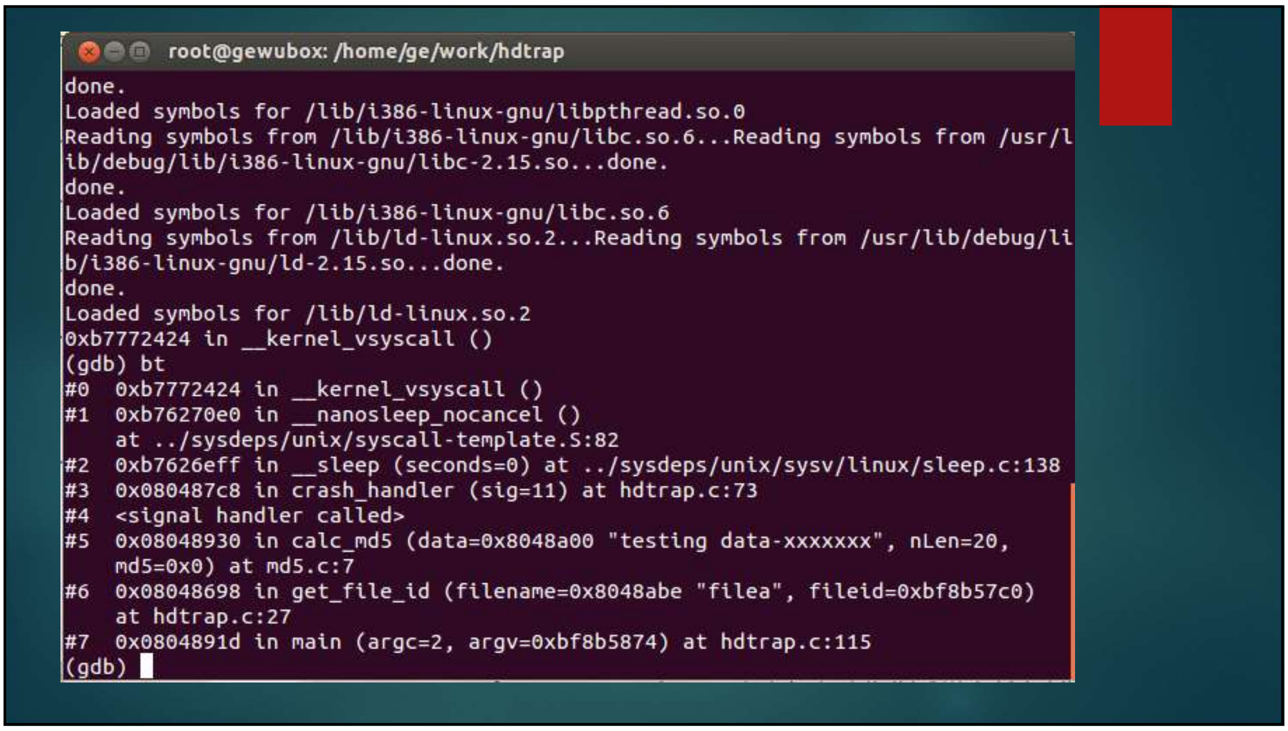


```
TEE(1)                        User Commands                        TEE(1)

NAME
       tee - read from standard input and write to standard output and files

SYNOPSIS
       tee [OPTION]... [FILE]...

DESCRIPTION
       Copy standard input to each FILE, and also to standard output.

       -a, --append
              append to the given FILEs, do not overwrite
```

echo "0" | sudo tee /proc/sys/kernel/yama/ptrace_scope



```
root@gewubox:/home/ge/work/hdtrap
root@gewubox:/home/ge/work/hdtrap# ./hdtrap
Stuntman for xsw by Raymond (rev1.0)
running...
Segmentation fault (core dumped)
root@gewubox:/home/ge/work/hdtrap# ./hdtrap -jit
Stuntman for xsw by Raymond (rev1.0)
running...
jit debug handler registered
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 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 "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>.
Attaching to process 3288
Reading symbols from /home/ge/work/hdtrap/hdtrap...done.
Reading symbols from /lib/i386-linux-gnu/libm.so.6...Reading symbols from /usr/l
ib/debug/lib/i386-linux-gnu/libm-2.15.so...done.
done.
Loaded symbols for /lib/i386-linux-gnu/libm.so.6
Reading symbols from /lib/i386-linux-gnu/libpthread.so.0...Reading symbols from
```

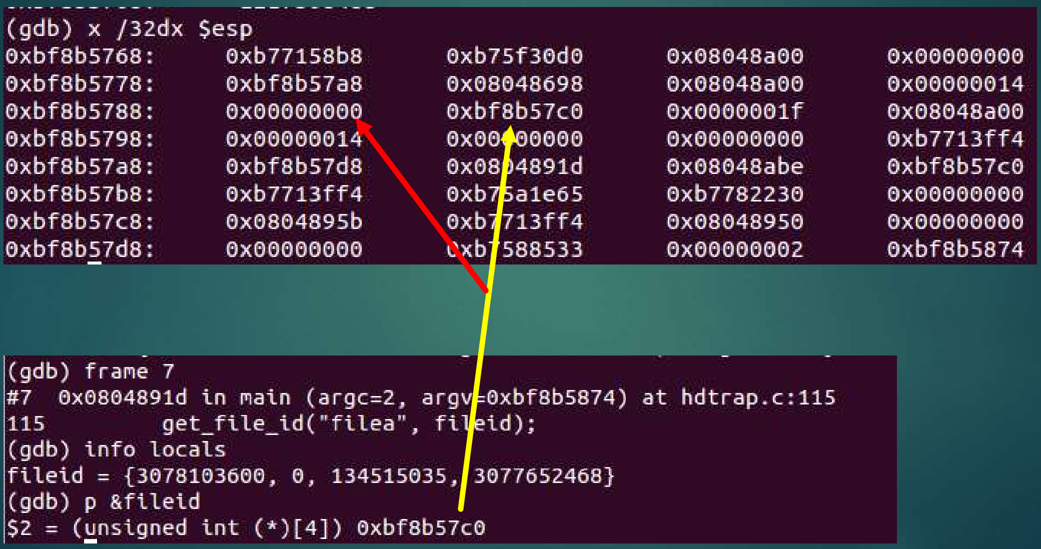


```
root@gewubox:/home/ge/work/hdtrap
done.
Loaded symbols for /lib/i386-linux-gnu/libpthread.so.0
Reading symbols from /lib/i386-linux-gnu/libc.so.6...Reading symbols from /usr/l
ib/debug/lib/i386-linux-gnu/libc-2.15.so...done.
done.
Loaded symbols for /lib/i386-linux-gnu/libc.so.6
Reading symbols from /lib/ld-linux.so.2...Reading symbols from /usr/lib/debug/li
b/i386-linux-gnu/ld-2.15.so...done.
done.
Loaded symbols for /lib/ld-linux.so.2
0xb7772424 in __kernel_vsyscall ()
(gdb) bt
#0  0xb7772424 in __kernel_vsyscall ()
#1  0xb76270e0 in __nanosleep_nocancel ()
    at ../sysdeps/unix/syscall-template.S:82
#2  0xb7626eff in __sleep (seconds=0) at ../sysdeps/unix/sysv/linux/sleep.c:138
#3  0x080487c8 in crash_handler (sig=11) at hdtrap.c:73
#4  <signal handler called>
#5  0x08048930 in calc_md5 (data=0x8048a00 "testing data-xxxxxxx", nLen=20,
    md5=0x0) at md5.c:7
#6  0x08048698 in get_file_id (filename=0x8048abe "filea", fileid=0xbf8b57c0)
    at hdtrap.c:27
#7  0x0804891d in main (argc=2, argv=0xbf8b5874) at hdtrap.c:115
(gdb) 
```

时光倒流

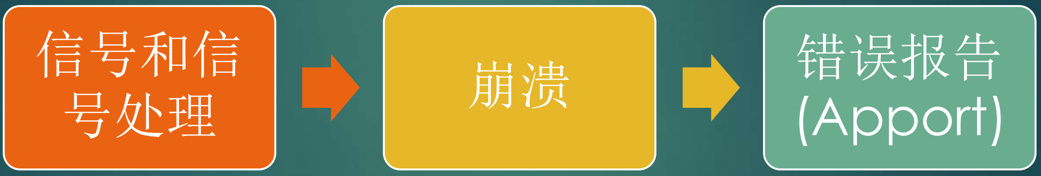
```
(gdb) frame 5
#5  0x08048930 in calc_md5 (data=0x8048a00 "testing data-xxxxxxx", nLen=20,
    md5=0x0) at md5.c:7
7         md5[0] = A;
(gdb) info locals
A = 0
B = 134515200
C = -1218498352
D = -1217308488
(gdb) p md5
$1 = (unsigned int *) 0x0
```

理解错位

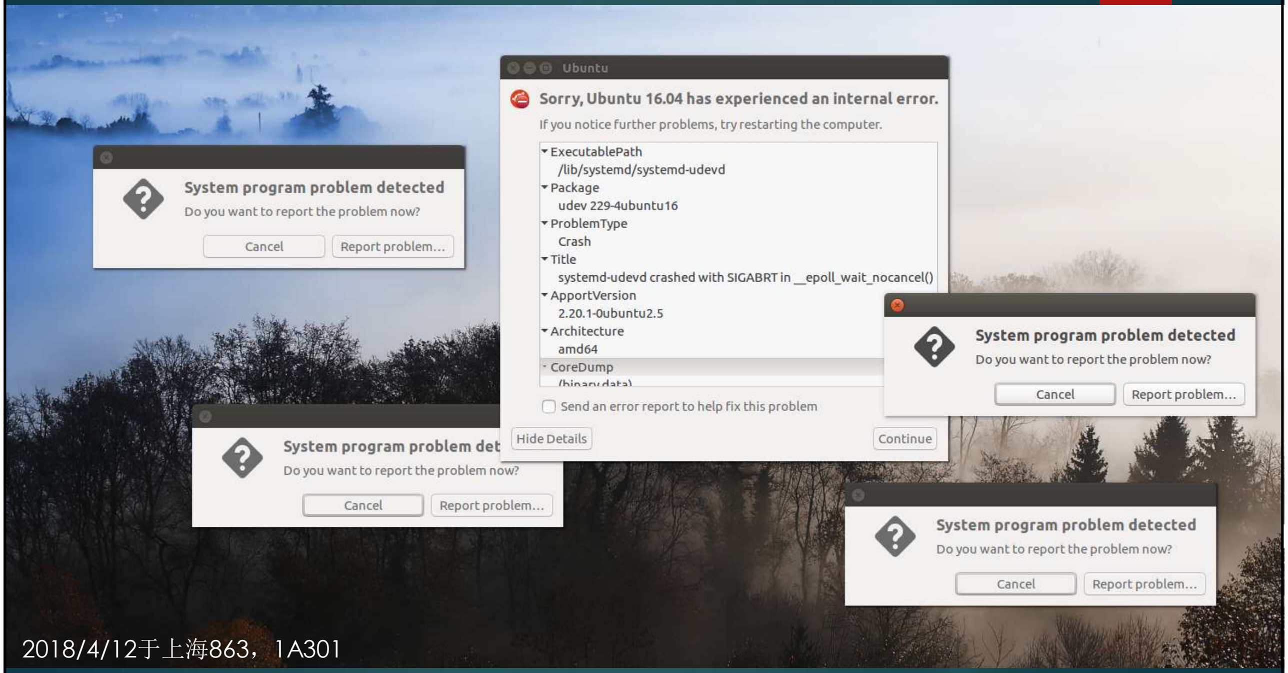


```
(gdb) x /32dx $esp
0xbf8b5768:     0xb77158b8      0xb75f30d0      0x08048a00      0x00000000
0xbf8b5778:     0xbf8b57a8      0x08048698      0x08048a00      0x00000014
0xbf8b5788:     0x00000000      0xbf8b57c0      0x0000001f      0x08048a00
0xbf8b5798:     0x00000014      0x00000000      0x00000000      0xb7713ff4
0xbf8b57a8:     0xbf8b57d8      0x0804891d      0x08048abe      0xbf8b57c0
0xbf8b57b8:     0xb7713ff4      0xb75a1e65      0xb7782230      0x00000000
0xbf8b57c8:     0x0804895b      0xb7713ff4      0x08048950      0x00000000
0xbf8b57d8:     0x00000000      0xb7588533      0x00000002      0xbf8b5874
```

```
(gdb) frame 7
#7  0x0804891d in main (argc=2, argv=0xbf8b5874) at hdtrap.c:115
115           get_file_id("filea", fileid);
(gdb) info locals
fileid = {3078103600, 0, 134515035, 3077652468}
(gdb) p &fileid
$2 = (unsigned int (*)[4]) 0xbf8b57c0
```



61



Ubuntu

Sorry, Ubuntu 16.04 has experienced an internal error.

If you notice further problems, try restarting the computer.

```
▾ExecutablePath
   /lib/systemd/systemd-udevd
▾Package
   udev 229-4ubuntu16
▾ProblemType
   Crash
▾Title
   systemd-udevd crashed with SIGABRT in __epoll_wait_nocancel()
▾ApportVersion
   2.20.1-0ubuntu2.5
▾Architecture
   amd64
▸CoreDump
   (binary data)
```

☐ Send an error report to help fix this problem

Hide Details　Continue

System program problem detected

Do you want to report the problem now?

Cancel　Report problem...

System program problem detected

Do you want to report the problem now?

Cancel　Report problem...

System program problem det

Do you want to report the problem now?

Cancel　Report problem...

System program problem detected

Do you want to report the problem now?

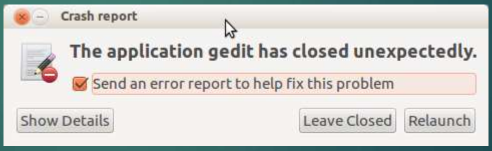
Cancel　Report problem...

2018/4/12于上海863，1A301

62

Apport

- https://wiki.ubuntu.com/Apport
- 自动介入
  - 崩溃
  - 其它可以自动检测到的错误情况，比如更新失败
- 收集错误信息
- 呈现界面


Crash report

The application gedit has closed unexpectedly.

☑ Send an error report to help fix this problem

63

介入方式

- /proc/sys/kernel/core_pattern

```
gedu@gedu-VirtualBox:~$ cat /proc/sys/kernel/core_pattern
|/usr/share/apport/apport %p %s %c %P
```

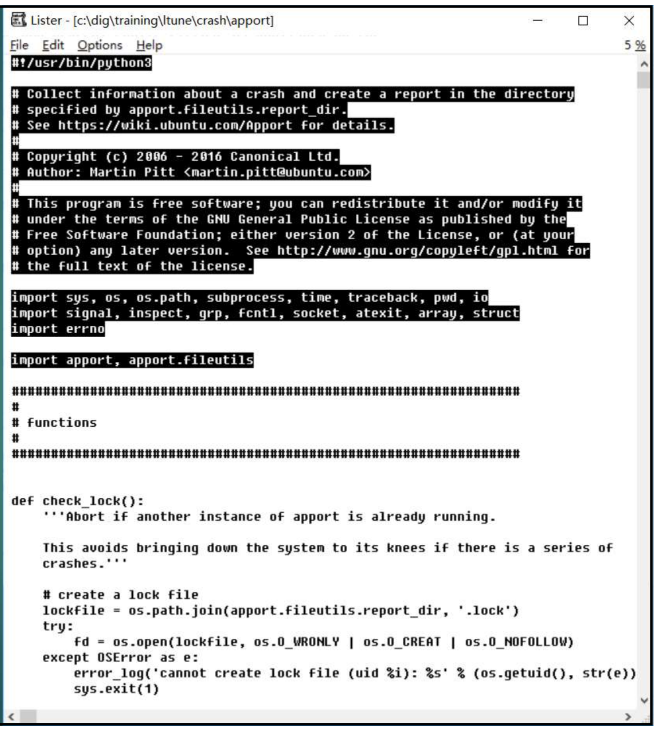
- 上面的 | 代表用户态程序

- 对于Python程序，会安装/etc/python*/sitecustomize.py来调用apport

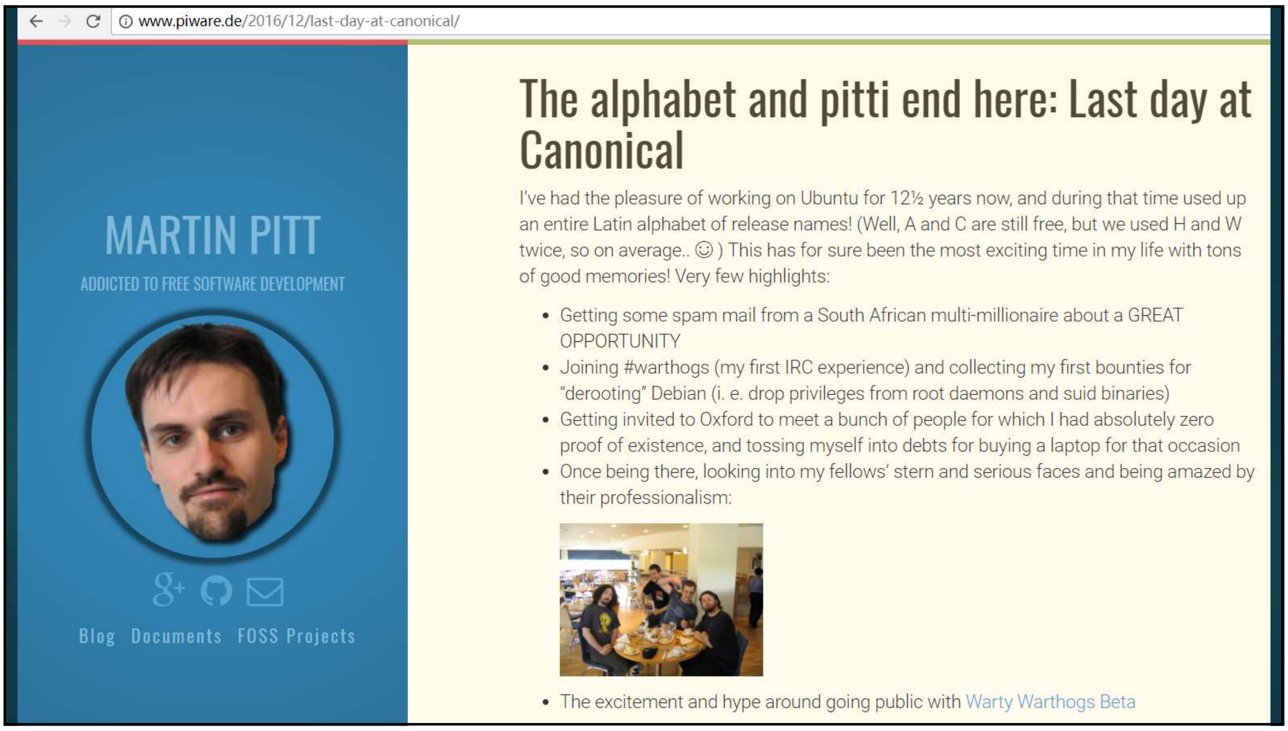
64

Apport脚本

- 用Python编写
- 作者Martin Pitt



```
#!/usr/bin/python3

# Collect information about a crash and create a report in the directory
# specified by apport.fileutils.report_dir.
# See https://wiki.ubuntu.com/Apport for details.
#
# Copyright (c) 2006 - 2016 Canonical Ltd.
# Author: Martin Pitt <martin.pitt@ubuntu.com>
#
# This program is free software; you can redistribute it and/or modify it
# under the terms of the GNU General Public License as published by the
# Free Software Foundation; either version 2 of the License, or (at your
# option) any later version.  See http://www.gnu.org/copyleft/gpl.html for
# the full text of the license.

import sys, os, os.path, subprocess, time, traceback, pwd, io
import signal, inspect, grp, fcntl, socket, atexit, array, struct
import errno

import apport, apport.fileutils

#################################################################
#
# functions
#
#################################################################


def check_lock():
    '''Abort if another instance of apport is already running.

    This avoids bringing down the system to its knees if there is a series of
    crashes.'''

    # create a lock file
    lockfile = os.path.join(apport.fileutils.report_dir, '.lock')
    try:
        fd = os.open(lockfile, os.O_WRONLY | os.O_CREAT | os.O_NOFOLLOW)
    except OSError as e:
        error_log('cannot create lock file (uid %i): %s' % (os.getuid(), str(e))
        sys.exit(1)
```

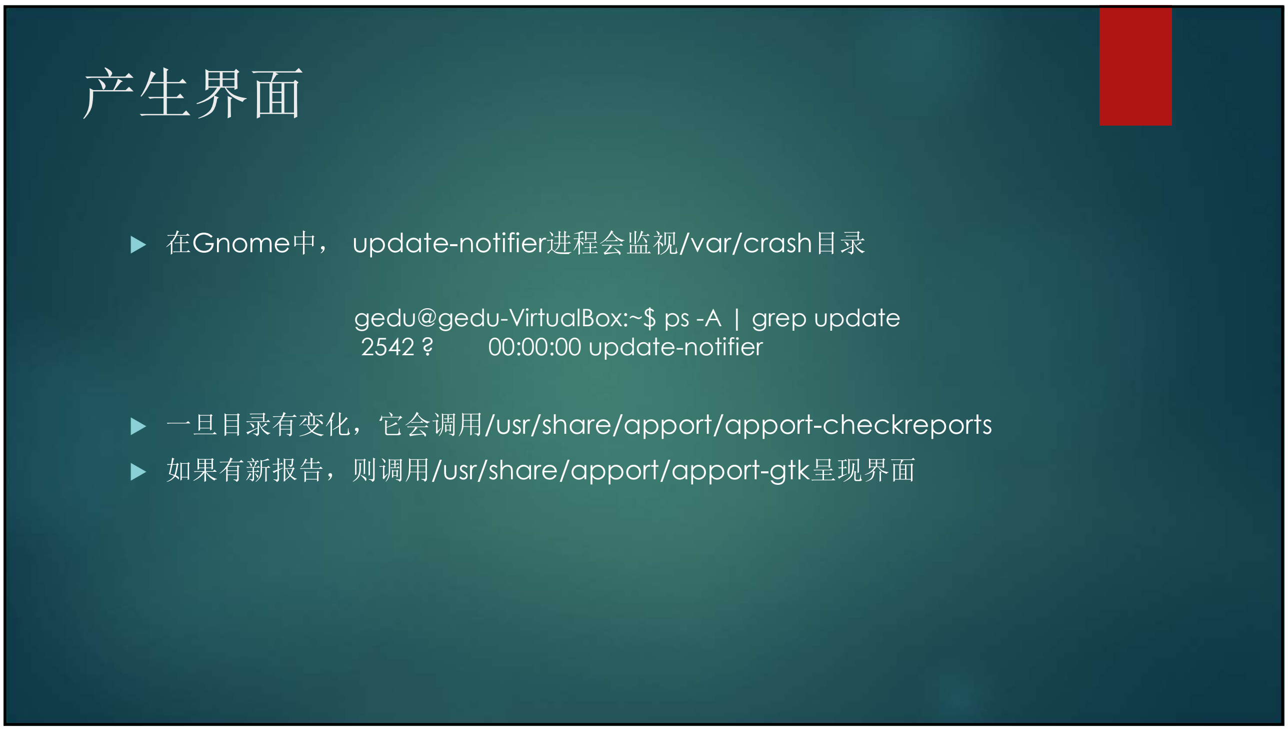

www.piware.de/2016/12/last-day-at-canonical/
MARTIN PITT
ADDICTED TO FREE SOFTWARE DEVELOPMENT
Blog Documents FOSS Projects

The alphabet and pitti end here: Last day at Canonical

I've had the pleasure of working on Ubuntu for 12½ years now, and during that time used up an entire Latin alphabet of release names! (Well, A and C are still free, but we used H and W twice, so on average.. ☺) This has for sure been the most exciting time in my life with tons of good memories! Very few highlights:

- Getting some spam mail from a South African multi-millionaire about a GREAT OPPORTUNITY
- Joining #warthogs (my first IRC experience) and collecting my first bounties for "derooting" Debian (i. e. drop privileges from root daemons and suid binaries)
- Getting invited to Oxford to meet a bunch of people for which I had absolutely zero proof of existence, and tossing myself into debts for buying a laptop for that occasion
- Once being there, looking into my fellows' stern and serious faces and being amazed by their professionalism:

- The excitement and hype around going public with Warty Warthogs Beta



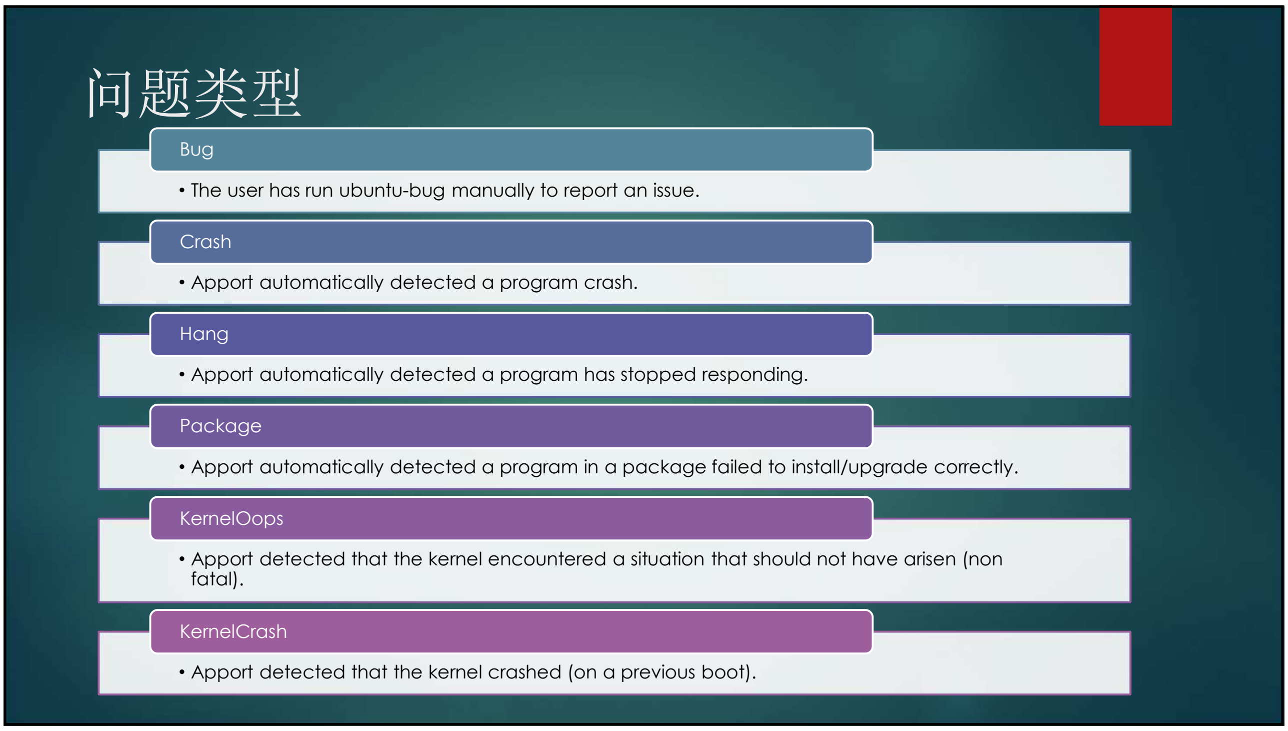
产生界面

- 在Gnome中， update-notifier进程会监视/var/crash目录

```
gedu@gedu-VirtualBox:~$ ps -A | grep update
 2542 ?        00:00:00 update-notifier
```

- 一旦目录有变化，它会调用/usr/share/apport/apport-checkreports
- 如果有新报告，则调用/usr/share/apport/apport-gtk呈现界面

67



问题类型

Bug
- The user has run ubuntu-bug manually to report an issue.

Crash
- Apport automatically detected a program crash.

Hang
- Apport automatically detected a program has stopped responding.

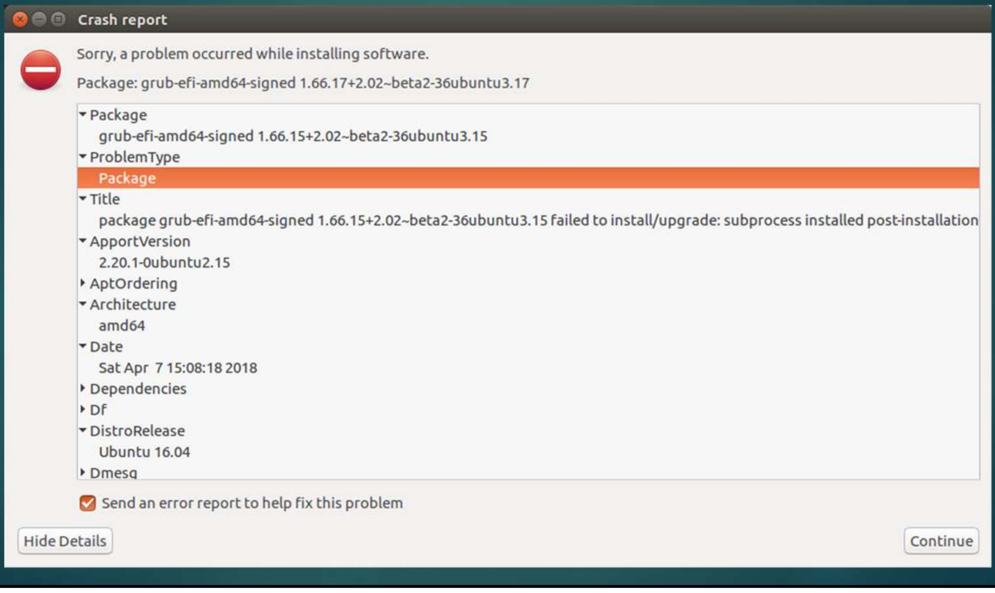
Package
- Apport automatically detected a program in a package failed to install/upgrade correctly.

KernelOops
- Apport detected that the kernel encountered a situation that should not have arisen (non fatal).

KernelCrash
- Apport detected that the kernel crashed (on a previous boot).

68

包类型的故障



```
Sorry, a problem occurred while installing software.
Package: grub-efi-amd64-signed 1.66.17+2.02~beta2-36ubuntu3.17

▾ Package
    grub-efi-amd64-signed 1.66.15+2.02~beta2-36ubuntu3.15
▾ ProblemType
    Package
▾ Title
    package grub-efi-amd64-signed 1.66.15+2.02~beta2-36ubuntu3.15 failed to install/upgrade: subprocess installed post-installation
▾ ApportVersion
    2.20.1-0ubuntu2.15
▸ AptOrdering
▾ Architecture
    amd64
▾ Date
    Sat Apr  7 15:08:18 2018
▸ Dependencies
▸ Df
▾ DistroRelease
    Ubuntu 16.04
▸ Dmesg

☑ Send an error report to help fix this problem
```

报告格式

```
DistroRelease: Ubuntu 12.04
ExecutablePath: /usr/bin/gcalctool
Package: gcalctool 5.8.24-0ubuntu2
ProcCmdline: gcalctool
ProcEnviron:
 SHELL=/bin/bash
 PATH=/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11:/usr/games
 LANG=de_DE.UTF-8
StackTrace:
 [...]
 #0  0x00002ae577bb37bf in poll () from /lib/libc.so.6
 No symbol table info available.
 #1  0x00002ae57786991e in g_main_context_check () from /usr/lib64/libglib-2.0.so.0
 No symbol table info available.
 [...]
CoreDump: base64
 eJzsXQmcFMXV7+XGA0dBREVoDxSPXQYEB...
```

针对段错误的深层信息

SegvAnalysis: when examining a Segmentation Fault (signal 11), Apport attempts to review the exact machine instruction that caused the fault, and checks the program counter, source, and destination addresses, looking for any virtual memory address (VMA) that is outside an allocated range (as reported in the ProcMaps attachment).

SegvReason: a VMA can be read from, written to, or executed. On a SegFault, one of these 3 CPU actions has taken place at a given VMA that either not allocated, or lacks permissions to perform the action. For example:

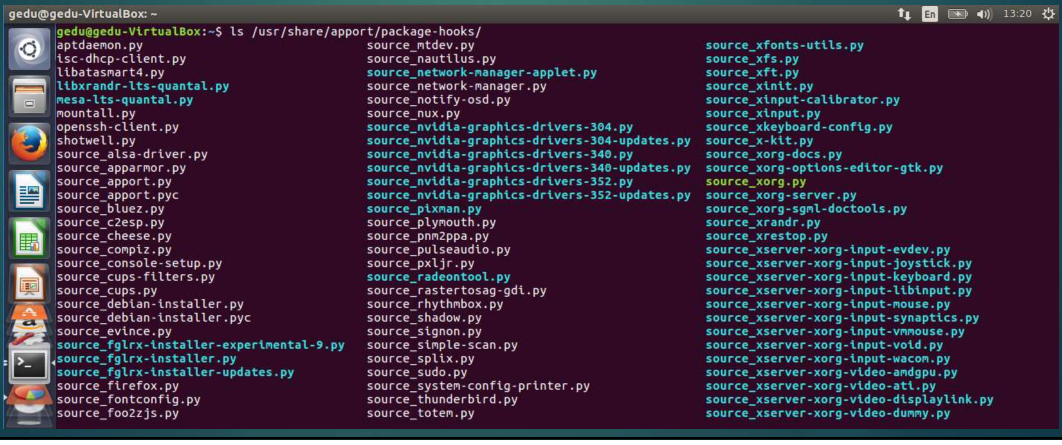
SegvReason: reading NULL VMA would mean that a NULL pointer was most likely dereferenced while reading a value.

SegvReason: writing unknown VMA would mean that something was attempting to write to the destination of a pointer aimed outside of allocated memory. (This is sometimes a security issue.)

SegvReason: executing writable VMA [stack] would mean that something was causing code on the stack to be executed, but the stack (correctly) lacked execute permissions. (This is almost always a security issue.)

收集更多信息 /usr/share/apport/package-hooks

- 在/usr/share/apport/目录中放入<binarypackagename>.py或者source_<sourcepackagename>.py
- https://wiki.ubuntu.com/Apport/DeveloperHowTo



gedu@gedu-VirtualBox: ~

```
gedu@gedu-VirtualBox:~$ ls /usr/share/apport/package-hooks/
aptdaemon.py                              source_mtdev.py                                source_xfonts-utils.py
isc-dhcp-client.py                        source_nautilus.py                             source_xfs.py
libatasmart4.py                           source_network-manager-applet.py               source_xft.py
libxrandr-lts-quantal.py                  source_network-manager.py                      source_xinit.py
mesa-lts-quantal.py                       source_notify-osd.py                           source_xinput-calibrator.py
mountall.py                               source_nux.py                                  source_xinput.py
openssh-client.py                         source_nvidia-graphics-drivers-304.py          source_xkeyboard-config.py
shotwell.py                               source_nvidia-graphics-drivers-304-updates.py  source_x-kit.py
source_alsa-driver.py                     source_nvidia-graphics-drivers-340.py          source_xorg-docs.py
source_apparmor.py                        source_nvidia-graphics-drivers-340-updates.py  source_xorg-options-editor-gtk.py
source_apport.py                          source_nvidia-graphics-drivers-352.py          source_xorg.py
source_apport.pyc                         source_nvidia-graphics-drivers-352-updates.py  source_xorg-server.py
source_bluez.py                           source_pixman.py                               source_xorg-sgml-doctools.py
source_c2esp.py                           source_plymouth.py                             source_xrandr.py
source_cheese.py                          source_pnm2ppa.py                              source_xrestop.py
source_compiz.py                          source_pulseaudio.py                           source_xserver-xorg-input-evdev.py
source_console-setup.py                   source_pxljr.py                                source_xserver-xorg-input-joystick.py
source_cups-filters.py                    source_radeontool.py                           source_xserver-xorg-input-keyboard.py
source_cups.py                            source_rastertosag-gdi.py                      source_xserver-xorg-input-libinput.py
source_debian-installer.py                source_rhythmbox.py                            source_xserver-xorg-input-mouse.py
source_debian-installer.pyc               source_shadow.py                               source_xserver-xorg-input-synaptics.py
source_evince.py                          source_signon.py                               source_xserver-xorg-input-vmmouse.py
source_fglrx-installer-experimental-9.py  source_simple-scan.py                          source_xserver-xorg-input-void.py
source_fglrx-installer.py                 source_splix.py                                source_xserver-xorg-input-wacom.py
source_fglrx-installer-updates.py         source_sudo.py                                 source_xserver-xorg-video-amdgpu.py
source_firefox.py                         source_system-config-printer.py                source_xserver-xorg-video-ati.py
source_fontconfig.py                      source_thunderbird.py                          source_xserver-xorg-video-displaylink.py
source_foo2zjs.py                         source_totem.py                                source_xserver-xorg-video-dummy.py
```

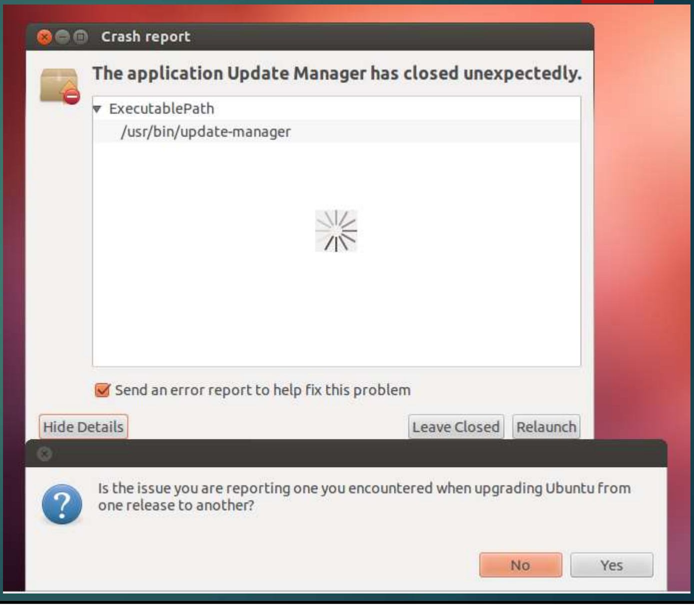
实例

```
gedu@gedu-VirtualBox:~$ cat /usr/share/apport/package-hooks/udev.py
'''apport package hook for udev

(c) 2009 Canonical Ltd.
Author: Martin Pitt <martin.pitt@ubuntu.com>
'''

import os
import apport.hookutils

def add_info(report):
    apport.hookutils.attach_hardware(report)

    user_rules = []
    for f in os.listdir('/etc/udev/rules.d'):
        if not f.startswith('70-persistent-') and f != 'README':
            user_rules.append(f)

    if user_rules:
        report['CustomUdevRuleFiles'] = ' '.join(user_rules)
```

可以提交互性问题

- 回答Yes和No

```
def add_info(report, ui):

    attach_file = False

    if ui and ui.yesno('Attach
some file?') == True:
        attach_file = True

    if attach_file == True:
        # user wants file to be
attached
```


Crash report

The application Update Manager has closed unexpectedly.

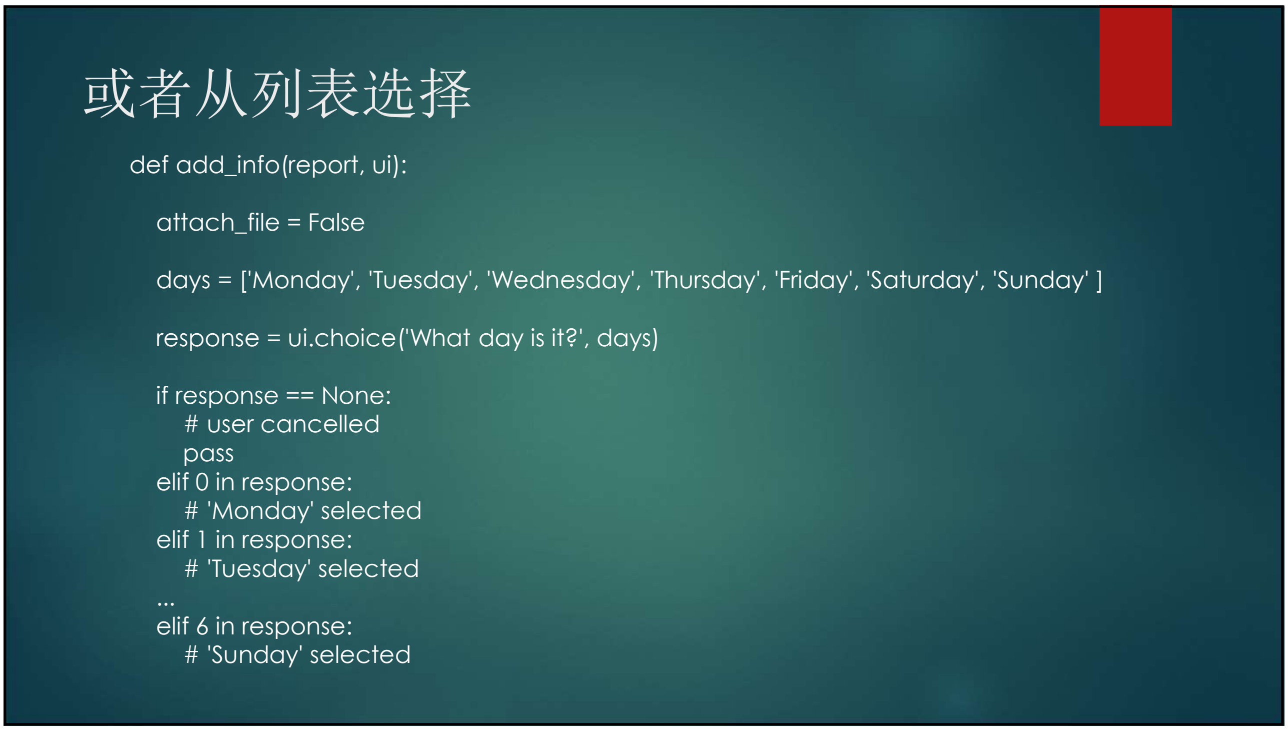
▾ ExecutablePath
/usr/bin/update-manager

☑ Send an error report to help fix this problem

Hide Details | Leave Closed | Relaunch

Is the issue you are reporting one you encountered when upgrading Ubuntu from one release to another?

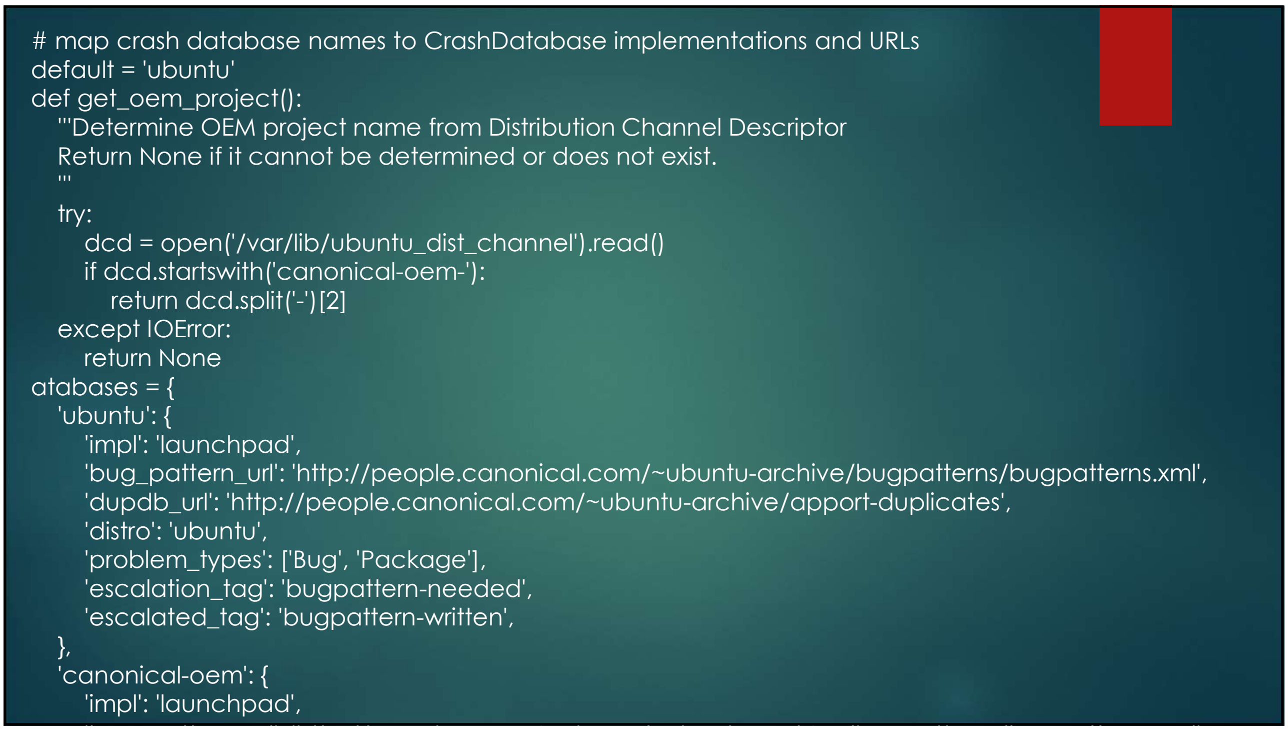
No | Yes



或者从列表选择

```
def add_info(report, ui):

    attach_file = False

    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday' ]

    response = ui.choice('What day is it?', days)

    if response == None:
        # user cancelled
        pass
    elif 0 in response:
        # 'Monday' selected
    elif 1 in response:
        # 'Tuesday' selected
    ...
    elif 6 in response:
        # 'Sunday' selected
```

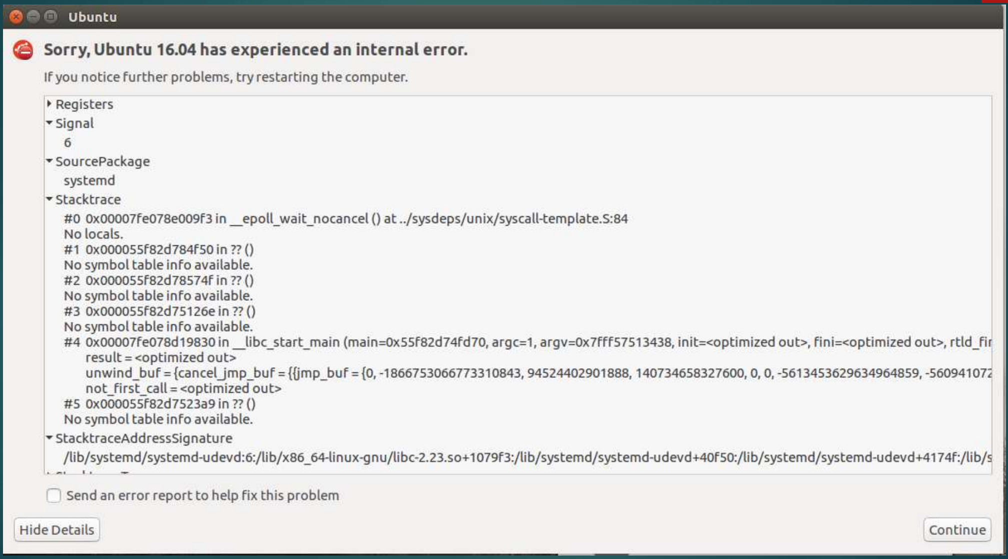
75



```
# map crash database names to CrashDatabase implementations and URLs
default = 'ubuntu'
def get_oem_project():
    '''Determine OEM project name from Distribution Channel Descriptor
    Return None if it cannot be determined or does not exist.
    '''
    try:
        dcd = open('/var/lib/ubuntu_dist_channel').read()
        if dcd.startswith('canonical-oem-'):
            return dcd.split('-')[2]
    except IOError:
        return None
atabases = {
    'ubuntu': {
        'impl': 'launchpad',
        'bug_pattern_url': 'http://people.canonical.com/~ubuntu-archive/bugpatterns/bugpatterns.xml',
        'dupdb_url': 'http://people.canonical.com/~ubuntu-archive/apport-duplicates',
        'distro': 'ubuntu',
        'problem_types': ['Bug', 'Package'],
        'escalation_tag': 'bugpattern-needed',
        'escalated_tag': 'bugpattern-written',
    },
    'canonical-oem': {
        'impl': 'launchpad',
```

76

案例: udevd进程崩溃



Sorry, Ubuntu 16.04 has experienced an internal error.

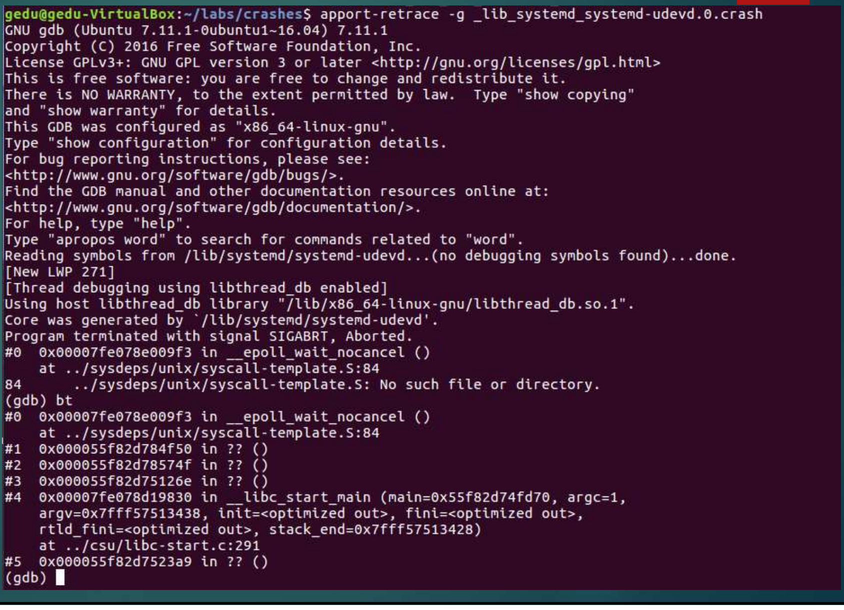
If you notice further problems, try restarting the computer.

```
▸ Registers
▾ Signal
    6
▾ SourcePackage
    systemd
▾ Stacktrace
    #0 0x00007fe078e009f3 in __epoll_wait_nocancel () at ../sysdeps/unix/syscall-template.S:84
    No locals.
    #1 0x000055f82d784f50 in ?? ()
    No symbol table info available.
    #2 0x000055f82d78574f in ?? ()
    No symbol table info available.
    #3 0x000055f82d75126e in ?? ()
    No symbol table info available.
    #4 0x00007fe078d19830 in __libc_start_main (main=0x55f82d74fd70, argc=1, argv=0x7fff57513438, init=<optimized out>, fini=<optimized out>, rtld_fir
        result = <optimized out>
        unwind_buf = {cancel_jmp_buf = {{jmp_buf = {0, -1866753066773310843, 94524402901888, 140734658327600, 0, 0, -5613453629634964859, -56094107z
        not_first_call = <optimized out>
    #5 0x000055f82d7523a9 in ?? ()
    No symbol table info available.
▾ StacktraceAddressSignature
    /lib/systemd/systemd-udevd:6:/lib/x86_64-linux-gnu/libc-2.23.so+1079f3:/lib/systemd/systemd-udevd+40f50:/lib/systemd/systemd-udevd+4174f:/lib/s
```

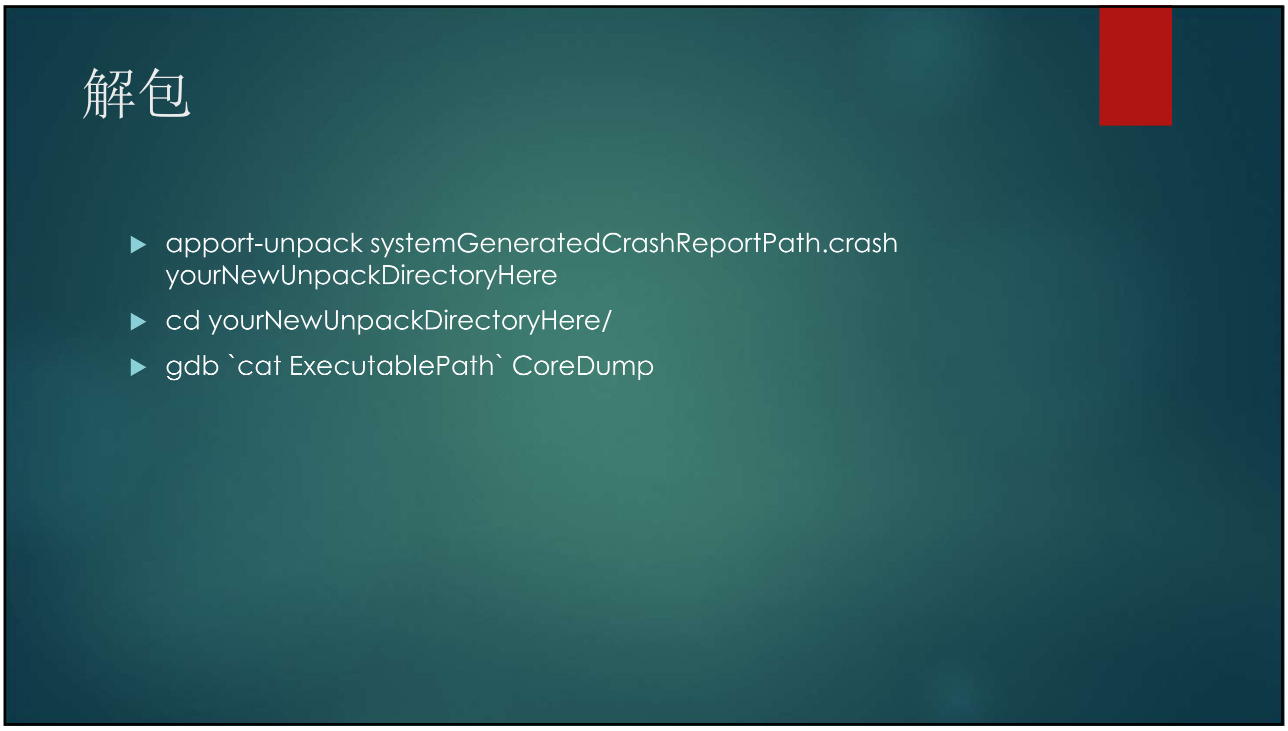
☐ Send an error report to help fix this problem

apport-retrace

- 需要安装
- apport-retrace -g CRASHFILE.crash



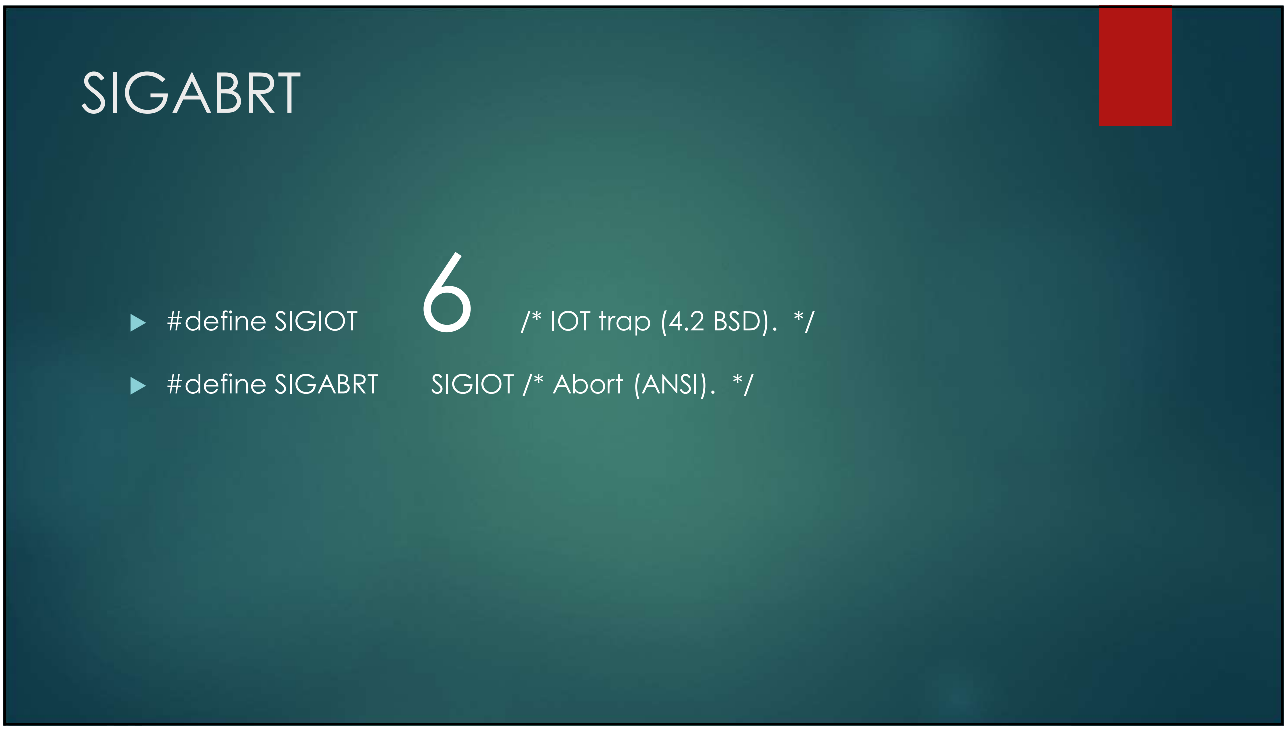
```
gedu@gedu-VirtualBox:~/labs/crashes$ apport-retrace -g _lib_systemd_systemd-udevd.0.crash
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free 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 "x86_64-linux-gnu".
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 /lib/systemd/systemd-udevd...(no debugging symbols found)...done.
[New LWP 271]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/lib/systemd/systemd-udevd'.
Program terminated with signal SIGABRT, Aborted.
#0  0x00007fe078e009f3 in __epoll_wait_nocancel ()
    at ../sysdeps/unix/syscall-template.S:84
84      ../sysdeps/unix/syscall-template.S: No such file or directory.
(gdb) bt
#0  0x00007fe078e009f3 in __epoll_wait_nocancel ()
    at ../sysdeps/unix/syscall-template.S:84
#1  0x000055f82d784f50 in ?? ()
#2  0x000055f82d78574f in ?? ()
#3  0x000055f82d75126e in ?? ()
#4  0x00007fe078d19830 in __libc_start_main (main=0x55f82d74fd70, argc=1,
    argv=0x7fff57513438, init=<optimized out>, fini=<optimized out>,
    rtld_fini=<optimized out>, stack_end=0x7fff57513428)
    at ../csu/libc-start.c:291
#5  0x000055f82d7523a9 in ?? ()
(gdb) 
```



解包

- apport-unpack systemGeneratedCrashReportPath.crash yourNewUnpackDirectoryHere
- cd yourNewUnpackDirectoryHere/
- gdb `cat ExecutablePath` CoreDump

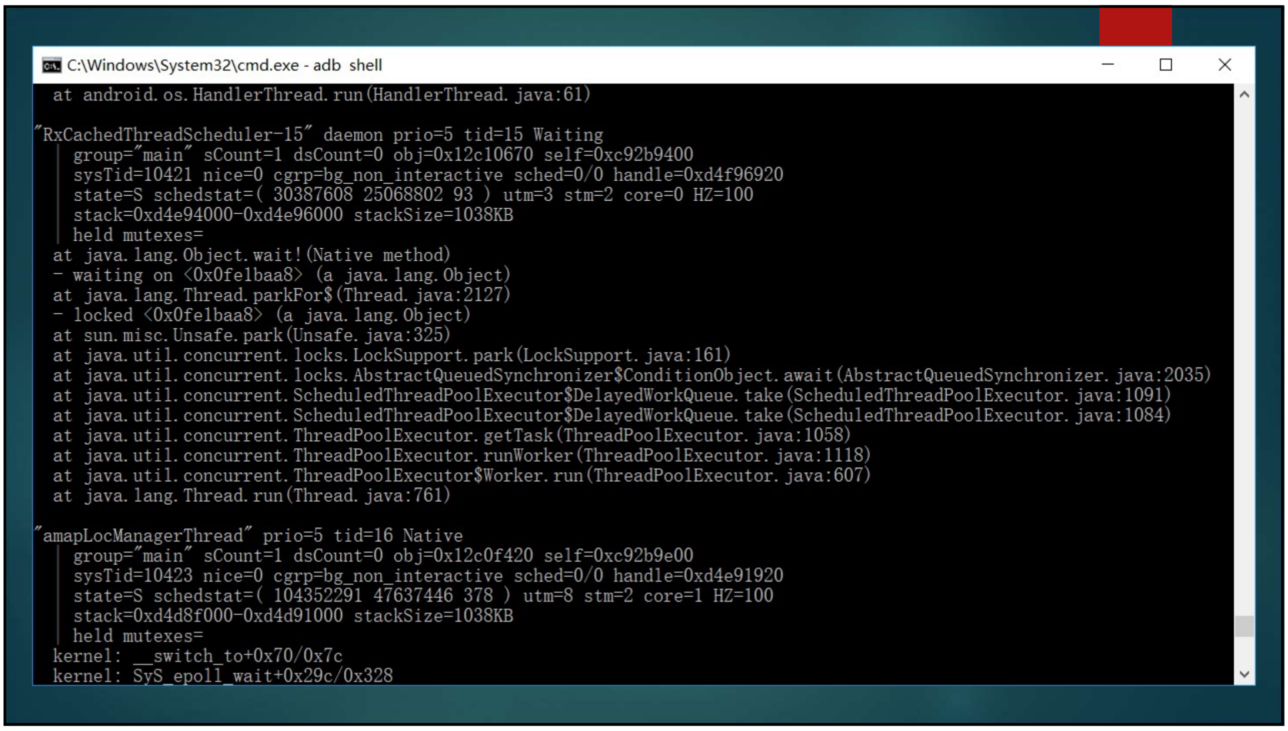
79



SIGABRT

- #define SIGIOT 6 /* IOT trap (4.2 BSD). */
- #define SIGABRT SIGIOT /* Abort (ANSI). */

80

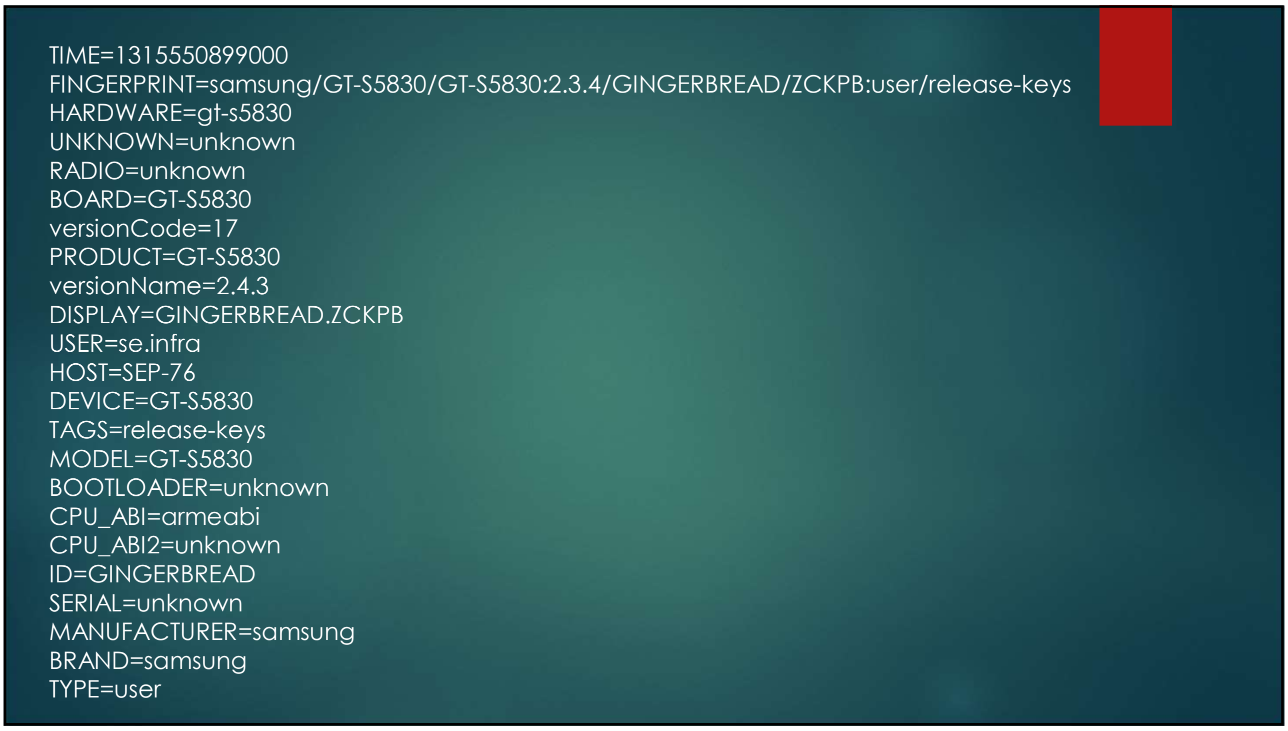


C:\Windows\System32\cmd.exe - adb shell

```
  at android.os.HandlerThread.run(HandlerThread.java:61)

"RxCachedThreadScheduler-15" daemon prio=5 tid=15 Waiting
  | group="main" sCount=1 dsCount=0 obj=0x12c10670 self=0xc92b9400
  | sysTid=10421 nice=0 cgrp=bg_non_interactive sched=0/0 handle=0xd4f96920
  | state=S schedstat=( 30387608 25068802 93 ) utm=3 stm=2 core=0 HZ=100
  | stack=0xd4e94000-0xd4e96000 stackSize=1038KB
  | held mutexes=
  at java.lang.Object.wait!(Native method)
  - waiting on <0x0fe1baa8> (a java.lang.Object)
  at java.lang.Thread.parkFor$(Thread.java:2127)
  - locked <0x0fe1baa8> (a java.lang.Object)
  at sun.misc.Unsafe.park(Unsafe.java:325)
  at java.util.concurrent.locks.LockSupport.park(LockSupport.java:161)
  at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2035)
  at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1091)
  at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1084)
  at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1058)
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1118)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:607)
  at java.lang.Thread.run(Thread.java:761)

"amapLocManagerThread" prio=5 tid=16 Native
  | group="main" sCount=1 dsCount=0 obj=0x12c0f420 self=0xc92b9e00
  | sysTid=10423 nice=0 cgrp=bg_non_interactive sched=0/0 handle=0xd4e91920
  | state=S schedstat=( 104352291 47637446 378 ) utm=8 stm=2 core=1 HZ=100
  | stack=0xd4d8f000-0xd4d91000 stackSize=1038KB
  | held mutexes=
  kernel: __switch_to+0x70/0x7c
  kernel: SyS_epoll_wait+0x29c/0x328
```

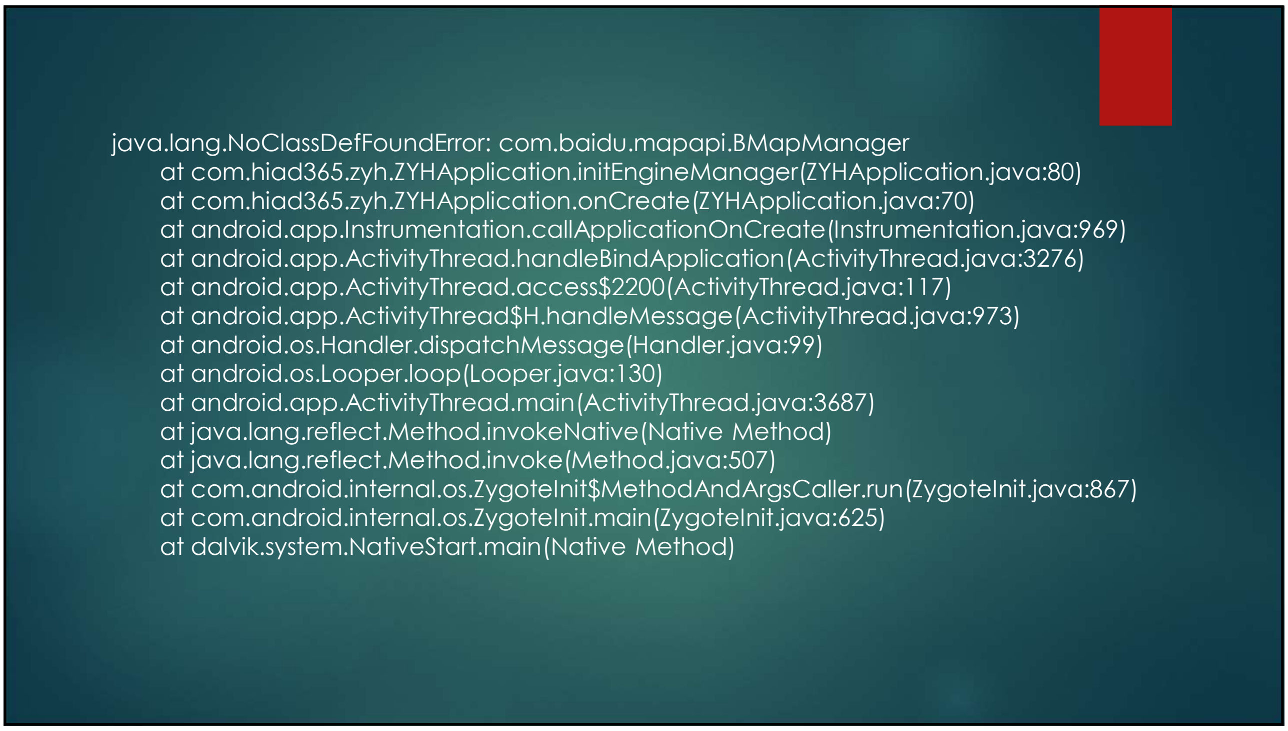
TombStones

- 默认在/data/tombstones文件夹
- 文件名为 “tombstone_00”, “tombstone_10”
- 需要root权限才可以访问（糟糕！）



```
TIME=1315550899000
FINGERPRINT=samsung/GT-S5830/GT-S5830:2.3.4/GINGERBREAD/ZCKPB:user/release-keys
HARDWARE=gt-s5830
UNKNOWN=unknown
RADIO=unknown
BOARD=GT-S5830
versionCode=17
PRODUCT=GT-S5830
versionName=2.4.3
DISPLAY=GINGERBREAD.ZCKPB
USER=se.infra
HOST=SEP-76
DEVICE=GT-S5830
TAGS=release-keys
MODEL=GT-S5830
BOOTLOADER=unknown
CPU_ABI=armeabi
CPU_ABI2=unknown
ID=GINGERBREAD
SERIAL=unknown
MANUFACTURER=samsung
BRAND=samsung
TYPE=user
```

83



```
java.lang.NoClassDefFoundError: com.baidu.mapapi.BMapManager
    at com.hiad365.zyh.ZYHApplication.initEngineManager(ZYHApplication.java:80)
    at com.hiad365.zyh.ZYHApplication.onCreate(ZYHApplication.java:70)
    at android.app.Instrumentation.callApplicationOnCreate(Instrumentation.java:969)
    at android.app.ActivityThread.handleBindApplication(ActivityThread.java:3276)
    at android.app.ActivityThread.access$2200(ActivityThread.java:117)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:973)
    at android.os.Handler.dispatchMessage(Handler.java:99)
    at android.os.Looper.loop(Looper.java:130)
    at android.app.ActivityThread.main(ActivityThread.java:3687)
    at java.lang.reflect.Method.invokeNative(Native Method)
    at java.lang.reflect.Method.invoke(Method.java:507)
    at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:867)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:625)
    at dalvik.system.NativeStart.main(Native Method)
```

84

```
java.lang.ClassNotFoundException: com.baidu.mapapi.BMapManager in loader
dalvik.system.PathClassLoader[/data/app/com.hiad365.zyh-1.apk]
        at dalvik.system.PathClassLoader.findClass(PathClassLoader.java:240)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:551)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:511)
        at com.hiad365.zyh.ZYHApplication.initEngineManager(ZYHApplication.java:80)
        at com.hiad365.zyh.ZYHApplication.onCreate(ZYHApplication.java:70)
        at android.app.Instrumentation.callApplicationOnCreate(Instrumentation.java:969)
        at android.app.ActivityThread.handleBindApplication(ActivityThread.java:3276)
        at android.app.ActivityThread.access$2200(ActivityThread.java:117)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:973)
        at android.os.Handler.dispatchMessage(Handler.java:99)
        at android.os.Looper.loop(Looper.java:130)
        at android.app.ActivityThread.main(ActivityThread.java:3687)
        at java.lang.reflect.Method.invokeNative(Native Method)
        at java.lang.reflect.Method.invoke(Method.java:507)
        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:867)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:625)
        at dalvik.system.NativeStart.main(Native Method)
```

85

切问而近思

欢迎关注格友公众号

86