多线程拷贝：

**#include <iostream>**

**#include <cstring>**

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <sys/stat.h>**

**#include <fcntl.h>**

**#include <unistd.h>**

**#include <pthread.h>**

**using namespace std;**

//最大使用的线程数

const int MAX\_THREADS = 5;

typedef struct TAG\_INFO

{

char \*fromfile; //源地址

char \*tofile; //目的地址

int num; //启动的第i-1个进程

}info;

int get\_size(const char \*filename)

{

struct stat st;

memset(&st, 0, sizeof(st));

stat(filename, &st);

return st.st\_size;

//st\_size的类型为\_\_off\_t

}

void\* threadDL(void \*param)

{

info info1 = \*((info\*)param);

FILE \*fin = fopen(info1.fromfile, "r+");

FILE \*fout = fopen(info1.tofile, "w+");

int size = get\_size(info1.fromfile);

//将文件指针分别设置在每个线程要读和写的位置

fseek(fin, size\*(info1.num)/MAX\_THREADS, SEEK\_SET);

fseek(fout, size\*(info1.num)/MAX\_THREADS, SEEK\_SET);

char buff[1024] = {'\0'};

int len = 0;

int total = 0;

while((len = fread(buff, 1, sizeof(buff), fin)) > 0)

{

fwrite(buff, 1, len, fout);

total += len;

//如果读入的数据大于文件总大小除线程总数则停止读入，因为每个线程要读或写的数据就等于文件总大小除线程总数

//可能会多写入一些数据，下一次写入时会覆盖多写入的数据，所以不用担心

if(total > size/MAX\_THREADS)

{

break;

}

}

fclose(fin);

fclose(fout);

}

int main(int argc, char \*argv[])

{

//先创建一个与文件1同样大小的文件

creat(argv[2], 0777);

truncate(argv[2], get\_size(argv[1]));

pthread\_t pid[MAX\_THREADS]; //存放线程标识数组

info info1;

//启动指定线程数的线程

for(int i = 0; i < MAX\_THREADS; i++)

{

memset(&info1, 0, sizeof(info1));

info1.fromfile = argv[1];

info1.tofile = argv[2];

info1.num = i;

pthread\_create(&pid[i], NULL, threadDL, (void\*)&info1);

}

//等待线程结束

for(int j = 0; j < MAX\_THREADS; j++)

{

//pthread\_join不能用在创建进程的for循环中，否则创建第一个进程后会等待第一个进程结束后创建第二个进程

pthread\_join(pid[j], NULL);

}

cout << "file copy success......" << endl;

return 0;

}

多进程拷贝：

#include <iostream>

#include <cstring>

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <pthread.h>

#include <stdlib.h>

using namespace std;

int getfilesize(const char \*src\_file)

{

int len;

FILE \*fp;

fp = fopen(src\_file,"r");

fseek(fp,0,SEEK\_END);

len = ftell(fp);

rewind(fp);

return len;

}

int main(int argc, char \*argv[])

{

creat(argv[2],0777);

int size=getfilesize(argv[2]);

truncate(argv[2],size);

char \*filein=argv[1]; //源文件

char \*fileout=argv[2]; //目的文件

FILE \*fin=fopen(argv[1],"r+");

FILE \*fout=fopen(argv[2],"w+");

pid\_t pid;

pid=fork(); //创建子进程

int len=0;

int totallen=0;

if(pid==0) //父进程

{

fseek(fin,0,SEEK\_SET);

fseek(fout,0,SEEK\_SET);

char buff[1024]={'\0'};

while((len=fread(buff,1,sizeof(buff),fin))>0)

{

fwrite(buff,1,len,fout);

totallen+=len;

if(totallen>size/2)

{

break;

}

}

exit(0); //结束进程

}

else //子进程

{

fseek(fin,size/2,SEEK\_SET);

fseek(fout,size/2,SEEK\_SET);

char buff[1024]={'\0'};

while((len=fread(buff,1,sizeof(buff),fin))>0)

{

fwrite(buff,1,len,fout);

totallen+=len;

if(totallen>size/2)

{

break;

}

}

exit(0); //结束进程

}

return 0;

}