操作系统Lab8实验报告

2012011350 计24 周琳钧

练习0：填写已有实验

本练习无需添加更多的代码，这里直接merge即可，略过

练习1：完成读文件操作的实现

1. 实验过程：本实验主要完成sys\_inode.c中的sys\_io\_nolock函数完成在SFS层面上对文件的读写。本练习比较简单，主要用到三个函数：sfs\_bmp\_load\_nolock（将逻辑块映射到真正的磁盘物理块）, sfs\_buf\_op（将文件内容读入到buf中），sfs\_block\_op（将文件按块读入到buf中）。实验流程如下：对于一段文件中的内容，由于随机访问性可能起始和结束并不是块对齐的，因此我们将目标内容划分成3个区域：从起点到该块结束，中间好多好多个块和最后一块的起始到文件结束，这样，第1、3块按照sys\_buf\_op来读，第2块按照sfs\_block\_op来读取即可。在读取过程中只需维护alen, buf, blkno, nblks几个变量就可轻松完成。
2. 问答题：Unix的PIPE机制是指前一个程序的输出可以用作后一个程序的输入，初步的实现方案是设置一个临时保存输出的文件，当程序1输出时，保存一份到这个文件中，然后当程序2要输入时，通过文件系统读取这个文件作为程序2的输入即可。在程序中可以用两个file对象，一个是读取该文件，另一个是写该文件，并创建一个inode即可。

练习2：完成基于文件系统的执行程序机制的实现

1. 实验过程：本实验共分为两部分，第一部分要完成对proc.c中do\_fork函数的改进，这个非常简单，只需在原有步骤的基础上加上一条函数copy\_files，将父进程的文件系统拷贝到子进程的文件系统中即可。第二部分代码量比较大，是要改写proc.c中的load\_icode函数，让操作系统真正实现能够通过读取硬盘的方式加载程序，主要流程描述如下：第1,2步分别为新进程创建mm和PDT，这与之前的代码是一致的，第3步开始出现了不一样的情况：首先调用load\_icode\_read函数加载elf格式文件，并通过魔数判断该elf文件是否是可用的，接着对于每一个程序段完成以下操作：首先调用load\_icode\_read函数加载程序起始段ph，然后调用mm\_map函数建立新的vma（这部分与原代码相同），接着就是分配内存，并从文件系统中读入程序段，包括TEXT/DATA段，BSS段，，这一层循环就结束了。关闭文件，这样程序的所有部分就被加载进了内存中。接下来就要完成对新进程的堆栈分配操作。首先从0xB0000000处向下分配堆栈，然后设置mm，sr3，CR3等寄存器（这部分与原代码相同），接着要将argc和argv的内容置入栈中，栈结构如下：

###

argc

argv\_ptr

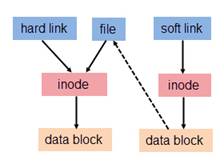
argv

STACKTOP

TOP

最后，设置trapframe即可（这一步和之前的代码相同）。整个load\_icode到这里就结束了。

1. 问答题：硬链接和软链接都是能够指向一个其他目录的文件，不同的是硬链接只是一个指向，文件系统并不为它分配inode，而软连接可以看做是一个独立的文件，文件系统会为它分配inode。可以用下图表示：



因此，根据这个访问策略，我们可以设计软连接和硬链接如下：当创建硬链接时，只需将该硬链接指向文件指向的那个inode即可，而创建软连接时，需要按创建文件那样创建一个新文件，并将文件内容设置为软连接指向的那个文件即可。

最后，展示一下我的完整操作系统ucore：

