统计操作系统缺页次数

实验目的

学习操作系统的存储管理原理；理解操作系统存储管理的分页、虚拟内存、“按需调页”思想及方法；掌握 Linux 内核对虚拟内存、虚存段、分页式存储管理、按需调页的实现机制。

实验内容

统计操作系统自内核加载以后，累计发生的缺页次数，以及总运行时间。

由于每发生一次缺页都要进入缺页中断服务函数 do_page_fault 一次，所以可以认为执行该函数的次数就是系统发生缺页的次数。因此可以定义一个全局变量 pfcount 作为计数变量，在执行 do_page_fault 时，该变量值加 1。至于经历的时间可以利用系统原有的变量 jiffies，这是一个系统的计时器，在内核加载完以后开始计时，以 10ms（缺省）为计时单位。

借助/proc 文件系统来读出变量的值。在/proc 文件系统下建立目录 pf 以及在该目录下的文件 pfcount 和 jiffies。

实验指导

1. 实验原理

由于每发生一次缺页都要进入缺页中断服务函数 do_page_fault 一次，所以可以认为执行该函数的次数就是系统发生缺页的次数。因此可以定义一个全局变量 pfcount 作为计数变量，在执行 do_page_fault 时，该变量值加 1。

至于系统自开机以来经历的时间，可以利用系统原有的变量 jiffies。这是一个系统的计时器，在内核加载完以后开始计时，以 10ms（缺省）为计时单位。

当然，读取变量 pfcount 和变量 jiffies 的值，还需要借助/proc 文件系统。在/proc 文件系统下建立目录 pf；并且在 pf 目录下，建立文件 pfcount 和 jiffies。

2. 实验实施

先在 include/linux/mm.h 文件中声明变量 pfcount：

```
--- linux-2.6.15/include/linux/mm.h.orig
+++ linux-2.6.15/include/linux/mm.h
***************
*****26,29*****
extern unsigned long num_physpages;
extern void * high_memory;
extern unsigned long vmalloc_earlyreserve;
extern int page_cluster;
+ extern unsigned long pfcount;
```

在 arch/i386/mm/fault.c 文件中定义变量 pfcount：

```
--- linux-2.6.15/arch/i386/mm/fault.c.orig
+++ linux-2.6.15/arch/i386/mm/fault.c
***************
****227,235****
+ unsigned long pfcount;
fastcall void __kprobes do_page_fault(struct pt_regs *regs,
                        unsigned long error_code)
{
   struct task_struct *tsk;
   struct mm_struct *mm;
   struct vm_area_struct * vma;
   unsigned long address;
   unsigned long page;
   int write, si_code;
```

每次产生缺页中断，并且确认是由缺页引起的，则将变量值递增 1。这个操作在 do_page_fault()函数中执行：

```
--- linux-2.6.15/arch/i386/mm/fault.c.orig
+++ linux-2.6.15/arch/i386/mm/fault.c
***************
****328,328****
goodarea:
+   pfcount++;
```

在 kernel/time.c 文件中加入 EXPORT_SYMBOL(pfcount)，让内核模块能够读取变量 pfcount；同理，内核模块也可以读取 jiffies：

```
--- linux-2.6.15/kernel/time.c.orig
+++ linux-2.6.15/kernel/time.c
***************
****687,687****
EXPORT_SYMBOL(jiffies);
+ extern unsigned long pfcount;
+ EXPORT_SYMBOL(pfcount);
```

以上部分是对 Linux 内核源代码的几处修改。若让它们起作用，显然，需要重新编译内核，产生新的内核的 image；并且，重新启动主机，装入新编译生成的 image。内核的编译和装入，可参见“编译 Linux 内核”一章。

读取 pfcount 和 jiffies 变量的内核模块，需要新编写一个文件：pf.c

```
#include <linux/proc_fs.h>
#include <linux/slab.h>
#include <linux/mm.h>
#include <linux/sched.h>
#include <linux/string.h>
#include <linux/types.h>
#include <linux/ctype.h>
#include <linux/kernel.h>
#include <linux/version.h>
#include <linux/module.h>


struct proc_dir_entry *proc_pf;                /*/proc/pf/ 目录项*/
struct proc_dir_entry *proc_pfcount, *proc_jiffies;  /* /proc/pf/pfcount 和/proc/pf/jiffies 文件项*/

/*下面这个函数用于建立/proc/pf/ 目录项*/
static inline struct proc_dir_entry *proc_pf_create(const char* name, mode_t mode,
get_info_t *get_info)
{
      return create_proc_info_entry(name, mode, proc_pf, get_info);
}

/*注意 不同的内核版本，修改的代码不一样

/*读取 pfcount 的值*/
int get_pfcount(char *buffer, char **start, off_t offset, int length)
{
       int len = 0;
       len = sprintf(buffer, "%ld\n", pfcount);
       /* pfcount is defined in arch/i386/mm/fault.c */
       return len;
}

/*读取 jiffies 的值*/
int get_jiffies(char *buffer, char **start, off_t offset, int length)
{
       int len = 0;
       len = sprintf(buffer, "%ld\n", jiffies);
       return len;
}
```

```
/*模块初始化进程，建立/proc 下的目录和项*/
int init_module(void)
{
        proc_pf = proc_mkdir("pf", 0);
        proc_pf_create("pfcount", 0, get_pfcount);
        proc_pf_create("jiffies", 0, get_jiffies);
        return 0;
}

/*模块清除进程，清除/proc 下的相关目录和文件*/
void cleanup_module(void)
{
            remove_proc_entry("pfcount", proc_pf);
            remove_proc_entry("jiffies", proc_pf);
            remove_proc_entry("pf", 0);
}
MODULE_LICENSE(“GPL”);
```

在编译内核模块前，先准备一个 Makefile：

```
TARGET = pf
KDIR = /usr/src/linux
PWD = $(shell pwd)
      obj-m += $(TARGET).o
default:
      make -C $(KDIR) M=$(PWD) modules
```

然后简单输入命令 make：

```
#make
```

结果，我们得到文件“pf.ko”！这意味着，你成功了。

然后执行加载模块命令：

```
#insmod pf.ko
```

这样就可以通过作为中介的/proc 文件系统，轻松地读取我们所需要的两个变量的值了。使用命令 cat /proc/pf/pfcount /proc/pf/jiffies，就可以在终端打印出至今为止的缺页次数和已经经历过的 jiffies 数目。

隔几分钟再使用命令 cat /proc/pf/pfcount /proc/pf/jiffies，查看一下打印出的缺页次数和 jiffies 数目。比较一下结果。

撰写实验报告的要求

1. 按照实验报告模板格式撰写；
2. 整个实验过程的解图；

3. 源程序的修改部分，运行结果的解图；
4. 实验过程中遇到的问题及解决方法等。
5. 心得体会