显示系统缺页次数

显示系统缺页次数

实验简介：

添加系统调用，显示系统启动以来的缺页次数。

实验原理：

》系统每次缺页都会执行缺页中断服务函数do_page_fault，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，定义一个全局变量pfcount记录此函数执行次数即为系统缺页次数。

》除了上节实验课学习的proc系统，用户空间和内核空间的交互还可以由系统调用来完成。通过对pfcount变量添加系统调用，就可以得到系统启动以来的缺页次数。

step1：添加pfcount变量

在linux-version_number/include/linux/mm.h中添加变量pfcount声明：

mm.h文件是是一些内存管理的变量的声明和一些函数原型，在这里为了保证代码统一，也将pfcount的声明放在这里。

```
77
78 extern void * high_memory;
79 extern int page_cluster;
80 extern long pfcount;
```

在linux-version_number/arch/x86/mm/fault.c中定义并使用变量：

实际需要在fault.c文件中使用pfcount变量，将其在函数外部定义为全局变量。

```
1492  */
1493 long pfcount;
1494 static noinline void
1495 __do_page_fault(struct pt_regs *regs, unsigned long hw_error_code,
1496                 unsigned long address)
```

step1：添加pfcount变量

在do_page_fault函数中加入对pfcount值的更新。

```
1494 static noinline void
1495 __do_page_fault(struct pt_regs *regs, unsigned long hw_error_code,
1496                 unsigned long address)
1497 {
1498         pfcount++;
1499         prefetchw(&current->mm->mmap_sem);
1500
1501         if (unlikely(kmmio_fault(regs, address)))
1502                 return;
1503
1504         /* Was the fault on kernel-controlled part of the address space? */
1505         if (unlikely(fault_in_kernel_space(address)))
1506                 do_kern_addr_fault(regs, hw_error_code, address);
1507         else
1508                 do_user_addr_fault(regs, hw_error_code, address);
1509 }
1510 NOKPROBE_SYMBOL(__do_page_fault);
```

step2：添加系统调用号

在系统调用表/arch/x86/entry/syscalls/syscall_64.tbl中添加自己的系统调用号

```
357 433     common  fspick                  __x64_sys_fspick
358 434     common  pidfd_open              __x64_sys_pidfd_open
359 435     common  clone3                  __x64_sys_clone3/ptregs
360 2020    64      pf_count                __x64_sys_pf_count
361
362 #
363 # x32-specific system call numbers start at 512 to avoid cache impact
```

step3：添加调用函数

在/kernel/sys.c 中添加系统调用函数

```
SYSCALL_DEFINE0(pf_count)
{
        printk("entering syscall pf_count\n");
        return pfcount;
}
```

其中SYSCALL_DEFINE0(name)是宏，其定义在/include/linux/syscalls.h文件中，相当于asmlinkage long sys_pf_count(void){ ... }

step4：编译内核

编译并安装内核。

详细步骤在第一节课件中有描述。

完成后重启。

step5：编写C程序并查看结果

```c
#include <stdio.h>
#include <unistd.h>  //封装系统API接口

int main()
{
    long pfcount = syscall(2020);
    printf("pfcount: %ld\n", pfcount);
    return 0;
}
```

运行查看系统缺页次数

>> gcc pfcount.c -o pfcount

```
amos@ubuntu:~/Desktop/4$ ./pfcount
pfcount: 1126652
```

作业

1. 完成本节实验并书写实验报告
2. 回顾上节实验课，利用proc文件系统显示系统缺页次数。

（其中代码已给出，实验步骤以及一些细节会有所不同，大家根据所学知识自行领会）