# “进程与文件描述符”实验报告

## 实验题目

使用fork(), exec(), dup2(), pipe() ，open()系统调用完成与下列shell命令等价的功能。（提示：为简化编程，不需要用strtok断词，直接实现能达到下述shell命令相同功能的程序即可）

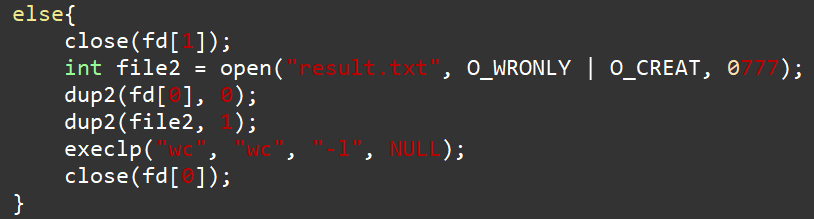
grep -v usr < /etc/passwd | wc -l > result.txt

## 实验步骤

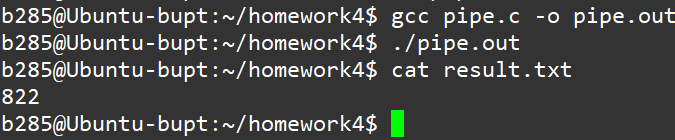
1. 创建父子进程，在父进程中使用dup2替换输入和输出，使其从文件输入，并输出到管道中：



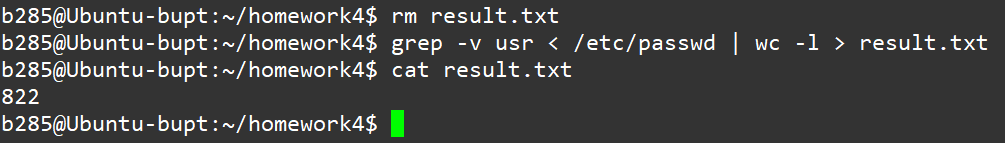
1. 在子进程中同样替换输入和输出，使其从管道输入，从文件result.txt中输出：



1. 编译运行程序，再查看result中的值：



1. 使用题目中的命令检查输出是否一致：



## 实验总结

本次实验让我初步尝试了管道的使用，重点在于父子进程对于管道读写的控制以及使用dup2命令修改输入输出的位置，这一点让我印象深刻，由此可见管道是进程间通信的一种常用手段。

## 源代码

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <fcntl.h>  #include <unistd.h>  #include <sys/wait.h>  int main(){  int fd[2];  pipe(fd);  pid\_t pid = fork();  if(pid == -1){  printf("error");  }  else if(pid != 0){  close(fd[0]);  int file1 = open("/etc/passwd", O\_RDONLY);  dup2(file1, 0);  dup2(fd[1], 1);  execlp("grep", "grep", "-v", "usr", NULL);  close(fd[1]);  }  else{  close(fd[1]);  int file2 = open("result.txt", O\_WRONLY | O\_CREAT, 0777);  dup2(fd[0], 0);  dup2(file2, 1);  execlp("wc", "wc", "-l", NULL);  close(fd[0]);  }  close(fd[0]);  close(fd[1]);  } |