**实验一 同步与异步write的效率比较**

1、程序的参数和输入

实验要求程序必须指定输出的文件名，而该文件是否按同步方式打开，则是可以选择的。因此程序至少带一个、至多两个输入参数。程序默认从标准输入STDIN\_FILENO读取输入文件，可以利用shell的输入定向功能选择具体的输入文件。

1. 系统调用times()的说明

#include <sys/times.h>

clock\_t times(struct tms \*buf);

struct tms {

clock\_t tms\_utime; /\* 记录进程除系统调用外所使用的CPU时间 \*/

clock\_t tms\_stime; /\* 记录进程的系统调用所使用的CPU时间 \*/

clock\_t tms\_cutime; /\* 记录子进程除系统调用外所使用的CPU时间 \*/

clock\_t tms\_cstime; /\* 记录子进程的系统调用所使用的CPU时间 \*/

};

times函数的返回值是进程迄今为止的存活时间。所有时间都是以“滴答”为单位的，函数 sysconf(\_SC\_CLK\_TCK)可获得所运行系统每秒的滴答数(参考课本P33)。

3、计算write耗费的时间

为了准确计算write耗费的时间，很重要的就是要避免将read的时间计入，因为I/O操作的时间通常是毫秒级的，不可以忽略。一种有效的方法是，设置一个与输入文件长度相同的缓冲区，一次性地将输入文件读入缓冲区，而后就不必再读输入文件。这样就可以有效避免计入read的时间。

有同学可能会问，难道可以在内存中创建一个几十上百兆乃至上G的缓冲区吗？回答是没问题！因为我们所运行的操作系统系统具有虚拟存储管理功能。

设置输入缓冲区时需要知道输入文件的长度。除了使用系统调用stat外，更简单的方法是利用lseek的返回值来获取文件的长度。

在按每一个给定大小的输出缓冲区计算写文件时间时，应当在开始写之前调用times()，记录下开始时间，然后在整个输入缓冲区都复制到输出文件之后，再调用times()，两次调用times()的时间间隔，就是在这个给定大小的输出缓冲区的限制下，复制整个输入文件所耗费的写时间。至于在每一次写的时候所执行的其他语句，它们相较于I/O操作，所花费的时间极小，可以忽略不计。

注意，在开始按一个给定大小的输出缓冲区复制输入文件时，应当先将输出文件的写位置复位到输出文件的开头（这意味着不能以O\_APPEND方式打开输出文件）。可以使用lseek做到这一点，以避免多次打开、关闭输出文件。