南京大学本科生实验报告

课程名称：**操作系统**　　任课教师：叶保留　　助教：尹熙喆/水兵

学院	**计算机科学与技术**	专业（方向）	**计算机科学与技术**
学号	**201220062**	姓名	**黄子睿**
Email	**201220062@smail.nju.edu.cn**	开始/完成日期	**5/18-5/19**

1. 实验名称：

OS LAB 5

2. 实验目的：

实现一个建议的文件系统，熟悉 liunx 文件架构的设计。

3. 正文：

Exercise1：想想为什么我们不使用文件名而使用文件描述符作为文件标识。

答：因为同一个文件可能会被打开多次，对应在 linux 文件体系结构中就是一个文件可以对应若干个不同的系统打开文件表中的 file 表项。利用文件名无法实现这一功能，但是使用文件描述符则可以通过进程打开文件表对应到需要的 file 表项，以实现文件的动态共享。

Exercise2：为什么内核在处理 exec 的时候，不需要对进程描述符表和系统文件打开表进行任何修改。

答：因为 exec 不创建新的进程，仅仅用替换原来进程所运行的程序与程序对应的上下文环境，因此 exec 不需要改变进程描述符表的内容。同时，exec 前后运行的程序可能（实际上往往如此）是针对同一个文件的同一次打开进行的，因此

文件相关的进程文件打开表与系统文件打开表都不需要改变。（一个直观的例子是，fork 之后的子进程常常利用 exec 来执行与父进程不同的程序，而此时子进程可能仍然需要与父进程在同一个文件的同义词打开上操作，因此 exec 不应当隐式地改变进程的文件环境）利用 exec 的这一特性，可以方便地实现文件的重定向。

Exercise3：我们可以通过 which 指令来找到一个程序在哪里，比如 which ls，就输出 ls 程序的绝对路径（看下面，绝对路径是/usr/bin/ls）。那我在/home/yxz 这个目录执行 ls 的时候，为什么输出/home/yxz/路径下的文件列表，而不是/usr/bin/路径下的文件列表呢？

答：利用了进程的工作目录，此时执行 ls 时 shell 进程的工作目录在/home/yxz，由此 ls 指令将通过进程记录的工作目录输出该工作路径下的文件列表，而不关心 ls 本身所在的绝对目录位置。

这里由衷地赞叹一句，lab5 是我见过这 5 个实验里函数抽象封装最好最完善的一个~好代码确实看得赏心悦目，而且用起来也得心应手，吹一波助教哥哥~

Challenge1：system 函数（自行搜索）通过创建一个子进程来执行命令。但一般情况下，system 的结果都是输出到屏幕上，有时候我们需要在程序中对这些输出结果进行处理。一种解决方法是定义一个字符数组，并让 system 输出到字符数组中。如何使用重定向和管道来实现这样的效果？

答：这里给出相关的伪代码，对应的源代码文件请参考./report/pipeRedirect.c：

```
def linux-System-Redirect(argv):
    buf := [ ]
    open pipe with two files pipeRead and pipeWrite
    redirect STD_OUT to pipeWrite with function dup() and dup2()
    system(string from argv)
    read contents from pipeRead to buf
    redirect to STD_OUT
    printf(“%s\n”, buf);
    revise buf as program needs e.g. reverse the output sentence.
    printf(“%s\n”, buf);
```

在这个过程中，最重要的两个要素是父子进程分享文件的方式以及如何更改程序的标准输出（实际上就是 system，printf 等函数对应的输出文件）。对于前者，子进程在 fork 时直接复制父进程的用户打开文件表，因此实际上两者的文件情况是完全一致的（指 fork 恰好完成时刻），因此在父进程中更改标准输出的方向为 pipe 的写口后，直接调用 system 函数创建子进程执行 shell 指令就可以将输出写进管道，这时候父进程再从 pipe 读口读取文件就可以了。对于后者，除了 pipe(int filedes)函数打开管道外，还需要调用 dup2(filedes[1], 1)函数，即关闭 1 对应的 STD_OUT 并将 filedes[1]对应的 pipe 写口复制到 STD_OUT 的位置，同时为了之后恢复 STD_OUT（利用 pipe 处理完 system 函数的输出之后还需要将处理的结果输出到控制台上），在调用 dup2 之前，首先通过 dup 保存 STD_OUT 的用户打开文件表项，之后再用 dup2 恢复即可。

这样通过管道与基于 dup 函数的文件重定向，就实现了此 challenge 要求的功能。