EECS 22L: Software Engineering Project in C Language

Lecture 9

Rainer Dömer

doemer@uci.edu

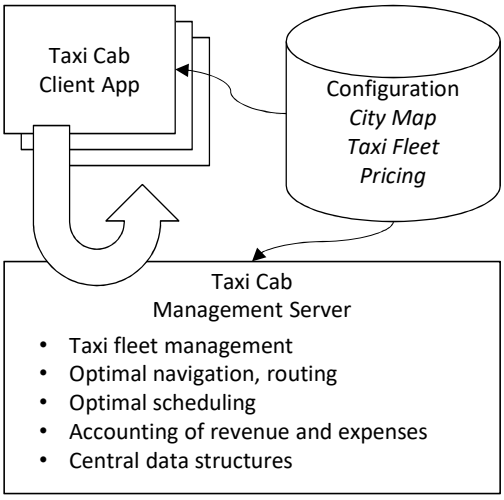
The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 9: Overview

- Project 2 Technical Discussion and Advise
  - Software architecture and components
- Discussion on Socket Communication
  - Client-server example
  - Blocking I/O communication
  - Multiplexing multiple connections
  - Clock server example

Project 2: Software Architecture

- Overall System Specification (designed by consultant)

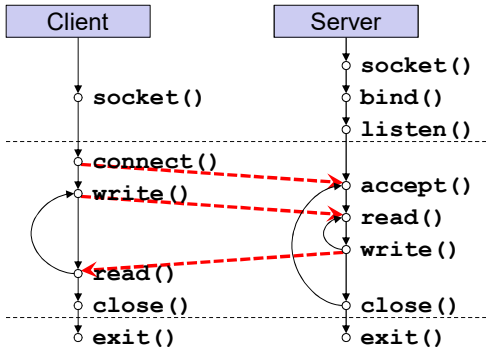


Project 2: Client-Server Communication

- Discussion on Socket Communication
  - Reference: Sockets Tutorial
    - `http://www.linuxhowtos.org/C_C++/socket.htm`
    - `http://www.linuxhowtos.org/data/6/client.c`
    - `http://www.linuxhowtos.org/data/6/server.c`
  - Reference: Linux manual pages
    - `man socket`
    - `man select`
    - `man select_tut`
  - Extended client-server example: `~eecs22/SocketTutorial.tar.gz`
    - `client2.c`
    - `server2.c`
    - ➢ This example can handle only one client at a time, others have to wait for their turn to connect

Project 2: Client-Server Communication

- Discussion on Socket Communication
  - Sequence Diagram for client-server example



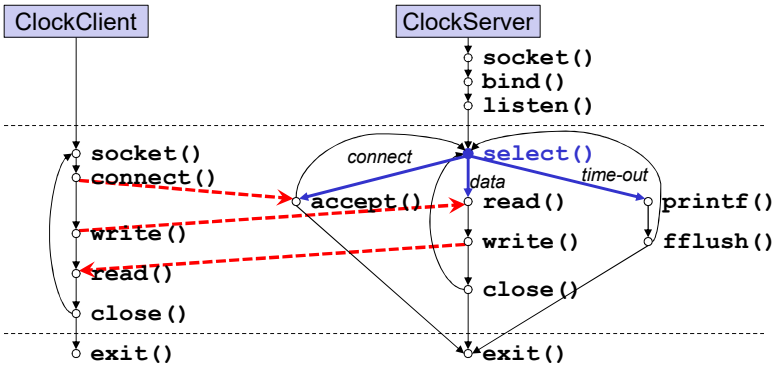
  - ➢ This simple example can handle only one client at a time, others have to wait for their turn to connect (they are blocked)
  - ➢ *Blocking communication* can stall both the client and the server!

Project 2: Client-Server Communication

- Discussion on Socket Communication
  - Handling multiple active client connections
    - Option 1: Parallel/concurrent (asynchronous) I/O
      - Use multiple processes (**`fork()`**) or threads (**`pthread_create()`**)
      - Requires Operating System (OS) knowledge (i.e. EECS 111), and very careful programming to avoid race-conditions and deadlocks
    - Option 2: Synchronous I/O multiplexing
      - Wait on multiple I/O requests, handle them first-come-first-served (FCFS)
      - Function **`select()`** monitors multiple file descriptors, with timeout option
  - ➢ Multiplexing multiple connections with **`select()`**
  - Clock server example: **`~eecs22/ClockServer.tar.gz`**
    - **`ClockServer.c`**
    - **`ClockClient.c`**
    - **`Makefile, README`**
  - ➢ Online demonstration!

Project 2: Client-Server Communication

- Multiplexing multiple client connections with **`select()`**
  - ClockServer example: **`~eecs22/ClockServer.tar.gz`**



  - ➢ Wait simultaneously to connect, to transfer data, or for time-out!
  - ➢ Keep sequential execution short
  - ➢ Limit client-server interaction to one request at a time

Project 2: Client-Server Communication

- Communication Example: *"Call a Taxi from UNI to SCA"*
  - ➢ Reconsider: Handle one request at a time, keep sequences short!
  - Client: **`Hello Server!`**
  - Server: **`ERROR invalid message "Hello Server!"`**
  - Client: **`REQUEST_POSITION Taxi7`**
  - Server: **`OK Taxi7 POSITION J2 ETA D4 11:45`**
  - Client: **`REQUEST_TAXI S8 TO D4 ASAP`**
  - Server: **`OK Taxi7 PICKUP S8 10:15 DROPOFF D4 11:42 $8.75 CONFIRM`**
  - Client: **`OK CONFIRMED`**
  - Server: **`OK Taxi7 POSITION D8 ETA S8 10:15`**

Possibly a long delay for client to respond!

Project 2: Client-Server Communication

- Communication Example: *"Call a Taxi from UNI to SCA"*
  - ➢ Reconsider: Handle one request at a time, keep sequences short!
  - – Client: `Hello Server!`
  - – Server: `ERROR invalid message "Hello Server!"`
  - – Client: `REQUEST_POSITION Taxi7`
  - – Server: `OK Taxi7 POSITION J2 ETA D4 11:45`
  - – Client: `REQUEST_TAXI S8 TO D4 ASAP`
  - – Server: `OK Taxi7 PICKUP S8 10:15 DROPOFF D4 11:42 $8.75`
    `CONFIRM #10042`
  - – Client: `CONFIRM #10042`
  - – Server: `OK Taxi7 POSITION D8 ETA S8 10:15`
  - ➢ *Unconfirmed requests should expire after some period (so that there is no problem when the client doesn't cancel the request)*

Decoupled requests by confirmation number

EECS22L: Software Engineering Project in C, Lecture 9 (c) 2017 R. Doemer 9

Project 2: Client-Server Communication

- Protocol Specification Example (Backus-Naur Form, BNF)
  - – Client Request

```
<request> ::= REQUEST_TAXI <location> TO <location> <special>*
            |   REQUEST_POSITION <taxi>
            |   CONFIRM <reservation>
            |   CANCEL <reservation>
<location>::= <pos>
            |   CORNER <street_name> AND <street_name>
            |   <landmark_name>
<special> ::= [ASAP]
            |   [AT <time>]
            |   [FOR <number_persons>]
            |   [NON_STOP]
<time>    ::= <hours>:<minutes>
```

  - – Server Response

```
<response>::= OK <taxi> PICKUP <pos> [<time>] DROPOFF <pos> [<time>]
                  $<amount> CONFIRM <reservation>
            |   DECLINED <reason>
            |   OK <taxi> POSITION <position> [ETA <position> <time>]
            |   INVALID <reservation>
            |   ERROR <message>
<amount>  ::= <dollars>.<cents>
```

EECS22L: Software Engineering Project in C, Lecture 9 (c) 2017 R. Doemer 10