TCP Server

1. TCP Server

1.1 CSCI 330

1.2 Unit Overview

Unit Overview

- TCP client & server programming
  - review concepts and necessary system calls
  - illustrate client with DNS lookup
- Server fork to process client request
- Example TCP server
  - list files in a directory

1.3 TCP programming

TCP programming

- provides multiple endpoints on a single node: port
- common abstraction: socket
- socket is end-point of communication link
  - identified as IP address + port number
  - can receive data
  - can send data

1.4 Socket system calls

Socket system calls

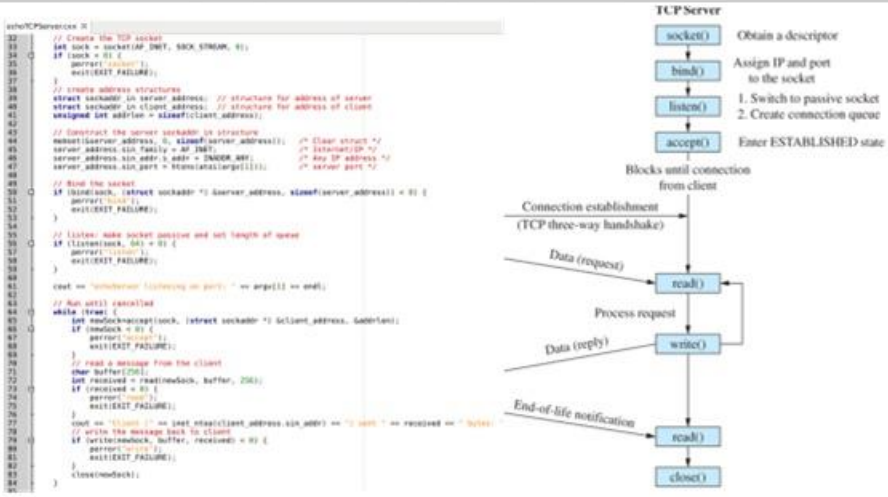
server

client

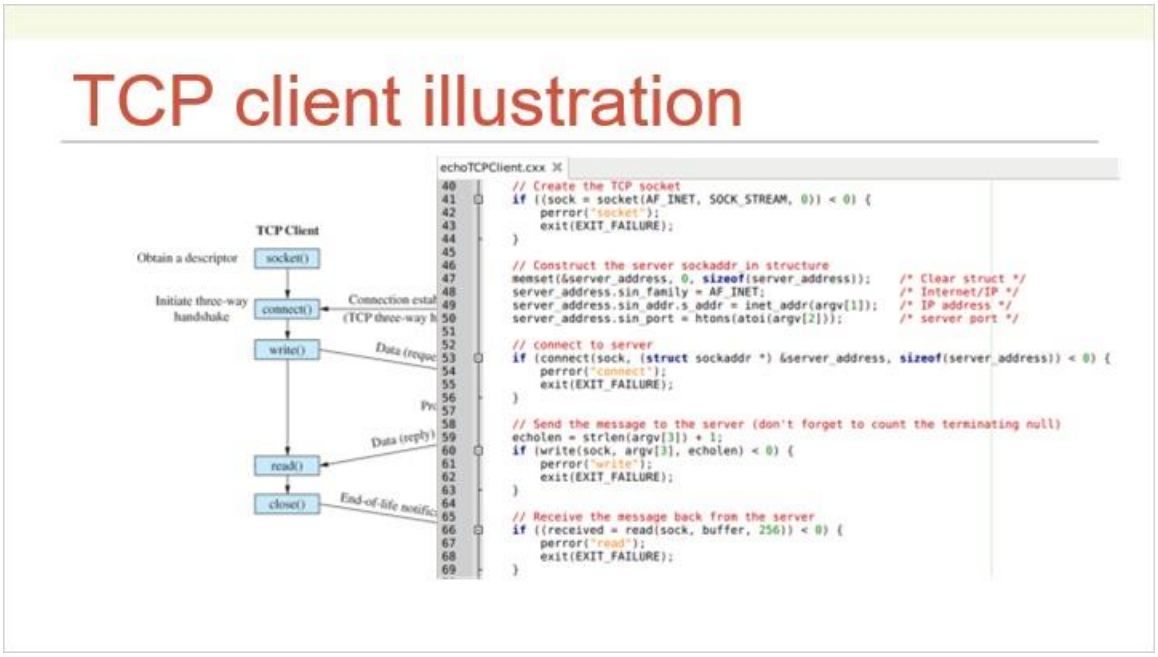
Primitive	Meaning
socket	Create a new communication endpoint
bind	Attach a local address to a socket
listen	Announce willingness to accept connections
accept	Block caller until a connection request arrives
connect	Actively attempt to establish a connection
write	Send(write) some data over the connection
read	Receive(read) some data over the connection
close	Release the connection

1.5 TCP server illustration

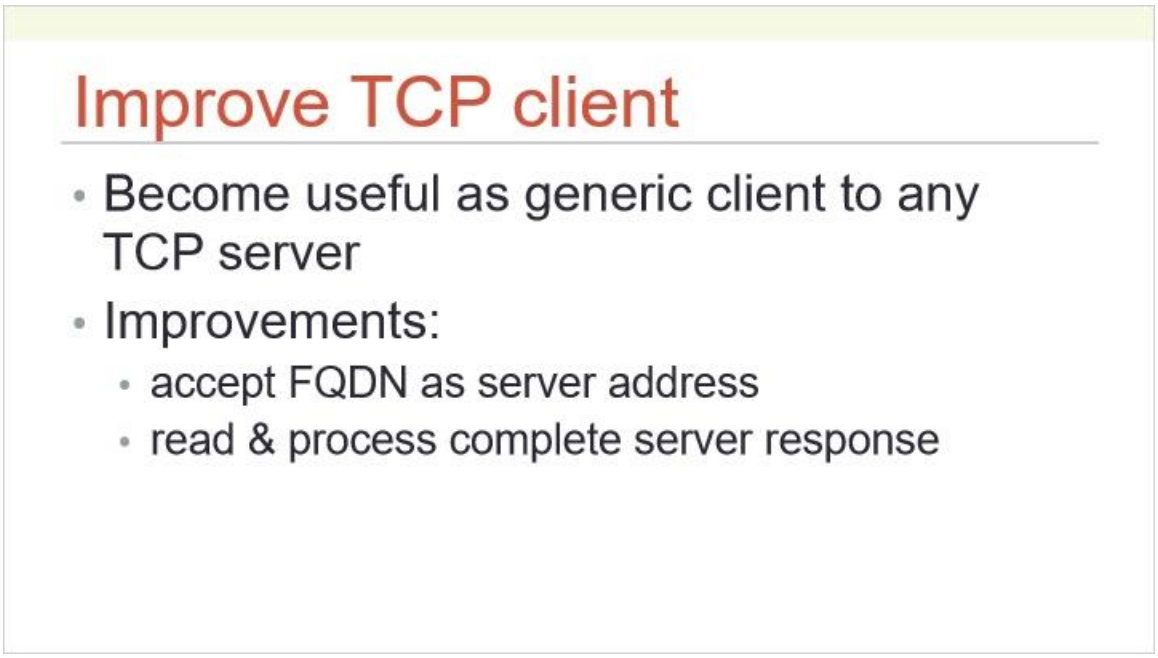
TCP server illustration



1.6 TCP client illustration



1.7 Improve TCP client



1.8 Accept FQDN as server address

Accept FQDN as server address

```
// lookup FQDN
struct addrinfo *res, hints;
memset(&hints, 0, sizeof(hints));
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
int error = getaddrinfo(argv[1], argv[2], &hints, &res);
if (error) {   …   }

// Create the TCP socket
int sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock < 0) {  …  }

// connect to server
if (connect(sock, res->ai_addr, res->ai_addrlen) < 0) {  …  }
```

1.9 Process complete server response

Process complete server response

```
// Receive the message back from the server
do {
   received = read(sock, buf, sizeof(buf));
   if (received < 0) {  …  }
   cout.write(buf, received);
} while (received > 0);
```

1.10 Generic TCP request client (1 of 2)

TCPClient.cxx - /home/student/Desktop/Week 14 - TCP Server - Geany

File Edit Search View Document Project Build Tools Help

TCPClient.cxx

```
/*
 * TCPClient.cxx
 *
 * TCP client
 *
 *  sends message to TCP server
 *  waits for message received from server
 *
 *  command line arguments:
 *      argv[1] FQDN of server
 *      argv[2] port number to send to
 *      argv[3] request to send
 *
 */
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
#include <cstdlib>
#include <cstring>
#include <cstdio>
#include <iostream>
using namespace std;
```

This is Geany 1.32.

1.11 Review: TCP server

Review: TCP server

```
while (true) {
    int connSock=accept(sock, (struct sockaddr *) &client_address, &addrlen);
    if (connSock < 0) {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    // read a message from the client
    char buffer[1024];
    int received = read(connSock, buffer, sizeof(buffer));
    if (received < 0) {
        perror("read");
        exit(EXIT_FAILURE);
    }
    cout << "Client (" << inet_ntoa(client_address.sin_addr) << ") sent " << received << " bytes: " << buffer << endl;
    // write the message back to client
    if (write(connSock, buffer, received) < 0) {
        perror("write");
        exit(EXIT_FAILURE);
    }
    close(connSock);
}
```

1.12 Review: TCP Server basic logic

Review: TCP Server basic logic

```
while (true) {
    connSock = accept(sock, ...)

    // process client's request
    //    via connSock

    close(connSock);
}
```

Server is busy !
No other client can connect

1.13 Better: TCP Server fork

Better: TCP Server fork

- server starts loop
  - blocks on accept for connection from client
  - after accept:
    - accept returns dedicated connection socket
    - server forks into parent and child process
- parent process
  - closes dedicated connection socket
  - continues to block for next accept
- child process
  - serves client request
  - communicates with client via dedicated connection socket

1.14 TCP Server fork: logic

TCP Server fork: logic

```
while (true) {
    connSock = accept(sock, ...);
    if (fork()) {      // parent process
       close(connSock);
    } else {           // child process
       // process client's request via connSock
       ...
     }
}
```

1.15 TCP server/fork illustration (1 of 2)

TCP server/fork illustration (1 of 2)

TCPServerFork.cxx ×

```
    // Run until cancelled
    while (true) {
        int connSock=accept(sock, (struct sockaddr *) &client_address, &addrlen);
        if (connSock < 0) {
            perror("accept");
            exit(EXIT_FAILURE);
        }
        // fork
        if (fork()) {       // parent process
            close(connSock);
        } else {            // child process
            processClientRequest(connSock);
        }
    }
    close(sock);
    return 0;
}
```

1.16 TCP server/fork illustration (2 of 2)

TCP server/fork illustration (2 of 2)

TCPServerFork.cxx

```
25  void processClientRequest( int connSock) {
26      int received;
27      char buffer[1024];
28
29      // read a message from the client
30      if ((received = read(connSock, buffer, sizeof(buffer))) <= 0) {
31          perror("read");
32          exit(EXIT_FAILURE);
33      }
34
35      cout << "Client sent " << received << " bytes: " << buffer << endl;
36
37      // write the message back to client
38      if (write(connSock, buffer, received) < 0) {
39          perror("write");
40          exit(EXIT_FAILURE);
41      }
42      close(connSock);
43      exit(EXIT_SUCCESS);
44  }
```

1.17 TCP server/fork illustration

TCPServerFork.cxx - /home/student/Desktop/Week 14 - TCP Server - Geany

File Edit Search View Document Project Build Tools Help

TCPClient.cxx TCPServerFork.cxx

```
1   /*
2    * TCPServerFork.cxx
3    *
4    * TCP echo server
5    *
6    *  loops/waits/forks for message received from client
7    *        send message back to client
8    *
9    *  command line arguments:
10   *      argv[1] port number to receive from
11   *
12   */
13  #include <sys/types.h>
14  #include <sys/socket.h>
15  #include <arpa/inet.h>
16  #include <unistd.h>
17  #include <netinet/in.h>
18
19  #include <cstdio>
20  #include <cstdlib>
21  #include <cstring>
22  #include <iostream>
23  using namespace std;
```

Compilation finished successfully.

1.18 Server example: list directory

Server example: list directory

- after accept, server forks to service client request
  - parent process will loop to next accept
- child process serves client request
  - read directory path name from client
  - open directory
  - read directory entries, send file names to client
  - end process

1.19 Server child: processClientRequest

Server child: processClientRequest

TCPServerReadDir.cxx

```
25  void processClientRequest(int connSock) {
26      int received;
27      char path[1024], buffer[1024];
28
29      // read a message from the client
30      if ((received = read(connSock, path, sizeof(path))) < 0) {
34      cout << "Client request: " << path << endl;
35
36      // open directory
37      DIR *dirp = opendir(path);
38      if (dirp == 0) {
48
49      // read directory entries
50      struct dirent *dirEntry;
51      while ((dirEntry = readdir(dirp)) != NULL) {
52          strcpy(buffer, dirEntry->d_name);
53          strcat(buffer, "\n");
54          if (write(connSock, buffer, strlen(buffer)) < 0) {
58          cout << "sent: " << buffer;
59      }
60      closedir(dirp);
61      cout << "done with client request\n";
62      close(connSock);
63      exit(EXIT_SUCCESS);
64  }
```

1.20 TCP server: opendir detail

TCP server: opendir detail

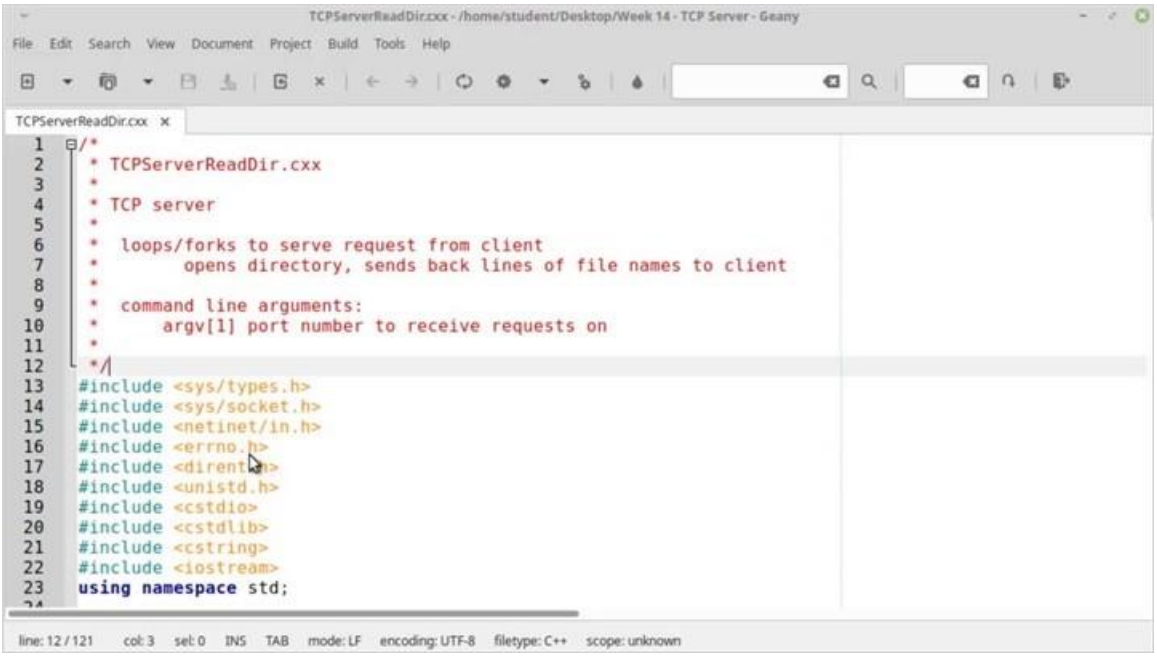
```
// open directory
DIR *dirp = opendir(path);
if (dirp == 0) {
    // tell client that an error occurred
    strcpy(buffer, path);
    strcat(buffer, ": could not open directory\n");
    if (write(connSock,buffer,strlen(buffer)) < 0) {
            perror("write");
            exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

1.21 TCP server: readdir detail

TCP server: readdir detail

```
struct dirent *dirEntry;
while ((dirEntry = readdir(dirp)) != NULL) {
    strcpy(buffer, dirEntry->d_name);
    strcat(buffer, "\n");
    if (write(connSock,buffer,strlen(buffer)) < 0) {
        perror("write");
        exit(EXIT_FAILURE);
    }
    cout << "sent: " << buffer;
}
```

1.22 TCPServerReadDir Example



1.23 TCP server: error detail via dup

1.24 Summary

Summary

- TCP server programming
  - TCPClient.cxx
- server fork to process client request