子任务1：

Server.c

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <time.h>**

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

**#include <netinet/in.h>**

**#include <errno.h>**

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

**#include <arpa/inet.h>**

**#define BUFSIZE 2048**

**#define BACKLOG 10**

**#define BLKSIZE 2048**

**#define MSG\_FILENAME 1**

**#define MSG\_CONTENT 2**

**#define MSG\_ACK 3**

**#define MSG\_DONE 4**

**#define MSG\_EXCEPTION 5**

**struct msg{**

**int32\_t type;**

**int32\_t data\_len;**

**char data[];**

**};**

**void print\_usage()**

**{**

**printf("usage:\n""\tserver {listen\_port}\n");**

**}**

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

**{**

**char buf[BUFSIZE],file\_path[256];**

**struct msg \*m;**

**char \*port;**

**FILE \*file=NULL;**

**int listensock, clientsock;**

**socklen\_t len;**

**struct sockaddr\_in server\_addr, client\_addr;**

**int ret=0;**

**int addrlen;**

**int datalen;**

**int headlen;**

**time\_t start\_time,end\_time;**

**if(argc<2)**

**{**

**print\_usage();**

**return -1;**

**}**

**port=argv[1];**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=INADDR\_ANY;**

**server\_addr.sin\_port=htons(atoi(port));**

**if ((listensock = socket(AF\_INET, SOCK\_STREAM, 0)) == -1)**

**{**

**printf("socket create error=%d!",errno);**

**ret=-2;**

**goto exit3;**

**}**

**if((bind(listensock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr))) == -1)**

**{**

**printf("bind error=%d!",errno);**

**ret=-3;**

**goto exit2;**

**}**

**if( (listen(listensock, BACKLOG)) == -1 )**

**{**

**printf("listen error=%d!",errno);**

**ret=-4;**

**goto exit2;**

**}**

**addrlen = sizeof(struct sockaddr\_in);**

**clientsock = accept(listensock, (struct sockaddr\*)&client\_addr, &addrlen);**

**if(clientsock<0)**

**{**

**printf("accept error=%d\n",errno);**

**ret=-5;**

**goto exit2;**

**}**

**printf("--%s:%d--\n",inet\_ntoa(client\_addr.sin\_addr),ntohs(client\_addr.sin\_port));**

**m=(struct msg\*)buf;**

**while(1)**

**{**

**//接收消息头**

**headlen=recv(clientsock,buf,sizeof(struct msg),0);**

**if(headlen<=0)**

**{**

**printf("recv error!\n");**

**goto exit1;**

**}**

**//循环接收消息数据，长度在消息头中已知**

**if(m->data\_len>0)**

**{**

**datalen=0;**

**while(datalen<m->data\_len)**

**{**

**datalen+=recv(clientsock,m->data+datalen,m->data\_len-datalen,0);**

**}**

**}**

**//处理消息**

**if(m->type==MSG\_FILENAME)**

**{**

**start\_time=time(NULL);**

**//获取文件名**

**printf("msg type is filename %s, masglen=%d\n",m->data,m->data\_len);**

**memcpy(file\_path,m->data,m->data\_len);**

**file\_path[m->data\_len]=0;**

**printf("file path is %s\n",file\_path);**

**file=fopen(file\_path,"w");**

**if(file==NULL)**

**{**

**printf("fopen error=%d\n",errno);**

**ret=-6;**

**goto exit1;**

**}**

**}else if(m->type==MSG\_CONTENT)**

**{**

**fwrite(m->data,1,m->data\_len,file);**

**}else if(m->type==MSG\_DONE)**

**{**

**printf("MSG DONE\n");**

**goto exit0;**

**}else if(m->type==MSG\_EXCEPTION)**

**{**

**printf("MSG\_EXCEPTION\n");**

**goto exit0;**

**}**

**}**

**exit0:**

**fflush(file);**

**end\_time=time(NULL);//结束时间**

**printf("Use time: %ld s\n",end\_time-start\_time);**

**exit1:**

**fclose(file);**

**close(clientsock);**

**exit2:**

**close(listensock);**

**exit3:**

**return ret;**

**}**

**Client.c**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <errno.h>**

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

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

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

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

**#include <pthread.h>**

**#define QUESIZE 4**

**#define BLKSIZE 2048**

**#define BUFSIZE 2056**

**#define MSG\_FILENAME 1**

**#define MSG\_CONTENT 2**

**#define MSG\_ACK 3**

**#define MSG\_DONE 4**

**#define MSG\_EXCEPTION 5**

**struct msg {**

**int type;**

**int data\_len;**

**char data[];**

**};**

**char \*bufs[QUESIZE];**

**volatile int head;**

**volatile int rear;**

**volatile int count;**

**pthread\_spinlock\_t cntlock;**

**void \*work(void \*args)**

**{**

**struct msg \*m;**

**char \*file\_path,\*file\_name,\*tmp;**

**FILE \*file;**

**int alive=1;**

**int datalen;**

**int i=0;**

**file\_path=(char\*)args;**

**//从文件完整路径中截取文件名**

**tmp=strrchr(file\_path,'/');**

**file\_name=tmp?(tmp+1):file\_path;**

**printf("file\_path=%s\n",file\_path);**

**printf("file\_name=%s\n",file\_name);**

**file=fopen(file\_path,"r");**

**if(file==NULL)**

**{**

**printf("fopen error=%d\n",errno);**

**}**

**//创建MSG\_FILENAME**

**printf("create msg\_filename\n");**

**m=(struct msg\*)bufs[rear];//获得空闲缓冲区**

**m->data\_len=strlen(file\_name);**

**m->type=MSG\_FILENAME;**

**memcpy(m->data,file\_name,m->data\_len);**

**printf("no %d msg,len=%d, count=%d. This msg's type is %d\n",rear,m->data\_len,count,m->type);**

**//更新队列状态**

**rear=(rear+1)%QUESIZE;**

**pthread\_spin\_lock(&cntlock);**

**count++;**

**pthread\_spin\_unlock(&cntlock);**

**printf("update rear,rear=%d\n",rear);**

**while(alive)**

**{**

**if (head==rear)**

**{**

**//忙等待**

**while (count>=QUESIZE)**

**{**

**i++;**

**}**

**}**

**m=(struct msg\*)bufs[rear];//获得空闲缓冲区**

**//读文件**

**datalen=fread(m->data,1,BLKSIZE,file);**

**if(datalen<=0)**

**{**

**m->data\_len=0;**

**if(feof(file))**

**{//文件结束**

**m->type=MSG\_DONE;**

**printf("in thread, we have reached end of file, msg type is msg\_done\n");**

**}**

**else if(ferror(file))**

**{//读文件错误**

**printf("fread error=%d\n",errno);**

**m->type=MSG\_EXCEPTION;**

**}**

**alive=0;**

**}**

**else**

**{**

**m->data\_len=datalen;**

**m->type=MSG\_CONTENT;**

**}**

**//更新队列状态**

**printf("no %d msg,len=%d, count=%d. This msg's type is %d\n",rear,m->data\_len,count,m->type);**

**rear=(rear+1)%QUESIZE;**

**pthread\_spin\_lock(&cntlock);**

**count++;**

**pthread\_spin\_unlock(&cntlock);**

**printf("update rear,rear=%d\n",rear);**

**}**

**fclose(file);**

**printf("close file\n");**

**printf("thread work over!\n");**

**}**

**void print\_usage()**

**{**

**printf("usage:\n"**

**"\tclient {server\_ip} {server\_port} {file\_path}\n");**

**}**

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

**{**

**struct msg \*m;**

**char \*ip,\*port,\*file\_path;**

**int clientsock;**

**struct sockaddr\_in server\_addr;**

**int ret=0;**

**int msglen;**

**time\_t start\_time,end\_time;**

**int i=0;**

**pthread\_t worker;**

**if(argc<4)**

**{**

**print\_usage();**

**return -1;**

**}**

**ip=argv[1];**

**port=argv[2];**

**file\_path=argv[3];//文件完整路径**

**head=0;**

**rear=0;**

**count=0;**

**//申请缓冲区**

**for(i=0; i<QUESIZE; i++)**

**{**

**bufs[i]=(char \*)malloc(BUFSIZE\*sizeof(char));**

**if(bufs[i]==NULL)**

**{**

**printf("malloc failed\n");**

**return 0;**

**}**

**}**

**//初始化自旋锁**

**pthread\_spin\_init(&cntlock,0);**

**//创建线程**

**printf("now create thread\n");**

**ret=pthread\_create(&worker,NULL,work,file\_path);**

**if(ret!=0)**

**{**

**printf("pthread\_create error=%d\n",ret);**

**ret=-8;**

**goto exit2;**

**}**

**//建立socket连接**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=inet\_addr(ip);**

**server\_addr.sin\_port=htons(atoi(port));**

**clientsock=socket(AF\_INET,SOCK\_STREAM,0);**

**if(clientsock<0)**

**{**

**printf("socket create error=%d\n",errno);**

**ret=-2;**

**goto exit2;**

**}**

**if(connect(clientsock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr\_in))<0)**

**{**

**printf("connect error=%d\n",errno);**

**ret=-3;**

**goto exit1;**

**}**

**printf("connect over\n");**

**start\_time=time(NULL);//开始时间**

**while (1)**

**{**

**if (head==rear)**

**{**

**//忙等待**

**while (count<=0)**

**{**

**i++;**

**}**

**}**

**m=(struct msg\*)bufs[head];//获得缓冲区**

**//发送消息**

**msglen=sizeof(struct msg)+m->data\_len;**

**// printf("sendmsglength=%d, type is %d\n",msglen,m->type);**

**if(send(clientsock,m,msglen,0)<0)**

**{**

**printf("send error=%d\n",errno);**

**ret=-4;**

**goto exit1;**

**}**

**if(m->type==MSG\_DONE||m->type==MSG\_EXCEPTION)**

**{**

**printf("now get buf data %d,type is %d,len=%d\n",head,m->type,msglen);**

**// printf("send ok!\n");**

**printf("send MSG\_DONE, and goto exit0\n");**

**goto exit0;**

**}**

**//更新队列状态**

**printf("now get buf data %d,type is %d,len=%d\n",head,m->type,msglen);**

**printf("send ok!\n");**

**head=(head+1)%QUESIZE;**

**pthread\_spin\_lock(&cntlock);**

**count--;**

**pthread\_spin\_unlock(&cntlock);**

**printf("update head,head=%d\n",head);**

**}**

**exit0:**

**end\_time=time(NULL);//结束时间**

**printf("Use time: %ld s\n",end\_time-start\_time);**

**exit1:**

**close(clientsock);**

**exit2:**

**pthread\_spin\_destroy(&cntlock);**

**//释放缓冲区**

**for(i=0; i<QUESIZE; i++)**

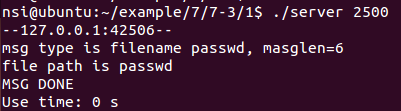
**{**

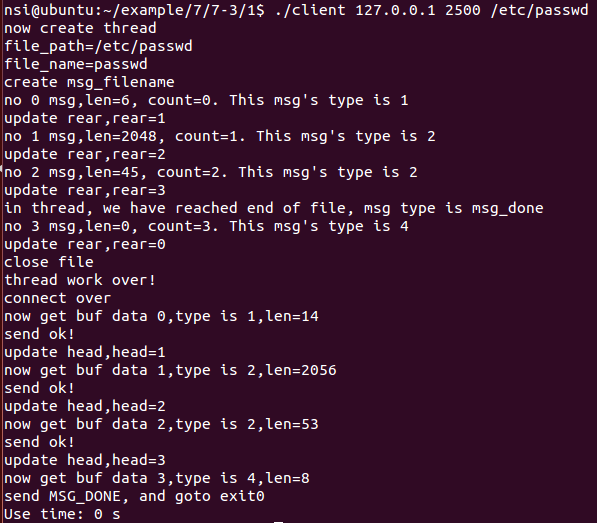
**free(bufs[i]);**

**}**

**return ret;**

**}**





子任务2：

Client.c

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <errno.h>**

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

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

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

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

**#include <pthread.h>**

**#define QUESIZE 4**

**#define BLKSIZE 2048**

**#define BUFSIZE 2048**

**#define MSG\_FILENAME 1**

**#define MSG\_CONTENT 2**

**#define MSG\_ACK 3**

**#define MSG\_DONE 4**

**#define MSG\_EXCEPTION 5**

**struct msg {**

**int type;**

**int data\_len;**

**char data[];**

**};**

**char \*bufs[QUESIZE];**

**volatile int head;**

**volatile int rear;**

**volatile int count;**

**//pthread\_spinlock\_t cntlock;**

**pthread\_mutex\_t cntlock;**

**pthread\_cond\_t cntwait;**

**void update\_headqueue\_stat();**

**void update\_rearqueue\_stat();**

**void \*work(void \*args)**

**{**

**struct msg \*m;**

**char \*file\_path,\*file\_name,\*tmp;**

**FILE \*file;**

**int alive=1;**

**int datalen;**

**int i=0;**

**file\_path=(char\*)args;**

**//从文件完整路径中截取文件名**

**tmp=strrchr(file\_path,'/');**

**file\_name=tmp?(tmp+1):file\_path;**

**printf("file\_path=%s\n",file\_path);**

**printf("file\_name=%s\n",file\_name);**

**file=fopen(file\_path,"r");**

**if(file==NULL)**

**{**

**printf("fopen error=%d\n",errno);**

**}**

**//创建MSG\_FILENAME**

**printf("create msg\_filename\n");**

**m=(struct msg\*)bufs[rear];//获得空闲缓冲区**

**m->data\_len=strlen(file\_name);**

**m->type=MSG\_FILENAME;**

**memcpy(m->data,file\_name,m->data\_len);**

**printf("no %d msg,len=%d, count=%d. This msg's type is %d\n",rear,m->data\_len,count,m->type);**

**//更新队列状态**

**rear=(rear+1)%QUESIZE;**

**// pthread\_spin\_lock(&cntlock);**

**pthread\_mutex\_lock(&cntlock);**

**count++;**

**if(count>=1)**

**{**

**pthread\_cond\_signal(&cntwait);**

**}**

**pthread\_mutex\_unlock(&cntlock);**

**// pthread\_spin\_unlock(&cntlock);**

**printf("update rear,rear=%d\n",rear);**

**//busy wait, get data and send circle**

**while(alive)**

**{**

**/\* if (head==rear)**

**{**

**//忙等待**

**while (count>=QUESIZE)**

**{**

**i++;**

**}**

**}**

**\*/**

**pthread\_mutex\_lock(&cntlock);**

**while(head==rear&&count>=QUESIZE)**

**{**

**pthread\_cond\_wait(&cntwait,&cntlock);**

**}**

**pthread\_mutex\_unlock(&cntlock);**

**m=(struct msg\*)bufs[rear];//获得空闲缓冲区**

**//读文件**

**datalen=fread(m->data,1,BLKSIZE,file);**

**if(datalen<=0)**

**{**

**m->data\_len=0;**

**if(feof(file))**

**{//文件结束**

**m->type=MSG\_DONE;**

**printf("in thread, file send over,msg type is msg\_done\n");**

**}**

**else if(ferror(file))**

**{//读文件错误**

**printf("fread error=%d\n",errno);**

**m->type=MSG\_EXCEPTION;**

**}**

**alive=0;**

**}**

**else**

**{**

**m->data\_len=datalen;**

**m->type=MSG\_CONTENT;**

**}**

**//更新队列状态**

**printf("no %d msg,len=%d, count=%d. This msg's type is %d\n",rear,m->data\_len,count,m->type);**

**rear=(rear+1)%QUESIZE;**

**//pthread\_spin\_lock(&cntlock);**

**pthread\_mutex\_lock(&cntlock);**

**count++;**

**if(count>=1)**

**{**

**pthread\_cond\_signal(&cntwait);**

**}**

**pthread\_mutex\_unlock(&cntlock);**

**//pthread\_spin\_unlock(&cntlock);**

**printf("update rear,rear=%d\n",rear);**

**}**

**fclose(file);**

**printf("close file\n");**

**printf("thread work over!\n");**

**}**

**void print\_usage()**

**{**

**printf("usage:\n"**

**"\tclient {server\_ip} {server\_port} {file\_path}\n");**

**}**

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

**{**

**struct msg \*m;**

**char \*ip,\*port,\*file\_path;**

**int clientsock;**

**struct sockaddr\_in server\_addr;**

**int ret=0;**

**int msglen;**

**time\_t start\_time,end\_time;**

**int i=0;**

**pthread\_t worker;**

**if(argc!=4)**

**{**

**print\_usage();**

**return -1;**

**}**

**ip=argv[1];**

**port=argv[2];**

**file\_path=argv[3];//文件完整路径**

**head=0;**

**rear=0;**

**count=0;**

**//申请缓冲区**

**for(i=0; i<QUESIZE; i++)**

**{**

**bufs[i]=(char \*)malloc(BUFSIZE\*sizeof(char));**

**if(bufs[i]==NULL)**

**{**

**printf("malloc failed\n");**

**return 0;**

**}**

**}**

**//初始化互斥锁和条件变量**

**//pthread\_spin\_init(&cntlock,0);**

**pthread\_mutex\_init(&cntlock,0);**

**pthread\_cond\_init(&cntwait,NULL);**

**//创建线程**

**printf("now create thread\n");**

**ret=pthread\_create(&worker,NULL,work,file\_path);**

**if(ret!=0)**

**{**

**printf("pthread\_create error=%d\n",ret);**

**ret=-8;**

**goto exit2;**

**}**

**//建立socket连接**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=inet\_addr(ip);**

**server\_addr.sin\_port=htons(atoi(port));**

**clientsock=socket(AF\_INET,SOCK\_STREAM,0);**

**if(clientsock<0)**

**{**

**printf("socket create error=%d\n",errno);**

**ret=-2;**

**goto exit2;**

**}**

**if(connect(clientsock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr\_in))<0)**

**{**

**printf("connect error=%d\n",errno);**

**ret=-3;**

**goto exit1;**

**}**

**printf("connect over\n");**

**start\_time=time(NULL);//开始时间**

**while (1)**

**{**

**/\***

**if (head==rear)**

**{**

**//忙等待**

**while (count<=0)**

**{**

**i++;**

**}**

**}**

**\*/**

**pthread\_mutex\_lock(&cntlock);**

**while(head==rear&&count<=0)**

**{**

**pthread\_cond\_wait(&cntwait,&cntlock);**

**}**

**pthread\_mutex\_unlock(&cntlock);**

**//get data and send out**

**m=(struct msg\*)bufs[head];//获得缓冲区**

**//发送消息**

**msglen=sizeof(struct msg)+m->data\_len;**

**// printf("sendmsglength=%d, type is %d\n",msglen,m->type);**

**if(send(clientsock,m,msglen,0)<0)**

**{**

**printf("send error=%d\n",errno);**

**ret=-4;**

**goto exit1;**

**}**

**if(m->type==MSG\_DONE||m->type==MSG\_EXCEPTION)**

**{**

**printf("now get buf data %d,type is %d,len=%d\n",head,m->type,msglen);**

**// printf("send ok!\n");**

**printf("send MSG\_DONE, and goto exit0\n");**

**goto exit0;**

**}**

**//更新队列状态**

**printf("now get buf data %d,type is %d,len=%d\n",head,m->type,msglen);**

**printf("send ok!\n");**

**head=(head+1)%QUESIZE;**

**//pthread\_spin\_lock(&cntlock);**

**pthread\_mutex\_lock(&cntlock);**

**count--;**

**//检查是否满足条件，以唤醒其他线程**

**if(count<=QUESIZE-1)**

**{**

**pthread\_cond\_signal(&cntwait);**

**}**

**pthread\_mutex\_unlock(&cntlock);**

**//pthread\_spin\_unlock(&cntlock);**

**printf("update head,head=%d\n",head);**

**}**

**exit0:**

**end\_time=time(NULL);//结束时间**

**printf("Use time: %ld s\n",end\_time-start\_time);**

**exit1:**

**close(clientsock);**

**exit2:**

**//pthread\_spin\_destroy(&cntlock);**

**pthread\_mutex\_destroy(&cntlock);**

**pthread\_cond\_destroy(&cntwait);**

**//释放缓冲区**

**for(i=0; i<QUESIZE; i++)**

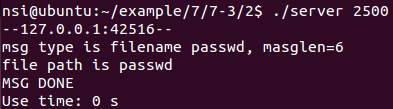
**{**

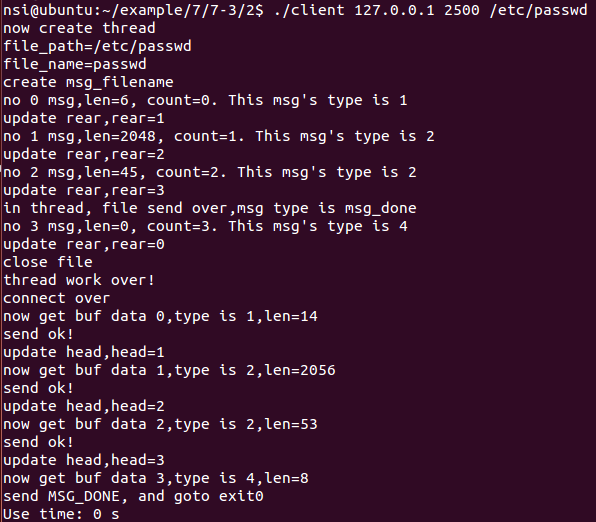
**free(bufs[i]);**

**}**

**return ret;**

**}**





子任务3：

Client.c

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <errno.h>**

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

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

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

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

**#include <pthread.h>**

**#include <semaphore.h>**

**#define QUESIZE 4**

**#define BLKSIZE 2048**

**#define BUFSIZE 2048**

**#define MSG\_FILENAME 1**

**#define MSG\_CONTENT 2**

**#define MSG\_ACK 3**

**#define MSG\_DONE 4**

**#define MSG\_EXCEPTION 5**

**struct msg {**

**int type;**

**int data\_len;**

**char data[];**

**};**

**char \*bufs[QUESIZE];**

**volatile int head;**

**volatile int rear;**

**//volatile int count;**

**//pthread\_spinlock\_t cntlock;**

**sem\_t count,empty\_count;**

**void \*work(void \*args)**

**{**

**struct msg \*m;**

**char \*file\_path,\*file\_name,\*tmp;**

**FILE \*file;**

**int alive=1;**

**int datalen;**

**int i=0;**

**file\_path=(char\*)args;**

**//从文件完整路径中截取文件名**

**tmp=strrchr(file\_path,'/');**

**file\_name=tmp?(tmp+1):file\_path;**

**printf("file\_path=%s\n",file\_path);**

**printf("file\_name=%s\n",file\_name);**

**file=fopen(file\_path,"r");**

**if(file==NULL)**

**{**

**printf("fopen error=%d\n",errno);**

**}**

**//创建MSG\_FILENAME**

**printf("create msg\_filename\n");**

**sem\_wait(&empty\_count);**

**m=(struct msg\*)bufs[rear];//获得空闲缓冲区**

**m->data\_len=strlen(file\_name);**

**m->type=MSG\_FILENAME;**

**memcpy(m->data,file\_name,m->data\_len);**

**printf("no %d msg,len=%d,. This msg's type is %d\n",rear,m->data\_len,m->type);**

**//更新队列状态**

**rear=(rear+1)%QUESIZE;**

**//pthread\_spin\_lock(&cntlock);**

**sem\_post(&count);**

**//pthread\_spin\_unlock(&cntlock);**

**printf("update rear,rear=%d\n",rear);**

**while(alive)**

**{**

**/\*if (head==rear)**

**{**

**//忙等待**

**while (count>=QUESIZE)**

**{**

**i++;**

**}**

**}\*/**

**sem\_wait(&empty\_count);**

**m=(struct msg\*)bufs[rear];//获得空闲缓冲区**

**//读文件**

**datalen=fread(m->data,1,BLKSIZE,file);**

**if(datalen<=0)**

**{**

**m->data\_len=0;**

**if(feof(file))**

**{//文件结束**

**m->type=MSG\_DONE;**

**printf("in thread, file send over,msg type is msg\_done\n");**

**}**

**else if(ferror(file))**

**{//读文件错误**

**printf("fread error=%d\n",errno);**

**m->type=MSG\_EXCEPTION;**

**}**

**alive=0;**

**}**

**else**

**{**

**m->data\_len=datalen;**

**m->type=MSG\_CONTENT;**

**}**

**//更新队列状态**

**printf("no %d msg,len=%d. This msg's type is %d\n",rear,m->data\_len,m->type);**

**rear=(rear+1)%QUESIZE;**

**//pthread\_spin\_lock(&cntlock);**

**sem\_post(&count);**

**//pthread\_spin\_unlock(&cntlock);**

**printf("update rear,rear=%d\n",rear);**

**}**

**fclose(file);**

**printf("close file\n");**

**printf("thread work over!\n");**

**}**

**void print\_usage()**

**{**

**printf("usage:\n"**

**"\tclient {server\_ip} {server\_port} {file\_path}\n");**

**}**

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

**{**

**struct msg \*m;**

**char \*ip,\*port,\*file\_path;**

**int clientsock;**

**struct sockaddr\_in server\_addr;**

**int ret=0;**

**int msglen;**

**time\_t start\_time,end\_time;**

**int i=0;**

**pthread\_t worker;**

**if(argc!=4)**

**{**

**print\_usage();**

**return -1;**

**}**

**ip=argv[1];**

**port=argv[2];**

**file\_path=argv[3];//文件完整路径**

**head=0;**

**rear=0;**

**// count=0;**

**//申请缓冲区**

**for(i=0; i<QUESIZE; i++)**

**{**

**bufs[i]=(char \*)malloc(BUFSIZE\*sizeof(char));**

**if(bufs[i]==NULL)**

**{**

**printf("malloc failed\n");**

**return 0;**

**}**

**}**

**//初始化信号量**

**//pthread\_spin\_init(&cntlock,0);**

**sem\_init(&count,0,0);**

**sem\_init(&empty\_count,0,QUESIZE);**

**//创建线程**

**printf("now create thread\n");**

**ret=pthread\_create(&worker,NULL,work,file\_path);**

**if(ret!=0)**

**{**

**printf("pthread\_create error=%d\n",ret);**

**ret=-8;**

**goto exit2;**

**}**

**//建立socket连接**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=inet\_addr(ip);**

**server\_addr.sin\_port=htons(atoi(port));**

**clientsock=socket(AF\_INET,SOCK\_STREAM,0);**

**if(clientsock<0)**

**{**

**printf("socket create error=%d\n",errno);**

**ret=-2;**

**goto exit2;**

**}**

**if(connect(clientsock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr\_in))<0)**

**{**

**printf("connect error=%d\n",errno);**

**ret=-3;**

**goto exit1;**

**}**

**printf("connect over\n");**

**start\_time=time(NULL);//开始时间**

**while (1)**

**{**

**/\*if (head==rear)**

**{**

**//忙等待**

**while (count<=0)**

**{**

**i++;**

**}**

**}\*/**

**sem\_wait(&count);**

**m=(struct msg\*)bufs[head];//获得缓冲区**

**//发送消息**

**msglen=sizeof(struct msg)+m->data\_len;**

**// printf("sendmsglength=%d, type is %d\n",msglen,m->type);**

**if(send(clientsock,m,msglen,0)<0)**

**{**

**printf("send error=%d\n",errno);**

**ret=-4;**

**goto exit1;**

**}**

**if(m->type==MSG\_DONE||m->type==MSG\_EXCEPTION)**

**{**

**printf("now get buf data %d,type is %d,len=%d\n",head,m->type,msglen);**

**// printf("send ok!\n");**

**printf("send MSG\_DONE, and goto exit0\n");**

**goto exit0;**

**}**

**//更新队列状态**

**printf("now get buf data %d,type is %d,len=%d\n",head,m->type,msglen);**

**printf("send ok!\n");**

**head=(head+1)%QUESIZE;**

**//pthread\_spin\_lock(&cntlock);**

**sem\_post(&empty\_count);**

**//pthread\_spin\_unlock(&cntlock);**

**printf("update head,head=%d\n",head);**

**}**

**exit0:**

**end\_time=time(NULL);//结束时间**

**printf("Use time: %ld s\n",end\_time-start\_time);**

**exit1:**

**close(clientsock);**

**exit2:**

**//pthread\_spin\_destroy(&cntlock);**

**sem\_destroy(&count);**

**sem\_destroy(&empty\_count);**

**//释放缓冲区**

**for(i=0; i<QUESIZE; i++)**

**{**

**free(bufs[i]);**

**}**

**return ret;**

**}**

子任务4：

Server.c

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <errno.h>**

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

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

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

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

**#include <pthread.h>**

**#include <semaphore.h>**

**#define QUESIZE 4**

**#define BACKLOG 10**

**#define BLKSIZE 2048**

**#define BUFSIZE 2048**

**#define MSG\_FILENAME 1**

**#define MSG\_CONTENT 2**

**#define MSG\_ACK 3**

**#define MSG\_DONE 4**

**#define MSG\_EXCEPTION 5**

**struct msg {**

**int32\_t type;**

**int32\_t data\_len;**

**char data[];**

**};**

**char \*bufs[QUESIZE];**

**volatile int head;**

**volatile int rear;**

**sem\_t count,empty\_count;**

**void \*writefile(void \*args)**

**{**

**struct msg \*m;**

**char file\_path[256];**

**FILE \*fp=NULL;**

**time\_t start\_time,end\_time;**

**while (1) {**

**sem\_wait(&count);**

**printf("wait buff,now write to buff. head=%d\n",head);**

**m=(struct msg\*)bufs[head];//获得缓冲区**

**printf("type is %d\n",m->type);**

**//处理消息**

**if(m->type==MSG\_FILENAME)**

**{**

**start\_time=time(NULL);//开始时间**

**//获取文件名**

**strcpy(file\_path,m->data);**

**printf("file\_path is %s\n",file\_path);**

**//打开文件**

**fp=fopen(file\_path,"w");**

**if(fp==NULL)**

**{**

**printf("fopen error=%d\n",errno);**

**return;**

**}**

**}**

**else if(m->type==MSG\_CONTENT)**

**{**

**printf("msg len=%d\n",m->data\_len);**

**printf("msg is \n%s\n",m->data);**

**fwrite(m->data,1,m->data\_len,fp);**

**}**

**else if(m->type==MSG\_DONE)**

**{**

**printf("MSG\_DONE\n");**

**goto exit0;**

**}**

**else if(m->type==MSG\_EXCEPTION)**

**{**

**printf("MSG\_EXCEPTION\n");**

**goto exit0;**

**}**

**//更新队列状态**

**head=(head+1)%QUESIZE;**

**sem\_post(&empty\_count);**

**}**

**exit0:**

**fflush(fp);**

**end\_time=time(NULL);//结束时间**

**printf("Use time: %ld s\n",end\_time-start\_time);**

**fclose(fp);**

**}**

**void print\_usage()**

**{**

**printf("usage:\n"**

**"\tserver {listen\_port}\n");**

**}**

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

**{**

**char buf[BUFSIZE];**

**char file\_path[256];**

**char \*port;**

**FILE \*fp=NULL;**

**struct msg \*m,\*pmsg;**

**struct sockaddr\_in server\_addr, client\_addr;**

**int listensock,clientsock;**

**int ret=0;**

**int addrlen=0;**

**int datalen=0;**

**int headlen=0;**

**int i=0;**

**pthread\_t thread\_write;**

**int alive=1;**

**if(argc<2)**

**{**

**print\_usage();**

**return -1;**

**}**

**port=argv[1];**

**head=0;**

**rear=0;**

**//申请缓冲区**

**for(i=0; i<QUESIZE; i++)**

**{**

**bufs[i]=(char \*)malloc(BUFSIZE\*sizeof(char));**

**if(bufs[i]==NULL)**

**{**

**printf("malloc failed\n");**

**}**

**}**

**//初始化信号量**

**sem\_init(&count,0,0);**

**sem\_init(&empty\_count,0,QUESIZE);**

**//创建线程**

**ret=pthread\_create(&thread\_write,NULL,writefile,NULL);**

**if(ret!=0)**

**{**

**printf("pthread\_create error=%d\n",ret);**

**ret=-8;**

**goto exit3;**

**}**

**//建立socket连接**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY);**

**server\_addr.sin\_port=htons(atoi(port));**

**listensock=socket(AF\_INET,SOCK\_STREAM,0);**

**if(listensock<0)**

**{**

**printf("socket create error=%d\n",errno);**

**ret=-2;**

**goto exit3;**

**}**

**if(bind(listensock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr\_in))<0)**

**{**

**printf("bind error=%d\n",errno);**

**ret=-3;**

**goto exit2;**

**}**

**if(listen(listensock,BACKLOG)<0)**

**{**

**printf("listen error=%d\n",errno);**

**ret=-4;**

**goto exit2;**

**}**

**addrlen=sizeof(struct sockaddr\_in);**

**clientsock=accept(listensock,(struct sockaddr\*)&client\_addr,&addrlen);**

**if(clientsock<0)**

**{**

**printf("accept error=%d\n",errno);**

**ret=-5;**

**goto exit2;**

**}**

**printf("--%s:%d--\n",inet\_ntoa(client\_addr.sin\_addr),ntohs(client\_addr.sin\_port));**

**//接收消息**

**while (alive)**

**{**

**sem\_wait(&empty\_count);**

**m=(struct msg\*)bufs[rear];//获得空闲缓冲区**

**//接收消息头**

**headlen=recv(clientsock,m,sizeof(struct msg),0);**

**if(headlen<=0)**

**{**

**goto exit1;**

**}**

**//接收消息数据**

**if(m->data\_len>0)**

**{**

**datalen=0;**

**while(datalen < m->data\_len)**

**{**

**datalen+=recv(clientsock,m->data,m->data\_len,0);**

**}**

**}**

**//更新队列状态**

**rear=(rear+1)%QUESIZE;**

**sem\_post(&count);**

**}**

**pthread\_join(thread\_write,NULL);**

**exit1:**

**close(clientsock);**

**exit2:**

**close(listensock);**

**exit3:**

**//销毁信号量**

**sem\_destroy(&count);**

**sem\_destroy(&empty\_count);**

**//释放缓冲区**

**for(i=0; i<QUESIZE; i++)**

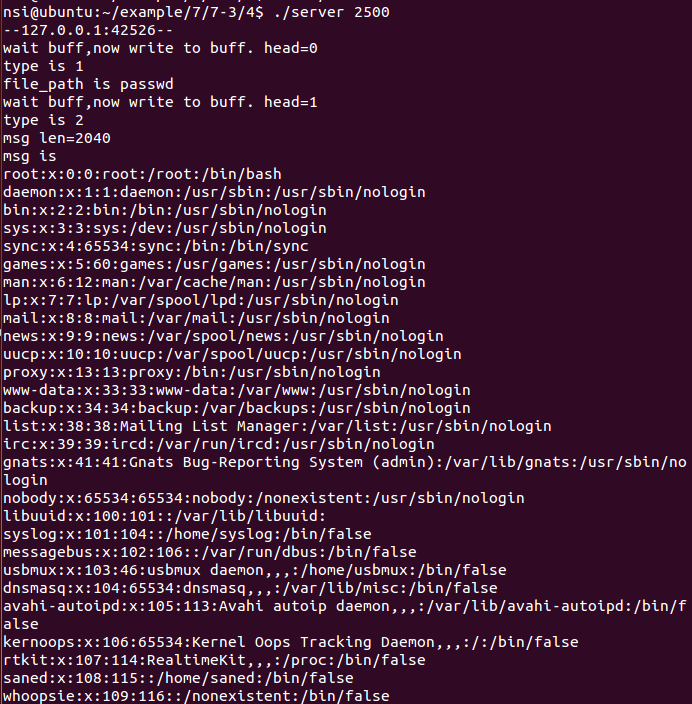
**{**

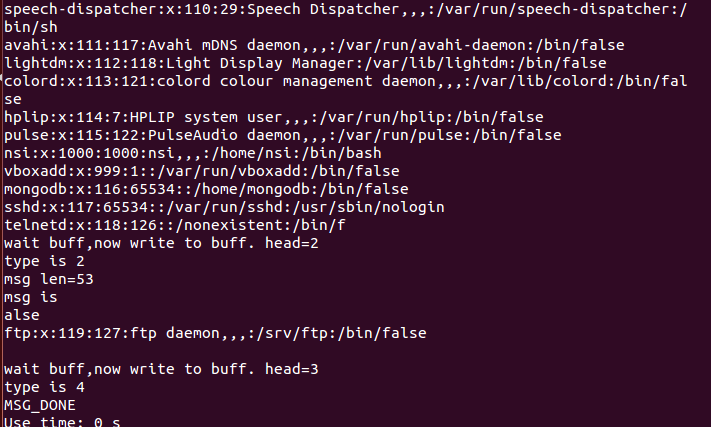
**free(bufs[i]);**

**}**

**return ret;**

**}**





子任务6：

Server.c

**#include <stdio.h>**

**#include <string.h>**

**#include <errno.h>**

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

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

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

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

**#define BACKLOG 64**

**#define BLKSIZE 2048**

**#define BUFSIZE 2056**

**#define MSG\_FILENAME 1**

**#define MSG\_CONTENT 2**

**#define MSG\_ACK 3**

**#define MSG\_DONE 4**

**#define MSG\_EXCEPTION 5**

**struct msg {**

**int32\_t type;**

**int32\_t data\_len;**

**char data[];**

**};**

**void print\_usage()**

**{**

**printf("usage:\n"**

**"\tserver {listen\_port}\n");**

**}**

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

**{**

**char buf[BUFSIZE],buf2[BUFSIZE],file\_path[256];**

**struct msg \*m;**

**char \*port;**

**FILE \*file=NULL;**

**struct sockaddr\_in server\_addr,client\_addr;**

**int serversock;**

**int ret=0;**

**int addrlen;**

**int msglen;**

**time\_t start\_time,end\_time;**

**if(argc!=2)**

**{**

**print\_usage();**

**return -1;**

**}**

**port=argv[1];**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=INADDR\_ANY;**

**server\_addr.sin\_port=htons(atoi(port));**

**if((serversock=socket(AF\_INET,SOCK\_DGRAM,0))<0)**

**{**

**printf("socket create error=%d\n",errno);**

**ret=-2;**

**goto exit3;**

**}**

**if(bind(serversock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr\_in))<0)**

**{**

**printf("bind error=%d\n",errno);**

**ret=-3;**

**goto exit2;**

**}**

**//MSG\_ACK**

**m=(struct msg\*)buf2;**

**m->type=MSG\_ACK;**

**m->data\_len=0;**

**m=(struct msg\*)buf;**

**addrlen=sizeof(struct sockaddr\_in);**

**while(1)**

**{**

**msglen=recvfrom(serversock,buf,BUFSIZE,0,(struct sockaddr\*)&client\_addr,&addrlen);**

**sendto(serversock,buf2,sizeof(struct msg),0,(struct sockaddr\*)&client\_addr,addrlen);**

**if(msglen<0)**

**{**

**printf("recv error!\n");**

**goto exit1;**

**}**

**if(m->type==MSG\_FILENAME)**

**{**

**start\_time=time(NULL);**

**memcpy(file\_path,m->data,m->data\_len);**

**file\_path[m->data\_len]=0;**

**file=fopen(file\_path,"w");**

**if(file==NULL)**

**{**

**printf("fopen error=%d\n",errno);**

**ret=-6;**

**goto exit2;**

**}**

**}**

**else if(m->type==MSG\_CONTENT)**

**{**

**fwrite(m->data,1,m->data\_len,file);**

**}**

**else if(m->type==MSG\_DONE)**

**{**

**printf("MSG\_DONE\n");**

**goto exit0;**

**}**

**else if(m->type==MSG\_EXCEPTION)**

**{**

**printf("MSG\_EXCEPTION\n");**

**goto exit0;**

**}**

**}**

**exit0:**

**fflush(file);**

**end\_time=time(NULL);**

**printf("Use time: %ld s\n",end\_time-start\_time);**

**exit1:**

**fclose(file);**

**exit2:**

**close(serversock);**

**exit3:**

**return ret;**

**}**

**Client.c**

**#include <stdio.h>**

**#include <string.h>**

**#include <errno.h>**

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

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

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

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

**#define BLKSIZE 2048**

**#define BUFSIZE 2056**

**#define MSG\_FILENAME 1**

**#define MSG\_CONTENT 2**

**#define MSG\_ACK 3**

**#define MSG\_DONE 4**

**#define MSG\_EXCEPTION 5**

**struct msg {**

**int32\_t type;**

**int32\_t data\_len;**

**char data[];**

**};**

**void print\_usage()**

**{**

**printf("usage:\n"**

**"\tclient {server\_ip} {server\_port} {file\_path}\n");**

**}**

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

**{**

**char buf[BUFSIZE],buf2[BUFSIZE];**

**struct msg \*m;**

**char \*ip,\*port,\*file\_path,\*file\_name,\*tmp;**

**FILE \*file;**

**int clientsock;**

**struct sockaddr\_in server\_addr;**

**int ret=0;**

**int datalen;**

**int msglen;**

**time\_t start\_time,end\_time;**

**int alive=1;**

**if(argc<4)**

**{**

**print\_usage();**

**return -1;**

**}**

**ip=argv[1];**

**port=argv[2];**

**file\_path=argv[3];//文件完整路径**

**//从文件完整路径中截取文件名**

**tmp=strrchr(file\_path,'/');**

**file\_name=tmp?(tmp+1):file\_path;**

**file=fopen(file\_path,"r");**

**if(file==NULL)**

**{**

**printf("fopen error=%d\n",errno);**

**return -5;**

**}**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=inet\_addr(ip);**

**server\_addr.sin\_port=htons(atoi(port));**

**clientsock=socket(AF\_INET,SOCK\_DGRAM,0);//UDP**

**if(clientsock<0)**

**{**

**printf("socket create error=%d\n",errno);**

**ret=-2;**

**goto exit2;**

**}**

**start\_time=time(NULL);//开始时间**

**//创建并发送MSG\_FILENAME**

**m=(struct msg\*)buf;**

**m->type=MSG\_FILENAME;**

**m->data\_len=strlen(file\_name);**

**memcpy(m->data,file\_name,m->data\_len);**

**msglen=sizeof(struct msg)+m->data\_len;**

**if(sendto(clientsock,buf,msglen,0,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr\_in))<0)**

**{**

**printf("send MSG\_FILENAME error=%d\n",errno);**

**ret=-4;**

**goto exit1;**

**}**

**//接收回复**

**recvfrom(clientsock,buf2,BUFSIZE,0,NULL,NULL);**

**//创建并发送MSG\_CONTENT**

**m=(struct msg\*)buf;**

**m->type=MSG\_CONTENT;**

**while (alive)**

**{**

**//读文件**

**datalen=fread(m->data,1,BLKSIZE,file);**

**if(datalen<=0)**

**{**

**if(feof(file))**

**{**

**printf("send MSG\_DONE\n");**

**m->type=MSG\_DONE;**

**m->data\_len=0;**

**}**

**else if(ferror(file))**

**{**

**printf("fread error=%d\n",errno);**

**ret=-6;**

**m->type=MSG\_EXCEPTION;**

**m->data\_len=0;**

**}**

**alive=0;**

**}**

**else**

**{**

**m->data\_len=datalen;**

**}**

**//发送消息**

**msglen=sizeof(struct msg)+m->data\_len;**

**if(sendto(clientsock,buf,msglen,0,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr\_in))<0)**

**{**

**printf("send MSG\_CONTENT error=%d\n",errno);**

**ret=-4;**

**goto exit1;**

**}**

**//接收回复**

**recvfrom(clientsock,buf2,BUFSIZE,0,NULL,NULL);**

**}**

**exit0:**

**end\_time=time(NULL);//结束时间**

**printf("Use time: %ld s\n",end\_time-start\_time);**

**exit1:**

**close(clientsock);**

**exit2:**

**fclose(file);**

**return ret;**

**}**

子任务7：

Server.c

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

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

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

**#include <netinet/in.h>**

**#include <string.h>**

**#include <errno.h>**

**#define BACKLOG 10**

**#define BUFSIZE 1024**

**void print\_usage();**

**void echo\_fd(int fd);**

**void echo\_cmd();**

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

**{**

**int sockfd,listenfd,connfd;**

**struct sockaddr\_in server\_addr,client\_addr;**

**char \*port;**

**if(argc!=2)**

**{**

**print\_usage();**

**return -1;**

**}**

**port=argv[1];**

**if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))<0)**

**{**

**printf("socket error=%d",errno);**

**return -1;**

**}**

**memset(&server\_addr,0,sizeof(struct sockaddr\_in));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY);**

**server\_addr.sin\_port=htons(atoi(port));**

**if((bind(sockfd,(struct sockaddr\*)&server\_addr,sizeof(server\_addr)))<0)**

**{**

**printf("bind error=%d\n",errno);**

**return -1;**

**}**

**if(listen(sockfd,BACKLOG)==-1)**

**{**

**printf("listen error=%d\n",errno);**

**return -1;**

**}**

**while(1)**

**{**

**int ret;**

**fd\_set read\_set;**

**FD\_ZERO(&read\_set);**

**FD\_SET(STDIN\_FILENO,&read\_set);**

**FD\_SET(sockfd,&read\_set);**

**ret=select(sockfd+1,&read\_set,NULL,NULL,NULL);**

**printf("ret=%d\n",ret);**

**if(ret==-1)**

**{**

**printf("select error!\n");**

**return 0;**

**}else if(ret>0){**

**if(FD\_ISSET(sockfd,&read\_set))**

**echo\_fd(sockfd);**

**if(FD\_ISSET(STDIN\_FILENO,&read\_set))**

**echo\_cmd();**

**}**

**}**

**return 0;**

**}**

**void print\_usage()**

**{**

**printf("usage:\n""\tserver {listen port}\n");**

**}**

**void echo\_fd(int fd)**

**{**

**struct sockaddr\_in client\_addr;**

**int addrlen=sizeof(client\_addr);**

**int connfd=accept(fd,(struct sockaddr\*)&client\_addr,&addrlen);**

**char buf[BUFSIZE]={0};**

**while(1)**

**{**

**int nread=read(connfd,buf,sizeof(buf));**

**if(nread==-1)**

**{**

**if(errno==EINTR)**

**continue;**

**else**

**exit(1);**

**}else if(nread==0)**

**{**

**close(connfd);**

**return;**

**}else{**

**printf("read from client, the msg is %s\n",buf);**

**close(connfd);**

**break;**

**}**

**}**

**}**

**void echo\_cmd()**

**{**

**char buf[100]={0};**

**int nread=read(STDIN\_FILENO,buf,sizeof(buf));**

**if(strncmp(buf,"exit",4)==0)**

**{**

**exit(1);**

**}**

**else**

**{**

**printf("read from stdin, the msg is %s\n",buf);**

**}**

**}**

**Client.c**

**#include <stdio.h>**

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

**#include <netinet/in.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**void print\_usage()**

**{**

**printf("usage:\n"**

**"\t./client ip port\n");**

**}**

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

**{**

**char \*ip,\*port;**

**int clientfd;**

**struct sockaddr\_in server\_addr;**

**int flag=1;**

**if(argc != 3)**

**{**

**print\_usage();**

**return -1;**

**}**

**ip=argv[1];**

**port=argv[2];**

**clientfd=socket(AF\_INET,SOCK\_STREAM,0);**

**bzero(&server\_addr,sizeof(server\_addr));**

**server\_addr.sin\_family=AF\_INET;**

**server\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY);**

**server\_addr.sin\_port=htons(atoi(port));**

**connect(clientfd,(struct sockaddr\*)&server\_addr,sizeof(server\_addr));**

**char buf[100]={0};**

**write(clientfd,"hello select test!",strlen("hello select test!"));**

**int nread=read(clientfd,buf,sizeof(buf));**

**if(nread==0)**

**{**

**printf("%s\n","server closed!");**

**close(clientfd);**

**}**

**return 0;**

**}**

