# 实验二: 统计操作系统缺页次数

林

# 目录

**一．实验目的---------------------------------------------3**

**二．实验内容--------------------------------------------3**

**三.实验步骤----------------------------------------------3**

# 统计操作系统缺页次数

# 一 实验目的

学习虚拟内存的基本原理和Linux虚拟内存管理技术；

深入理解、掌握Linux的按需调页过程；

掌握内核模块的概念和操作方法，和向/proc文件系统中增加文件的方法；

综合运用内存管理、系统调用、proc文件系统、内核编译的知识。

# 二 实验内容

#### 1.原理

Linux的虚拟内存技术采用按需调页，当CPU请求一个不在内存中的页面时,会发生缺页,缺页被定义为一种异常（缺页异常），会触发缺页中断处理流程。每种CPU结构都提供一个do\_page\_fault处理缺页中断。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。本实验通过动态加载模块的方法，利用/proc文件系统作为中介来获取该值。

#### 2.实验环境

操作系统：Ubuntu 12.04（内核版本为3.2.0-23-generic-pae）

内核源码：linux-3.2.58

# 三 实验步骤

#### 下载一份内核源代码并解压

Linux受GNU通用公共许可证（GPL）保护，其内核源代码是完全开放的。现在很多Linux的网站都提供内核代码的下载。推荐使用Linux的官方网站：<http://www.kernel.org>。

在terminal下可以通过wget命令下载源代码：

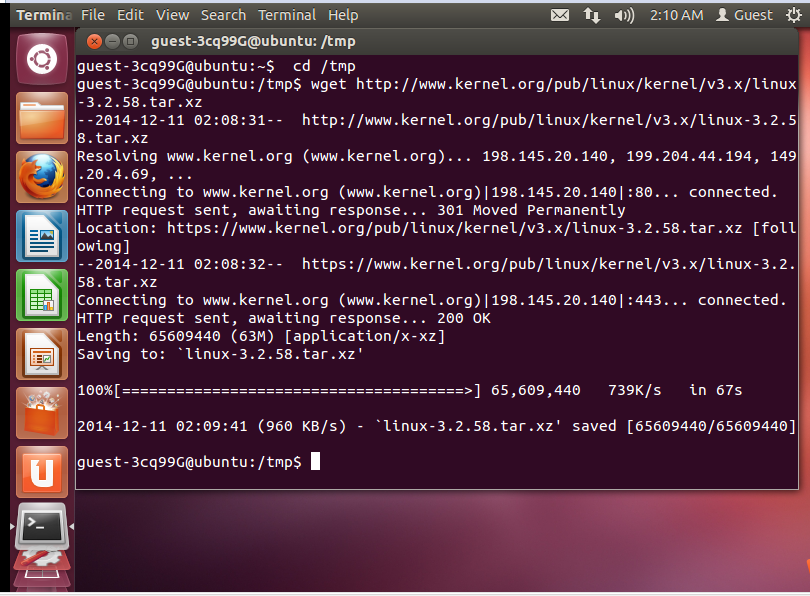
$ cd /tmp

$ wget <http://www.kernel.org/pub/linux/kernel/v3.x/linux-3.2.58.tar.xz>

切换到root身份，解压源代码到/usr/src目录下：

# xz –d linux-3.2.58.tar.xz

# tar –xvf linux-3.2.58.tar –C /usr/src



#### 2.修改内核源代码，添加统计变量

1、切换到预编译内核目录

#cd /usr/src/linux-3.2.58

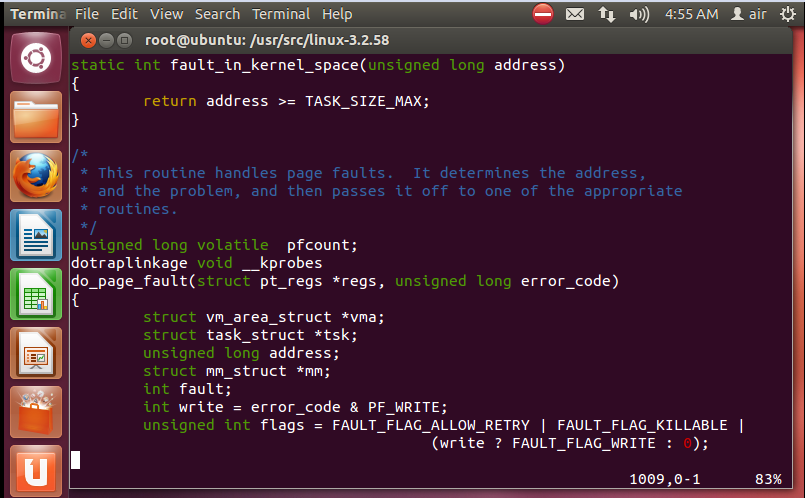
2、修改处理内存访问异常的代码

//用vi编辑器打开fault.c，一般使用Intel x86体系结构，则修改arch/x86/目录下的文件

#vi arch/x86/mm/fault.c

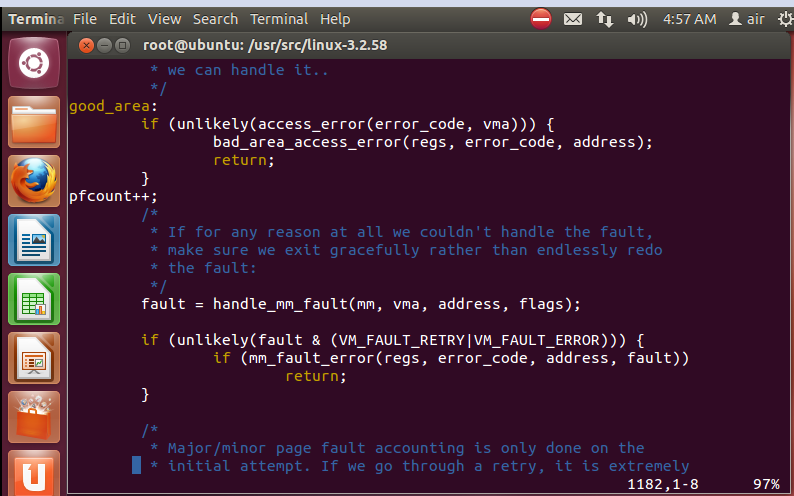
//在do\_page\_fault函数的上一行定义统计缺页次数的全局变量pfcount

Unsigned long volatile pfcount;



//将pfcount加入到do\_page\_fault中，用以统计缺页次数

pfcount++;



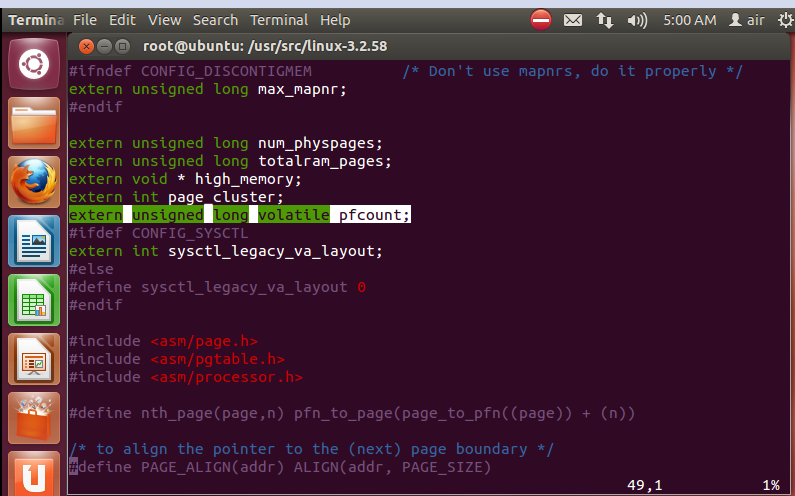
3、修改内存管理代码

//用vi编辑器打开头文件mm.h

#vi include/linux/mm.h

//在mm.h中加入全局变量pfcount的声明，代码加在extern int page\_cluster;语句之后

extern unsigned long volatile pfcount;

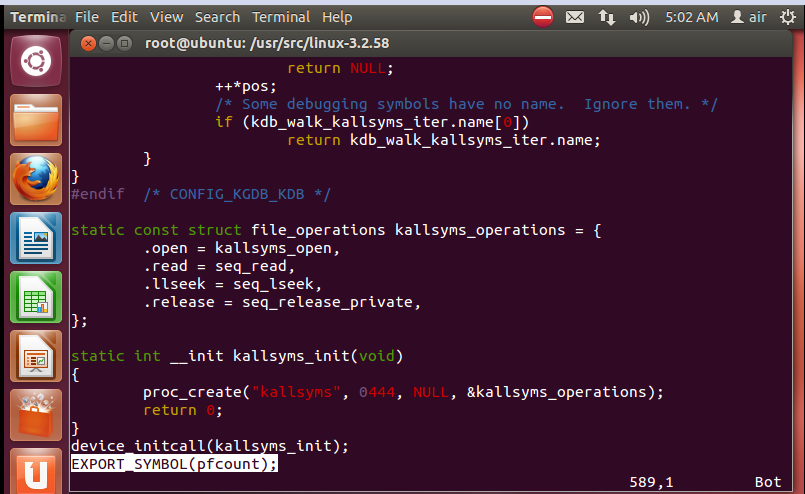


4、导出pfcount全局变量，让整个内核（包括模块）都可以访问。方法是：

#vi kernel/kallsyms.c

//在文件最后加入一行代码

EXPORT\_SYMBOL(pfcount);



#### 配置编译新内核

在编译内核前，一般来说都需要对内核进行相应的配置。配置是精确控制新内核功能的机会。配置过程也控制哪些需编译到内核的二进制映像中(在启动时被载入)，哪些是需要时才装入的内核模块（module）。

首先进入内核源代码目录：

# cd /usr/src/linux-3.2.58

如果不是第一次编译的话，有必要将内核源代码树置于一种完整和一致的状态（如果是第一次可跳过此步）。因此，推荐执行命令make mrproper。它将清除目录下所有配置文件和先前生成核心时产生的.o文件：

#make mrproper

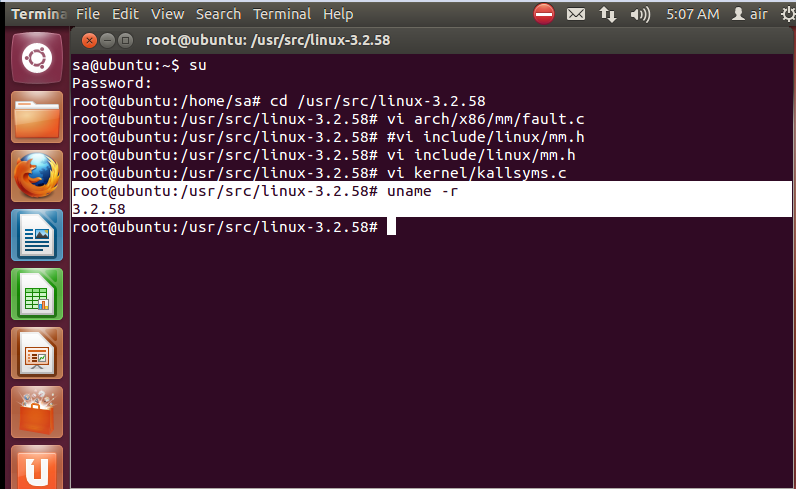
然后配置编译选项（此处使用原内核的配置文件，完整的配置命令看操作提示）：

# cp /boot/config-3.2.0-23-generic-pae .config

该命令的作用是将原内核配置文件拷贝的当前目录下，并命名为.config。若需要进一步修改配置请参照操作提示。

用编译Linux内核预备实验中的方法完成新内核的配置、编译、替换，重启后验证是否完成替换。

#uname –r //如果为3.2.58（与你采用的新内核版本一至）说明替换完成



#### 4.编写读取pfcount值的模块代码

系统重启后，执行如下操作：

cd /home/sa

#mkdir source //在当前用户目录下创建source文件夹，用于存放编写的用户程序

#cd source //切换到source目录

#vi pf.c //新建用于构建模块的代码

--------------------------------------------

/\*pf.c\*/

/\*modules program\*/

#include <linux/init.h>

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/mm.h>

#include <linux/proc\_fs.h>

#include <linux/string.h>

#include <asm/uaccess.h>

struct proc\_dir\_entry \*proc\_pf;

struct proc\_dir\_entry \*proc\_pfcount;

extern unsigned long volatile pfcount;

static inline struct proc\_dir\_entry \*proc\_pf\_create(const char\* name, mode\_t mode, read\_proc\_t \* get\_info)

{

return create\_proc\_read\_entry(name,mode,proc\_pf,get\_info,NULL);

}

int get\_pfcount(char \*buffer, char \*\*start, off\_t offset, int length, int \*peof, void \*data)

{

int len = 0;

len = sprintf(buffer, "%ld \n", pfcount);

return len;

}

static int pf\_init(void)

{

proc\_pf = proc\_mkdir("pf", 0);

proc\_pf\_create("pfcount", 0, get\_pfcount);

return 0;

}

static void pf\_exit(void)

{

remove\_proc\_entry("pfcount",proc\_pf);

remove\_proc\_entry("pf",0);

}

module\_init(pf\_init);

module\_exit(pf\_exit);

MODULE\_LICENSE("GPL");

#### 编译、构建内核模块

cd /home/sa/source

#vi Makefile //在source目录下建立Makefile文件

在Makefile中添加如下内容：

obj-m := pf.o

编译内核模块：

make -C /usr/src/linux-3.2.58 SUBDIRS=/home/sa/source modules

#### 加载模块到内核中

执行加载模块命令：

#insmod pf.ko

查看统计缺页次数：

#cat /proc/pf/pfcount

