Sockets Review (from CS252/ CS354 slides)

Programming With Sockets

Client Side

```
int cs =socket(PF_INET, SOCK_STREAM, proto)
…
Connect(cs, addr, sizeof(addr))
…
Write(cs, buf, len)
Read(cs, buf, len);
Close(cs)
```

See:

http://www.cs.purdue.edu/homes/cs354/lab5-http-server/client.cpp

Programming With Sockets

- Server Side

```
…
int masterSocket = socket(PF_INET, SOCK_STREAM, 0);
…
int err = setsockopt(masterSocket, SOL_SOCKET, SO_REUSEADDR, (char *) &optval, sizeof( int ) );
…
int error = bind( masterSocket, (struct sockaddr *)&serverIPAddress, sizeof(serverIPAddress) );
…
error = listen( masterSocket, QueueLength);
…
while ( 1 ) {
…
    int slaveSocket = accept( masterSocket,
        (struct sockaddr*)&clientIPAddress, (socklen_t*)&alen);
    read(slaveSocket, buf, len);
    write(slaveSocket, buf, len);
   close(slaveSocket);
}
```

- See: http://www.cs.purdue.edu/homes/cs354/lab5-http-server/lab5-src/daytime-server.cc

Client for Daytime Server

```
//-----------------------------------------------------------------------
// Program:   client
//
// Purpose:   allocate a socket, connect to a server, and print all output
//
// Syntax:    client host port
//
//               host  - name of a computer on which server is executing
//               port  - protocol port number server is using
//
//-----------------------------------------------------------------------

#define closesocket close
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
```

Client for Daytime Server

```
void
printUsage()
{
  printf( "Usage: client <host> <port> <name>\n");
  printf( "\n");
  printf( "          host: host where the server is running.\n");
  printf( "          port: port that the server is using.\n");
  printf( "\n");
  printf( "Examples:\n");
  printf( "\n");
  printf( "          client localhost 422422\n");
  printf( "          client lore 1345\n");
  printf( "\n");
}
```

Client for Daytime Server

```
int
main(int argc, char **argv)
{
  // Check command-line argument for protocol port and extract
  // host and port number
  if ( argc < 4 ) {
    printUsage();
    exit(1);
  }

  // Extract host name
  char * host = argv[1];

  // Extract port number
  int port = atoi(argv[2]);

  char * name = argv[3];

  // Initialize socket address structure
  struct  sockaddr_in socketAddress;

  // Clear sockaddr structure
  memset((char *)&socketAddress,0,sizeof(socketAddress));
```

Client for Daytime Server

```
// Set family to Internet
socketAddress.sin_family = AF_INET;

// Test for port legal value
if (port > 0) {
  socketAddress.sin_port = htons((u_short)port);
}
else {
  fprintf(stderr,"bad port number %s\n", argv[2]);
  exit(1);
}

// Get host table entry for this host
struct  hostent  *ptrh = gethostbyname(host);
if ( ptrh == NULL ) {
  fprintf(stderr, "invalid host: %s\n", host);
  perror("gethostbyname");
  exit(1);
}

// Copy the host ip address to socket address structure
memcpy(&socketAddress.sin_addr, ptrh->h_addr, ptrh->h_length);
```

Client for Daytime Server

```
// Get TCP transport protocol entry
  struct  protoent *ptrp = getprotobyname("tcp");
  if ( ptrp == NULL ) {
    fprintf(stderr, "cannot map \"tcp\" to protocol number");
    perror("getprotobyname");
    exit(1);
  }

  // Create a tcp socket
  int sock = socket(PF_INET, SOCK_STREAM, ptrp->p_proto);
  if (sock < 0) {
    fprintf(stderr, "socket creation failed\n");
    perror("socket");
    exit(1);
  }
```

Client for Daytime Server

```
// Connect the socket to the specified server
if (connect(sock, (struct sockaddr *)&socketAddress,
        sizeof(socketAddress)) < 0) {
  fprintf(stderr,"connect failed\n");
  perror("connect");
  exit(1);
}

// In this application we don't need to send anything.
// For your HTTP client you will need to send the request
// as specified in the handout using send().

int m = read(sock,buffer,100);
buffer[m]=0;
printf("buffer=%s\n", buffer);

write(sock, name, strlen(name));
write(sock,"\r\n",2);
```

Client for Daytime Server

```
// Receive reply
  // Data received
  char buf[1000];
  int n = recv(sock, buf, sizeof(buf), 0);
  while (n > 0) {
    // Write n characters to stdout
    write(1,buf,n);

    // Continue receiving more
    n = recv(sock, buf, sizeof(buf), 0);
  }
```

Client for Daytime Server

```
// Close the socket.
  closesocket(sock);

  // Terminate the client program gracefully.
  exit(0);
}
```

Daytime Server

```
const char * usage =
"                                                                \n"
"daytime-server:                                                 \n"
"                                                                \n"
"Simple server program that shows how to use socket calls        \n"
"in the server side.                                             \n"
"                                                                \n"
"To use it in one window type:                                   \n"
"                                                                \n"
"   daytime-server <port>                                        \n"
"                                                                \n"
"Where 1024 < port < 65536.             \n"
"                                                                \n"
"In another window type:                                        \n"
"                                                                \n"
"   telnet <host> <port>                                         \n"
"                                                                \n"
"where <host> is the name of the machine where daytime-server  \n"
"is running. <port> is the port number you used when you run   \n"
"daytime-server.                                               \n"
"                                                               \n"
"Then type your name and return. You will get a greeting and   \n"
"the time of the day.                                          \n"
"                                                               \n";
```

Daytime Server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <time.h>
```

Daytime Server

```
int QueueLength = 5;

// Processes time request
void processTimeRequest( int socket );
// Get time of day
  time_t now;
  time(&now);
  char   *timeString = ctime(&now);

  // Send name and greetings
  const char * hi = "\nHi ";
  const char * timeIs = " the time is:\n";
  write( fd, hi, strlen( hi ) );
  write( fd, name, strlen( name ) );
  write( fd, timeIs, strlen( timeIs ) );

  // Send the time of day
  write(fd, timeString, strlen(timeString));

  // Send last newline
  const char * newline="\n";
  write(fd, newline, strlen(newline));
}
```

Daytime Server

```
int
main( int argc, char ** argv )
{
  // Print usage if not enough arguments
  if ( argc < 2 ) {
    fprintf( stderr, "%s", usage );
    exit( -1 );
  }

  // Get the port from the arguments
  int port = atoi( argv[1] );

  // Set the IP address and port for this server
  struct sockaddr_in serverIPAddress;
  memset( &serverIPAddress, 0, sizeof(serverIPAddress) );
  serverIPAddress.sin_family = AF_INET;
  serverIPAddress.sin_addr.s_addr = INADDR_ANY;
  serverIPAddress.sin_port = htons((u_short) port);
```

Daytime Server

```
// Allocate a socket
 int masterSocket =  socket(PF_INET, SOCK_STREAM, 0);
 if ( masterSocket < 0) {
   perror("socket");
   exit( -1 );
 }

 // Set socket options to reuse port. Otherwise we will
 // have to wait about 2 minutes before reusing the sae port number
 int optval = 1;
 int err = setsockopt(masterSocket, SOL_SOCKET, SO_REUSEADDR,
           (char *) &optval, sizeof( int ) );

 // Bind the socket to the IP address and port
 int error = bind( masterSocket,
        (struct sockaddr *)&serverIPAddress,
        sizeof(serverIPAddress) );
 if ( error ) {
   perror("bind");
   exit( -1 );
 }
```

Daytime Server

```
// Put socket in listening mode and set the
 // size of the queue of unprocessed connections
 error = listen( masterSocket, QueueLength);
 if ( error ) {
   perror("listen");
   exit( -1 );
 }

 while ( 1 ) {

   // Accept incoming connections
   struct sockaddr_in clientIPAddress;
   int alen = sizeof( clientIPAddress );
   int slaveSocket = accept( masterSocket,
              (struct sockaddr *)&clientIPAddress,
              (socklen_t*)&alen);
```

Daytime Server

```
if ( slaveSocket < 0 ) {
      perror( "accept" );
      exit( -1 );
    }

    // Process request.
    processTimeRequest( slaveSocket );

    // Close socket
    close( slaveSocket );
  }

}
```

Daytime Server

```
void
processTimeRequest( int fd )
{
  // Buffer used to store the name received from the client
  const int MaxName = 1024;
  char name[ MaxName + 1 ];
  int nameLength = 0;
  int n;

  // Send prompt
  const char * prompt = "\nType your name:";
  write( fd, prompt, strlen( prompt ) );

  // Currently character read
  unsigned char newChar;

  // Last character read
  unsigned char lastChar = 0;
```

Daytime Server

```
//
 // The client should send <name><cr><lf>
 // Read the name of the client character by character until a
 // <CR><LF> is found.
 //

 while ( nameLength < MaxName &&
     ( n = read( fd, &newChar, sizeof(newChar) ) ) > 0 ) {

   if ( lastChar == '\015' && newChar == '\012' ) {
     // Discard previous <CR> from name
     nameLength--;
     break;
   }

   name[ nameLength ] = newChar;
   nameLength++;

   lastChar = newChar;
 }

 // Add null character at the end of the string
 name[ nameLength ] = 0;

 printf( "name=%s\n", name );
```

Daytime Server

```
  // Get time of day
   time_t now;
   time(&now);
   char   *timeString = ctime(&now);

   // Send name and greetings
   const char * hi = "\nHi ";
   const char * timeIs = " the time is:\n";
   write( fd, hi, strlen( hi ) );
   write( fd, name, strlen( name ) );
   write( fd, timeIs, strlen( timeIs ) );

   // Send the time of day
   write(fd, timeString, strlen(timeString));

   // Send last newline
   const char * newline="\n";
   write(fd, newline, strlen(newline));
}
```