



# 示例：Linux 的缺 页中断处理

# Linux 缺页中断相关源程序

- ◆ 系统初始化阶段，设置缺页中断的中断矢量

```
/* arch/i386/kernel/traps.c */  
set_intr_gate(14,&page_fault);
```

- ◆ 页表项中的“有效位”

- ◆ 缺页中断发生时，中断响应程序的汇编代码

```
/* arch/i386/kernel/entry.S */
```

```
ENTRY(page_fault)
```

```
    pushl $ SYMBOL_NAME(do_page_fault)  
    jmp error_code
```

- ◆ 跳转至缺页中断响应程序的 C 代码

参见 “28. 缺页响应程序的 C 函数 do\_page\_fault().doc”

# 设置缺页中断的中断矢量

```
/* arch/i386/kernel/traps.c */  
  
.....  
  
void __init trap_init(void)  
{  
  
.....  
  
set_intr_gate(14,&page_fault);  
  
.....  
  
}
```

# 设置缺页中断的中断矢量

```
/* arch/i386/kernel/traps.c */  
void set_intr_gate(unsigned int n, void *addr)  
{  
    _set_gate(idt_table+n,14,0,addr);  
}
```

# 设置缺页中断的中断矢量

```
/* arch/i386/kernel/traps.c */
#define _set_gate(gate_addr,type,dpl,addr) \
do { \
    int __d0, __d1; \
    __asm__ __volatile__ ("movw %%dx,%%ax\n\t" \
        "movw %4,%%dx\n\t" \
        "movl %%eax,%0\n\t" \
        "movl %%edx,%1" \
        : "=m" (*((long *) (gate_addr))), \
          "=m" (*(1+(long *) (gate_addr))), "=&a" (__d0), "=&d" (__d1) \
        : "i" ((short) (0x8000+(dpl<<13)+(type<<8))), \
          "3" ((char *) (addr)), "2" (__KERNEL_CS << 16)); \
    } while (0)
```

# i386 页表项中的 “有效位”

31	12	6	5	2	1	0
页表或页帧的物理地址第 31 位至第 12 位		D	A	U/S	R/W	P

P=1 则地址转换有效； P=0 则地址转换无效

R/W=1 则该页可写，可读，且可执行；

R/W=0 则该页可读，可执行，但不可写

U/S=1 则该页可在任何特权级下访问；

U/S=0 则该页只能在特权级 0、1 和 2 下访问；

A：访问位

D：已写标志位

# 缺页中断响应程序的汇编代码

```
/* arch/i386/kernel/entry.S */
```

```
ENTRY(page_fault)
```

```
    pushl
```

```
    $SYMBOL_NAME(do_page_fault)
```

```
    jmp error_code
```

# 缺页中断响应程序的汇编代码

```
/* arch/i386/kernel/entry.S */  
error_code:  
    ..... # 保护现场  
    movl ORIG_EAX(%esp), %esi # 错误代码  
    movl ES(%esp), %edi      # C 响应函数的首地址  
    .....  
    pushl %esi               # 错误代码压栈  
    .....  
    call *%edi               # 跳转至 C 响应函数  
    addl $8,%esp             # 从 C 函数返回，调整堆栈  
    jmp ret_from_exception
```



# 缺页中断响应程序的汇编代码

```
/* arch/i386/kernel/entry.S */
```

```
ret_from_exception:
```

```
.....
```

```
    jmp restore_all
```

```
restore_all:
```

```
    RESTORE_ALL
```

# 缺页中断响应程序的汇编代码

```
/* arch/i386/kernel/entry.S */
#define RESTORE_ALL \
    popl %ebx; \
    popl %ecx; \
    popl %edx; \
    popl %esi; \
    popl %edi; \
    popl %ebp; \
    popl %eax; \
1: popl %ds; \
2: popl %es; \
   addl $4,%esp;\
3: iret; \
```

A white, fluffy cloud shape is centered horizontally on a solid blue background. The cloud has a soft, irregular outline with many small, rounded protrusions and indentations, giving it a realistic, puffy appearance. It is positioned in the middle of the frame, both vertically and horizontally.

**End**