



## 第8章 数学协处理器(math)

### 8.1 概述

内核目录 kernel/math 目录下包含数学协处理器仿真处理代码文件，但该程序目前还没有真正实现对数学协处理器的仿真代码，仅含有一个程序外壳，见列表 8-1 所示。

列表 8-1 linux/kernel/math 目录

	名称	大小	最后修改时间(GMT)	说明
	<a href="#">Makefile</a>	936 bytes	1991-11-18 00:21:45	
	<a href="#">math_emulate.c</a>	1023 bytes	1991-11-23 15:36:34	

### 8.2 Makefile 文件

#### 8.2.1 功能描述

math 目录下程序的编译管理文件。

#### 8.2.2 代码注释

程序 8-1 linux/kernel/math/Makefile

```

1 #
2 # Makefile for the FREAX-kernel character device drivers.
3 #
4 # Note! Dependencies are done automagically by 'make dep', which also
5 # removes any old dependencies. DON'T put your own dependencies here
6 # unless it's something special (ie not a .c file).
7 #
8 # FREAX(Linux) 内核字符设备驱动程序的 Makefile 文件。
9 # 注意！依赖关系是由 'make dep' 自动进行的，它也会自动去除原来的依赖信息。不要把你自己的
10 # 依赖关系信息放在这里，除非是特别文件的（也即不是一个.c 文件的信息）。
11
12 AR      =gar      # GNU 的二进制文件处理程序，用于创建、修改以及从归档文件中抽取文件。
13 AS      =gas      # GNU 的汇编程序。
14 LD      =gld      # GNU 的连接程序。
15 LDFLAGS =-s -x    # 连接程序所有的参数，-s 输出文件中省略所有符号信息。-x 删除所有局部符号。
16 CC      =gcc      # GNU C 语言编译器。
17 # 下一行是 C 编译程序选项。-Wall 显示所有的警告信息；-O 优化选项，优化代码长度和执行时间；
18 # -fstrength-reduce 优化循环执行代码，排除重复变量；-fomit-frame-pointer 省略保存不必要
19 # 的框架指针；-fcombine-regs 合并寄存器，减少寄存器类的使用；-finline-functions 将所有简
20 # 单短小的函数代码嵌入调用程序中；-mstring-insns Linux 自己填加的优化选项，以后不再使用；

```

```

# -nostdinc -I../include 不使用默认路径中的包含文件，而使用指定目录中的(../../include)。
14 CFLAGS =-Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs \
15         -finline-functions -mstring-insns -nostdinc -I../../include
# C 前处理选项。-E 只运行 C 前处理，对所有指定的 C 程序进行预处理并将处理结果输出到标准输
# 出设备或指定的输出文件中；-nostdinc -I../../include 同前。
16 CPP     =gcc -E -nostdinc -I../../include
17
# 下面的规则指示 make 利用下面的命令将所有的.c 文件编译生成.s 汇编程序。该规则的命令
# 指使 gcc 采用 CFLAGS 所指定的选项对 C 代码编译后不进行汇编就停止（-S），从而产生与
# 输入的各个 C 文件对应的汇编代码文件。默认情况下所产生的汇编程序文件名是原 C 文件名
# 去掉.c 而加上.s 后缀。-o 表示其后是输出文件的名称。其中$.s（或$@）是自动目标变量，
# $<代表第一个先决条件，这里即是符合条件*.c 的文件。
18 .c.s:
19     $(CC) $(CFLAGS) \
20     -S -o $.s $<
# 下面规则表示将所有.s 汇编程序文件编译成.o 目标文件。22 行是实现该操作的具体命令。
21 .s.o:
22     $(AS) -c -o $.o $<
23 .c.o:                                # 类似上面，*.c 文件→*.o 目标文件。不进行连接。
24     $(CC) $(CFLAGS) \
25     -c -o $.o $<
26
27 OBJS = math_emulate.o    # 定义目标文件变量 OBJS。
28
29 math.a: $(OBJS)           # 在有了先决条件 OBJS 后使用下面的命令连接成目标 math.a 库文件。
30     $(AR) rcs math.a $(OBJS)
31     sync
32
# 下面的规则用于清理工作。当执行'make clean'时，就会执行下面的命令，去除所有编译
# 连接生成的文件。'rm'是文件删除命令，选项-f 含义是忽略不存在的文件，并且不显示删除信息。
33 clean:
34     rm -f core *.o *.a tmp_make
35     for i in *.c;do rm -f `basename $$i .c`.s;done
36
# 下面得目标或规则用于检查各文件之间的依赖关系。方法如下：
# 使用字符串编辑程序 sed 对 Makefile 文件（即是本文件）进行处理，输出为删除 Makefile
# 文件中'### Dependencies'行后面的所有行，并生成 tmp_make 临时文件。然后对 kernel/math/
# 目录下的每个 C 文件执行 gcc 预处理操作。
# -M 标志告诉预处理程序输出描述每个目标文件相关性的规则，并且这些规则符合 make 语法。
# 对于每一个源文件，预处理程序输出一个 make 规则，其结果形式是相应源程序文件的目标
# 文件名加上其依赖关系——该源文件中包含的所有头文件列表。把预处理结果都添加到临时
# 文件 tmp_make 中，然后将该临时文件复制成新的 Makefile 文件。
37 dep:
38     sed '/\#\#\# Dependencies/q' < Makefile > tmp_make
39     (for i in *.c;do echo -n `echo $$i | sed 's,\.c,\.s,`"; \
40         $(CPP) -M $$i;done) >> tmp_make
41     cp tmp_make Makefile
42
43 ### Dependencies:

```

## 8.3 math-emulation.c 程序

### 8.3.1 功能描述

数学协处理器仿真处理代码文件。该程序目前还没有实现对数学协处理器的仿真代码。仅实现了协处理器发生异常中断时调用的两个 C 函数。math\_emulate() 仅在用户程序中包含协处理器指令时，对进程设置协处理器异常信号。

### 8.3.2 代码注释

程序 8-2 linux/kernel/math/math\_emulate.c

```

1  /*
2   * linux/kernel/math/math_emulate.c
3   *
4   * (C) 1991 Linus Torvalds
5   */
6
7  /*
8   * This directory should contain the math-emulation code.
9   * Currently only results in a signal.
10  */
11  /*
12   * 该目录里应该包含数学仿真代码。目前仅产生一个信号。
13   */
14
15 #include <signal.h>          // 信号头文件。定义信号符号常量，信号结构以及信号操作函数原型。
16
17 #include <linux/sched.h>    // 调度程序头文件，定义了任务结构 task_struct、初始任务 0 的数据，
18                               // 还有一些有关描述符参数设置和获取的嵌入式汇编函数宏语句。
19
20 #include <linux/kernel.h>    // 内核头文件。含有一些内核常用函数的原形定义。
21
22 #include <asm/segment.h>    // 段操作头文件。定义了有关段寄存器操作的嵌入式汇编函数。
23
24
25 // 协处理器仿真函数。
26 // 中断处理程序调用的 C 函数，参见 (kernel/math/system_call.s, 169 行)。
27 void math_emulate(long edi, long esi, long ebp, long sys_call_ret,
28                   long eax, long ebx, long ecx, long edx,
29                   unsigned short fs, unsigned short es, unsigned short ds,
30                   unsigned long eip, unsigned short cs, unsigned long eflags,
31                   unsigned short ss, unsigned long esp)
32 {
33     unsigned char first, second;
34
35     /* 0x0007 means user code space */
36     /* 0x0007 表示用户代码空间 */
37     // 选择符 0x000F 表示在局部描述符表中描述符索引值=1，即代码空间。如果段寄存器 cs 不等于 0x000F
38     // 则表示 cs 一定是内核代码选择符，是在内核代码空间，则出错，显示此时的 cs:eip 值，并显示信息
39     // “内核中需要数学仿真”，然后进入死机状态。
40     if (cs != 0x000F) {
41         printk("math_emulate: %04x:%08x\n|r", cs, eip);
42         panic("Math emulation needed in kernel");
43     }
44 }

```

---

```
31 // 取用户数据区堆栈数据 first 和 second, 显示这些数据, 并给进程设置浮点异常信号 SIGFPE。
32 first = get\_fs\_byte((char *)(&eip++));
33 second = get\_fs\_byte((char *)(&eip++));
34 printk("%04x:%08x %02x %02x\n", cs, eip-2, first, second);
35 current->signal |= 1<<(SIGFPE-1);
36 }
37
38 // 协处理器出错处理函数。
39 // 中断处理程序调用的 C 函数, 参见 (kernel/math/system_call.s, 145 行)。
40 void math\_error(void)
41 {
42 // 协处理器指令。(以非等待形式)清除所有异常标志、忙标志和状态字位 7。
43 \_\_asm\_\_("fnclx");
44 // 如果上个任务使用过协处理器, 则向上个任务发送协处理器异常信号。
45 if (last\_task\_used\_math)
46     last\_task\_used\_math->signal |= 1<<(SIGFPE-1);
47 }
```

---