Discussion Session 03

Project 1 | Framing | Parsing
Overview

1. Framing and Parsing
2. Generating HTTP Requests
3. Modular Server Code
Calculator Example - Server

Calculator Server -

1. Create a socket - `socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)`
2. Bind the socket - `bind(servSock, (struct sockaddr *) &servAddr, sizeof(servAddr))`
3. Start Listening - `listen(servSock, MAX_PENDING)`
4. Loop and Accept Connections - `accept(servSock, (struct sockaddr *) &echoClntAddr, &clntLen)`

Notes:

1. `clntLen` should be an unsigned int, or you can cast it to `(socklen_t *)`
2. `bind()` is also a function in std namespace. Use `::bind()` for the sockets bind function
Calculator Example - Client

Request Format -

ADD 10
SUB 5
SET 20
ADD 30

The server applies these operations on an accumulator which is set to 0 initially
Framing

Framing -

Extracting a complete message based on delimiters

Request1
Request2
Request3
Request4

Separate out into Request1, Request2, Request3 and Request4
Generating HTTP Request

A sample HTTP request on TritonHTTP Spec -

```
GET / HTTP/1.1
Host: www.cs.ucsd.edu
User-Agent: MyTester v1.0
Cookie: 123

Here - Carriage Return (\r), \n - Line Feed (\n)
```
Creating HTTP Request

Script - unix2dos.sh

```bash
#!/bin/bash

if [ "$#" -ne 2 ]; then
    echo "Usage: $0 input-file.txt output-file.txt"
    exit 1
fi

if [ ! -f $1 ]; then
    echo "File $1 not found"
    exit 2
fi

sed 's/$"\r"/echo \"\r\"/ < $1 > $2

sh unix2dos.sh input_file.txt output_file.txt
```
Creating HTTP Request - Example

1. Create a request in a text editor and save the file
2. Use the unix2dos script to ensure lines end with CRLF
3. Use the newly created file in your client to send requests

Request from Text Editor

GET / HTTP/1.1\nHost: www.cs.ucsd.ed
User-Agent: MyTester v1.0
Cookie: 123 (press Enter)

GET /image.png HTTP/1.1\nHost: www.cs.ucsd.ed
User-Agent: MyTester v1.0
Cookie: 345323 (press Enter)

unix2dos.sh

Valid request with CRLF

GET / HTTP/1.1\nHost: www.cs.ucsd.ed
User-Agent: MyTester v1.0
Cookie: 123

GET /image.png HTTP/1.1\nHost: www.cs.ucsd.ed
User-Agent: MyTester v1.0
Cookie: 345323
Parsing

Converting the request string into a Request Object

1. Extract each line
2. Parse initial line and headers

Request {
    initialLine: "GET /image.png HTTP/1.1"
    headers: {
        "Host": "Server",
        "User-Agent": "MyTester v1.0",
        "Cookie": "123"
    }
}
Modular Server Code - Guidelines

These are a few guidelines about how to write a modular server. Your code structure can be different

1. `HttpdServer::launch` method starts the server and accepts incoming connections
2. A separate class for handling client connection
3. Separate out Framing and Parsing into methods (or even separate files)
4. Separate method for sending out response to the client
Questions?