 线程的创建和使用 线程的创建是用下面的几个函数来实现的. #include int pthread_create(pthread_t *thread,pthread_attr_t *attr, void *(*start_routine)(void *),void *arg). void pthread_exit(void *retval). int pthread_join(pthread *thread,void **thread_return). pthread_create创建一个线程,thread是用来表明创建线程的ID,attr指出线程创建时候的属性,我们用NULL来表明使用缺省属性.start_routine函数指针是线程创建成功后开始执行的函数,arg是这个函数的唯一一个参数.表明传递给start_routine的参数. pthread_exit函数和exit函数类似用来退出线程.这个函数结束线程,释放函数的资源,并在最后阻塞,直到其他线程使用pthread_join函数等待它.然后将*retval的值传递给**thread_return.由于这个函数释放所以的函数资源,所以retval不能够指向函数的局部变量. pthread_join和wait调用一样用来等待指定的线程. 下面我们使用一个实例来解释一下使用方法.在实践中,我们经常要备份一些文件.下面这个程序可以实现当前目录下的所有文件备份.备份后的后缀名为bak #include #include #include #include #include #include #include #include #include #include #include #define BUFFER 512 struct copy_file { int infile. int outfile. }. void *copy(void *arg) { int infile,outfile. int bytes_read,bytes_write,*bytes_copy_p. char buffer[BUFFER],*buffer_p. struct copy_file *file=(struct copy_file *)arg. infile=file->infile. outfile=file->outfile. /* 因为线程退出时,

所有的变量空间都要被释放,所以我们只好自己分配内存了 */
if((bytes_copy_p=(int *)malloc(sizeof(int)))==NULL)
pthread_exit(NULL). bytes_read=bytes_write=0. *bytes_copy_p=0.
/* 还记得怎么拷贝文件吗 */
while((bytes_read=read(infile,buffer,BUFFER))!=0) {
if((bytes_read==-1)amp.(errno!=EINTR))break. else
if(bytes_read>0) { buffer_p=buffer.
while((bytes_write=write(outfile,buffer_p,bytes_read))!=0) {
if((bytes_write==-1)amp.(errno!=EINTR))break. else
if(bytes_write==bytes_read)break. else if(bytes_write>0) { buffer_p
=bytes_write. bytes_read-=bytes_write. } }
if(bytes_write==-1)break. *bytes_copy_p =bytes_read. } } 100Test