/\*

\* usock\_server : listen on a Unix socket ; fork ;

\* child does lookup ; replies down same socket

\* argv[1] is the name of the local datafile

\* PORT is defined in dict.h

\*/

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <unistd.h>

#include <errno.h>

#include "dict.h"

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

struct sockaddr\_un server,client;

int sd,cd,n;

Dictrec tryit;

if (argc != 3) {

fprintf(stderr,"Usage : %s <dictionary source>""<Socket name>\n",argv[0]);

exit(errno);

}

/\* Setup socket.

\* Fill in code. \*/

if((sd=socket(AF\_UNIX,SOCK\_STREAM,0))==-1){

perror("socket");

exit(1);

}

/\* Initialize address.

\* Fill in code. \*/

server.sun\_family = AF\_UNIX;

strcpy(server.sun\_path,argv[2]);

//unlink(server.sun\_path);

/////////

//n=strlen(server.sun\_path)+sizeof(server.sun\_family);

/////////

/\* Name and activate the socket.

\* Fill in code. \*/

if( bind(sd, (struct sockaddr \*)&server,/\*n\*/sizeof(struct sockaddr\_un)) < 0) {

perror("bind");

exit(1);

}

if(listen(sd,10)==-1){

perror("listen");

exit(1);

}

/\* main loop : accept connection; fork a child to have dialogue \*/

for (;;) {

/\* Wait for a connection.

\* Fill in code. \*/

cd=accept(sd,(struct sockaddr\*)&client,/\*sizeof(struct sockaddr\_un)\*/&n);

if(cd==-1){

perror("accept");

exit(1);

}

/\* Handle new client in a subprocess. \*/

switch (fork()) {

case -1 :

DIE("fork");

case 0 :

close (sd); /\* Rendezvous socket is for parent only. \*/

/\* Get next request.

\* Fill in code. \*/

while (recv(cd,&tryit,sizeof(Dictrec),0)) {

/\* Lookup request. \*/

switch(lookup(&tryit,argv[1]) ) {

/\* Write response back to client. \*/

case FOUND:

/\* Fill in code. \*/

send(cd,&tryit,sizeof(Dictrec),0);

break;

case NOTFOUND:

/\* Fill in code. \*/

strcpy(tryit.text,"XXXX");

send(cd,&tryit,sizeof(Dictrec),0);

break;

case UNAVAIL:

DIE(argv[1]);

} /\* end lookup switch \*/

} /\* end of client dialog \*/

/\* Terminate child process. It is done. \*/

exit(0);

/\* Parent continues here. \*/

default :

close(cd);

break;

} /\* end fork switch \*/

} /\* end forever loop \*/

} /\* end main \*/