Socket Programming Lec 2

Rishi Kant

Review of Socket programming

- Decide which type of socket – stream or datagram. Based on type create socket using socket() function
- For datagram - Define the IP and port you wish to connect from, and connect to in 2 separate sockaddr_in structures
- For stream – Define the IP and port you wish to listen on (server) or connect to (client) in a sockaddr_in structure
- For datagrams and stream servers, bind the sockaddr_in structure to the socket allocated using the bind() function

Review of Socket programming

- For servers, use the listen() function to signal the OS to monitor incoming connections and use the accept() function to wait for a client
- For clients, use the connect() function to initiate the 3-way handshake and set up the stream connection to the server
- For streams, use the send() and recv() functions to transmit and receive bytes
- For datagram, use the sendto() and recvfrom() functions to transmit and receive packets

Example from book (pg 33)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define PORT 5432
#define MAX_LINE 256

int main (int argc, char **argv) {
  FILE *fp;
  struct hostent *hp;
  struct sockaddr_in sin;
  char *host;
  char buf[MAX_LINE];
  int s;
  int len;

  if (argc == 2) {
    host = argv[1];
  } else {
    fprintf (stderr, "usage: simplex-talk host\n");
    exit(1);
  }

  hp = gethostbyname (host);
  if (!hp) {
    fprintf (stderr, "simplex-talk: unknown host: %s\n", host);
    exit (1);
  }
```

Example from book (pg 33)

```
 bzero ( (char *) &sin, sizeof (sin));
  sin.sin_family = AF_INET;
  bcopy (hp->h_addr, (char *) &sin.sin_addr, hp->h_length);
  sin.sin_port = htons (PORT);

  if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror ("simple-talk: socket");
    exit (1);
  }

  if (connect (s, (struct sockaddr *) &sin, sizeof (sin)) < 0) {
    perror ("simplex-talk: connect");
    close (s);
    exit (1);
  }

  while (fgets (buf, sizeof(buf), stdin)) {
    buf[MAX_LINE-1] = '\0';
    len = strlen (buf) + 1;
    send (s, buf, len, 0);
  }
}
```

Example from book (pg 33)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define PORT 5432
#define MAX_PENDING 5
#define MAX_LINE 256

int main () {
  FILE *fp;
  struct sockaddr_in sin, client;
  char buf[MAX_LINE];
  int s, new_s;
  int len;

  bzero ( (char *) &sin, sizeof (sin));
  sin.sin_family = AF_INET;
  sin.sin_addr.s_addr = INADDR_ANY;
  sin.sin_port = htons (PORT);

  if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror ("simple-talk: socket");
    exit (1);
  }

  if (bind (s, (struct sockaddr *) &sin, sizeof (sin)) < 0) {
}
```

Example from book (pg 33)

```
perror ("simplex-talk: bind");
   close (s);
   exit (1);
 }

 listen (s, MAX_PENDING);

 while (1) {
   if ((new_s = accept (s, (struct sockaddr *) &client, &len)) < 0) {
     perror ("simplex-talk: accept");
     exit (1);
   }
   while (len = recv (new_s, buf, sizeof (buf), 0)) {
     fputs (buf, stdout);
   }
   close (new_s);
 }
```

How to compile your program

gcc –o <output file> <list of source files> -lsocket –lnsl

e.g.

gcc –o project proj.c –lsocket -lnsl

Pitfalls

- Forgetting to convert from host to network byte order and back [htons, htonl etc]
- Forgetting to check if a function generated an error by checking return value
- Forgetting to check the number of bytes transmitted/received by send()/recv()
- Forgetting to use the addressof (&) operator
- Forgetting to include the proper header files
- Forgetting to flush output streams [fflush()]
- Forgetting to set the initial value of length before passing it to accept() or recvfrom() – problem I faced, but not shown in any of the examples

Polling streams

- Read operations are blocking calls, so we need a way to check when a stream is ready to be read from
- Accomplished by using the select() function
- Very good tutorial: http://www.ecst.csuchico.edu/~beej/guide/net/html/

Polling streams

- Include files:

```
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>
```

- FD_ZERO(fd_set *set) -- clears a file descriptor set
- FD_SET(int fd, fd_set *set) -- adds *fd* to the set
- FD_CLR(int fd, fd_set *set) -- removes *fd* from the set
- FD_ISSET(int fd, fd_set *set) -- tests to see if *fd* is in the set
- int select(int numfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout)

Java Sockets

- http://java.sun.com/docs/books/tutorial/networking/sockets/index.html
- Socket and ServerSocket classes for TCP
- DatagramSocket for UDP
- Clients use the Socket and servers use ServerSocket
- Simple example:

```
// Server lines
ServerSocket svr = new ServerSocket (2000);
Socket s = Svr.accept();

// Client lines
Socket s = new Socket (“localhost”, 2000);

// Streams
s.getInputStream();
s.getOutputStream();
```