সি দিয়ে ওয়েব সার্ভার

এসো নিজে করি : ওয়েব সার্ভার

লেখক : ওমর শেহাব

এটি একটি শিশুতোষ লেখা। যারা মাত্র সি প্রোগ্রামিং শেখা শুরু করেছে তাদের জন্য এটি লিখছি যাতে তারা সি ল্যাংগুয়েজের সামর্থ্য সম্পর্কে কিছুটা ধারণা পায়।

আজকে থেকে এগার বছর আগের কথা। আমি তখন শাবিপ্রবি কম্পিউটার বিজ্ঞান ও প্রকৌশল বিভাগে তৃতীয় বর্ষে পড়ছি। আমাদের একটা কোর্স ছিল সিএসই ৩৫০। এই কোর্সে একটি সফটওয়্যার বানাতে হত। তখন আমার শিক্ষক শাহরিয়ার ভাইয়ের কাছ থেকে সি ল্যাংগুয়েজ দিয়ে একটি ওয়েব সার্ভার বানানোর আইডিয়া পেলাম।

ওয়েব সার্ভার বানানো নতুন কোন ব্যাপার না। ইতিমধ্যেই বাজারে তখন ফ্রি ওপেনসোর্স অ্যাপাচি সার্ভার পাওয়া যেত। শাহরিয়ার ভাই একই বছর অপারেটিং সিস্টেম ক্লাশে মালটিথ্রেড প্রোগ্রামিং, থ্রেড স্কেজ্যুলিং এসব পড়িয়েছিলেন। তার মতে এই প্রজেক্ট করলে আমি যা থিওরী পড়েছি সেগুলো হাতে কলমে নাড়াচাড়া করতে পারব। আমার মনে হল বুদ্ধিটা খারাপ না।

সেই কোডটি এই লিংকে (http://bit.ly/1GEzoX7) পাওয়া যাবে। আমি ভাবলাম যারা মাত্র সি শেখা শুরু করেছে তাদের কেউ কেউ এই ব্যাপারে আগ্রহী হতে পার। আমি একটু একটু করে এই কোডের ব্যাপারগুলো ধরিয়ে দিব। একটি জিনিস মনে রাখতে হবে আমি তখনও ছাত্র ছিলাম এ কারণে কোডের মান খুব বেশি আহামরি না।

তাহলে শুরু হোক! আমি একটি একটি করে ব্লক দিব আর তার নিচে ওই ব্লক নিয়ে কথা বলব।

```
/*
 * Course Code 350
 *
 * A Multithreaded Tiny HTTP Server
 * Demonstrating Thread Pool Management
 * Following the Thread Pool Management
 * Outline from Unix Network Programming Vol 1
 * by W. Richard Stevens.
 *
 * Project accomplished by: Abu Mohammad Omar Shehab Uddin Ayub
 * Reg No. 2000 330 096
 * Section B
 * 3rd Year 2nd Semester
 * Dept of CSE, SUST
 *
 * Idea and guided by: Mahmud Shahriar Hussain
 * Lecturer
 * Dept of CSE, SUST
 *
 * Course Instructor: Rukhsana Tarannum Tazin
 * Lecturer
 * Dept of CSE, SUST
 */
```

যারা ইতিমধ্যে প্রোগ্রামিং শুরু করেছ তারা নিশ্চয়ই জান যেকোন কোডের আগে কমেন্টে এই কোডের উদ্দেশ্য ও প্রোগ্রামারের নাম দেয়া স্বাভাবিক ভদ্রতা।

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

সি ল্যাংগুয়েজের নিজস্ব যে ফিচার বা সাপোর্ট আছে সেগুলো বিভিন্ন হেডার ফাইলে কোড করা থাকে। যেমন আমি পোসিক্স থ্রেড নিয়ে কাজ করতে চাই কাজেই পিথ্রেড.এইচ ফাইলটি ব্যবহার করছি।

```
#define MAX_CLIENT 5
#define MAX_THREAD 3
#define MAX_REQ_HEADER 10
#define MAX 10
```

```
#define MAX_LENGTH 6
#define MIME_LENGTH 20
```

এখানে আমি আগে থেকে কিছু জিনিস ঠিক করে নিচ্ছি। যেমন: আমি চাই এক সাথে পাঁচটির বেশি ব্রাউজার (ফায়ারফক্স, ক্রোম এগুলোকে ব্রাউজার বলে) আমার সার্ভারের পেজ ব্রাউজ করতে পারবে না কাজেই আমি MAX_CLIENT 5 দিয়েছি।

এর পর আমি আমার পছন্দ মত কিছু ডেটা স্ট্রাকচার ঠিক করে নিব।

```
typedef struct {
   pthread_t threadID;
   long threadCount;
}Thread;
```

যেমন থ্রেড সম্পর্কিত তথ্য রাখার জন্য আমি থ্রেড নামে একটি স্ট্রাকচার ব্যবহার করছি।

```
typedef struct {
   char hostName[100];
   char filePath[100];
}Request;
```

কোন ব্রাউজার থেকে যখন কেউ একটি ওয়েবপেইজ দেখার অনুরোধ সার্ভারের কাছে পাঠাবে সেই অনুরোধটিকে রিকোয়েস্ট নামে একটি স্ট্রাকচারে রাখব। এখানে কোন সার্ভার থেকে অনুরোধটি এসেছে এবং কোন ওয়েব পেজটি দেখতে চাচ্ছে এই তথ্যগুলো থাকবে।

```
typedef struct{
   int code;
   char date[30];
   char server[30];
   char last_modified[30];
   long content_length;
   char content_type[10];
   char file_path[80];
}Reply;
```

যখনই কোন ব্রাউজার ব্যবহার করে কেউ একটি পেজ দেখতে চাইবে তার জন্য ঠিকঠাক মতো জবাব তৈরি করা হবে। পেজটি থাকলে সেটি পাঠানো হবে আর না থাকলে বলা হবে পেজ নেই। এর জন্য বেশ কিছু তথ্য পাঠাতে হয়। এগুলো রিপ্লাই নামে একটি স্ট্রাকচারের মাধ্যমে গুছিয়ে পাঠানো হয়।

```
Thread thrdPool[MAX_THREAD];
Request request[MAX_THREAD];
Reply reply[MAX_THREAD];
int clntConnection[MAX_CLIENT], clntGet, clntPut;
```

এখানে আমি আমার সার্ভারের লোড নেবার সামর্থ্য ঠিক করে দিচ্ছি। লোড মানে সে একসাথে কয়জন ওয়েব ব্যবহারকারীর রিকোয়েস্ট সামলাতে পারবে।

```
thread_mutex_t clntMutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t srvrMutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t srvrMutex2 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t emitMutex2 = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t clntCondition = PTHREAD_COND_INITIALIZER;
```

মিউটেক্স জিনিসটা হল মালটিথ্রেডেড প্রোগ্রামিংয়ে অনেকটা ছিটকিনির মত কাজ করে। একটি ওয়েবসার্ভারের কাছে যখন কোন পেজ দেখার রিকোয়েস্ট আসে তখন সে সেটি নিয়ে সারাক্ষণ ব্যস্ত থাকে না। সে কি করে একটি থ্রেড তৈরি করে সেই থ্রেডের কাছে সেই রিকোয়েস্টের দায়িত্ব দিয়ে নিজে নতুন রিকোয়েস্টের জন্য অপেক্ষা করতে থাকে। নতুন রিকোয়েস্ট আসলে সেটিকে নতুন আরেকটি থ্রেডের কাছে পাঠিয়ে দেয় আর আবারো সে আরেকটি রিকোয়েস্টের জন্য অপেক্ষা করতে থাকে। থ্রেডগুলো অপারেটিং সিস্টেমের সাহায্য নিয়ে নিজের মত করে রিকোয়েস্ট পরে পরে যে ওয়েবপেজটি দেখতে চাওয়া হয়েছে সেটি পাঠিয়ে দেয়। একটি থ্রেড কখনও জানতে পারে না অন্য থ্রেডের ভিতরে কি চলছে। তাদের সবার নিজস্ব জগৎ থাকে। তবে সবগুলো থ্রেডকে লাইনে রাখার জন্য কিছু কোড আছে, কিছু ভ্যারিয়েবল আছে। সেই ভ্যারিয়েবলগুলো সব থ্রেড দেখতে পারে। এখন একই ভ্যারিয়েবল নিয়ে যদি একাধিক থ্রেড একই সময়ে কাজ করতে চায় তাহলে তো ভজঘট লেগে যাবে। এখানেই মিউটেক্স কাজে আসে। কোন থ্রেড যখন কোন শেয়ারড ভ্যারিয়েবল নিয়ে কাজ করে মিউটেক্স বা ছিটকিনি তুলে দেয়। অন্য থ্রেড যখন কাজ করতে গিয়ে দেখে ছিটকিনি তোলা তখন তারা চুপচাপ অপেক্ষা করে। শেয়ারড ভ্যারিয়েবল নিয়ে কাজ শেষ হলে সেই থ্রেড ছিটকিনি বা মিউটেক্স নামিয়ে দেয়। তখন অপেক্ষা করতে থাকা অন্য থ্রেড কাজ শুরু করে আর আবারও ছিটকিনি তুলে দেয়।

```
static int numThreads;
```

```
void *request_handler(void *arg);
int find_method (char *req);
Request parse_request(char *rbuff);
Reply prepare_reply(int code, Request rq);
char *getMimeType(char *str);
void emit_reply(int conn, Reply rply);
```

একটু গুছিয়ে কাজ করার জন্য পুরো কোডটিকে আমি কিছু ফাংশনে ভাগ করেছি। এখানে সেগুলো ডিক্লেয়ার করলাম।

```
int main(void)
{
```

মেইন ফাংশন শুরু হল অর্থাৎ প্রোগ্রামের মূল অংশটি শুরু হচ্ছে।

```
int i, serverSockfd, clientSockfd;
int serverLen, clientLen;
struct sockaddr_in serverAddress, clientAddress, tempAddress;
```

কাজের সুবিধার জন্য কিছু ভ্যারিয়েবল দরকার।

```
//defining number of threads
 numThreads = MAX_THREAD;
```

```
// initializing queue parameters
 clntGet = clntPut = 0;
```

একবারে একসাথে কয়টি রিকোয়েস্ট নিয়ে কাজ করা যাবে অর্থাৎ সার্ভারের সামর্থ ঠিক করে দিচ্ছি।

```
//creating all the threads
for(i = 0; i < numThreads; i++)
{
    pthread_create(&thrdPool[i].threadID, NULL, &request_handler, (void *) i);
} // end of for loop
```

যখনই কোন রিকোয়েস্ট আসবে একটি করে থ্রেড তার দায়িত্ব নিবে। রিকোয়েস্ট আসার পরপর নতুন থ্রেড তৈরি করতে গেলে প্রোগ্রাম স্লো হয়ে যায় তাই আগেই কিছু তৈরি করে রাখি।

```
//creating the socket descriptor
serverSockfd = socket(AF_INET, SOCK_STREAM,0);
```

```
if(serverSockfd < 0)
{
   error("\nError opening socket");
   exit(1);
}
```

```
serverAddress.sin_family = AF_INET;
serverAddress.sin_addr.s_addr = INADDR_ANY;
serverAddress.sin_port = htons(80);
serverLen = sizeof(serverAddress);
```

```
if(bind(serverSockfd, (struct sockaddr *)&serverAddress, serverLen) < 0)
{
   printf("\nError in binding.");
   exit(1);
}
```

অন্য যেকোন ওয়েবসার্ভারের মত আমার সার্ভারও কম্পিউটারের ৮০ নম্বর পোর্টে কান পেতে রাখবে। এখন আমার জানা জরুরী যে ইতিমধ্যে এই পোর্ট অন্য কোন সফটওয়্যার দখল করে ফেলেছে কিনা। সাবধানের মার নাই!

```
listen(serverSockfd, 10);
```

আমি কাআআআআন পেতেএএএ রইইই!

```
for( ; ; )
{
   fflush(stdout);
   //printf("\nWaiting for clients...");
   clientLen = sizeof(tempAddress);
   //printf("\n\nBlocked on accept()");
   clientSockfd = accept(serverSockfd, (struct sockaddr *)&clientAddress,
&clientLen);
   //printf("\nAllocated client descriptor# %d", clientSockfd);
   pthread_mutex_lock(&srvrMutex);
   clntConnection[clntPut] = clientSockfd;
   if(++clntPut == MAX_CLIENT)
   clntPut = 0;

   if(clntPut == clntGet)
   {
      printf("\nCan't handle any more request...\nTerminating...");
      exit(1);
   }

   pthread_cond_signal(&clntCondition);
   pthread_mutex_unlock(&srvrMutex);
   fflush(stdout);

} // end of infinite for loop
```

ইনফিনিটি লুপ খুব একটা ভাল জিনিস না। এটি আরো ভাল ভাবে করতে পারা উচিৎ। এখানে আমি যখনই কোন রিকোয়েস্ট পাচ্ছি ছিটকিনি তুলে দিয়ে একটি থ্রেড ধরে এনে তাকে রিকোয়েস্টের দায়িত্ব দিয়ে দিচ্ছি। যদি দেখি যে ইতিমধ্যে অনেক রিকোয়েস্ট জমে গেছে ভদ্রভাবে না করে দিচ্ছি। তারপর আবার ছিটকিনি বা মিউটেক্স নামিয়ে দিচ্ছি।

```
} // end of main
```

মেইন ফাংশন শেষ!

কোন থ্রেড যখন একটি রিকোয়েস্ট পায় তখন রিকোয়েস্ট হ্যান্ডলার নামে একটি ফাংশনের মাধ্যমে সেটি সামলায়। এখন আমরা সেই ফাংশনটি দেখব।

```
void *request_handler(void *arg)
{
```

রিকোয়েস্ট হ্যান্ডলার শুরু হল।

```
int connfd;
char *reqBuff;
fflush(stdout);
//printf("\nThread %d starting", (int) arg);
```

এক আধটু ভ্যারিয়েবল সব জায়গাতেই লাগে!

```
for( ; ; )
{
   pthread_mutex_lock(&clntMutex);
   //printf("\nMutex locked by thread# %d.", (int) arg);
   while(clntGet == clntPut)
   pthread_cond_wait(&clntCondition, &clntMutex);
```

আবারও ইনফিনিটি লুপ ব্যবহার করছি, দু:খিত! আমি এখন শেয়ারড ভ্যারিয়েবলগুলো ব্যবহার করছি কাজেই মিউটেক্স বা ছিটকিনি লাগছে।

```
connfd = clntConnection[clntGet];
if(++clntGet == MAX_CLIENT)
clntGet = 0;
```

সার্ভার ওভারলোড হয়ে যাচ্ছে কিনা সেটাও খেয়াল রাখতে হচ্ছে।

```
// thread operation
// printf("\nIn between mutex lock-unlock. Thread# %d, Connection# %d", (int)
arg, connfd);
reqBuff = (char *) malloc (1000);
read(connfd, reqBuff, 1000);
```

এক একটি রিকোয়েস্টের জন্য কি পরিমাণ রেম (RAM) খরচ হবে সেগুলো ঠিক করছি।

```
int temp;
temp = find_method(reqBuff);
```

কোন ব্রাউজার যখন রিকোয়েস্ট পাঠায় তার ভিতরে মেথড বলে একটি জিনিস বলা থাকে। এটি গেট মেথড হতে পারে, পুট মেথড হতে পারে অথবা বাজে ব্রাউজার হলে অর্থহীন কিছু পাঠাতে পারে। এই মেথডটি কি সেটি আমি বের করছি। আমার সার্ভার শুধু গেট মেথড সামলাতে পারে, অন্য কোন মেথড হলে ভদ্রভাবে না করে দেয়।

```
if(temp == 0)
{
   //printf("\nfind_method returned 0");
   request[(int)arg] = parse_request(reqBuff);
   //printf("\nHost: %s\nFile Path: %s", request[(int)arg].hostName,
request[(int)arg].filePath);
```

```
   reply[(int)arg] = prepare_reply(200, request[(int)arg]);
   printf("\n\nprinting the reply structure");
   //printf("\nCode: %d\nDate: %s\nServer: %s\nlast_modified:
%s\ncontent_length: %ld\ncontent type: %s\nfile path: %s",
reply[(int)arg].code, reply[(int)arg].date, reply[(int)arg].server,
reply[(int)arg].last_modified, reply[(int)arg].content_length,
reply[(int)arg].content_type, reply[(int)arg].file_path);
   emit_reply(connfd, reply[(int)arg]);
}
```

এটি গেট মেথড তাই কাজে নেমে পড়ছি।

```
else if(temp == 1)
{
   printf("\nfind_method returned 1");
   reply[(int)arg] = prepare_reply(501, request[(int)arg]);
   emit_reply(connfd, reply[(int)arg]);
}
```

এটি পুট বা অন্য কোন স্বীকৃত মেথড যার সাপোর্ট আমি দিইনি। দু:খিত!

```
else
{
   printf("\nfind_method returned -1");
   reply[(int)arg] = prepare_reply(400, request[(int)arg]);
   emit_reply(connfd, reply[(int)arg]);
}
```

আজগুবি মেথড।

```
pthread_mutex_unlock(&clntMutex);
//printf("\nMutex unlocked by thread# %d.", (int) arg);
thrdPool[(int) arg].threadCount++;
```

ছিটকিনি!

```
   // shahriar bhai told that closing the connection formally is not so
important
   // good performance achieved @ closing it formally
   close(connfd);
}
```

যখনই কোন রিসোর্স ব্যবহার করব না তখনই সেটি মেমোরি থেকে সরিয়ে নিব।

```
} // end of request_handler function
```

ফাংশন শেষ!

ব্রাউজারকে ওয়েবসার্ভার যখন উত্তর পাঠায় সেই উত্তরগুলোতে কি কি তথ্য থাকবে সেটি এমিট রিপ্লাই ফাংশনটি সামলায়।

```
void emit_reply(int conn, Reply rply)
{
```

ফাংশন শুরু হল।

```
//pthread_mutex_lock(&emitMutex2);
if(rply.code == 200)
{
```

ইয়াহু! পেজটি সার্ভারে পাওয় গেছে!

```
char *ok_code = "HTTP/1.1 200 OK\r\n";
write(conn, ok_code, strlen(ok_code));
```

সুখবরের একটি কোড আছে। সেটি সেট করি।

```
char *date;
date = (char *)malloc(100);
strcpy(date, "Date: ");
strcat(date, rply.date);
strcat(date, "\r\n");
write(conn, date, strlen(date));
```

তারিখ সেট করি।

```
char *server;
server = (char *)malloc(100);
strcpy(server, "Server: ");
strcat(server, rply.server);
strcat(server, "\r\n");
write(conn, server, strlen(server));
```

কে উত্তর পাঠাচ্ছে সেটি জানাই।

```
char *last_modified;
last_modified = (char *)malloc(100);
strcpy(last_modified, "Last-Modified: ");
strcat(last_modified, rply.last_modified);
```

```
strcat(last_modified, "\r\n");
write(conn, last_modified, strlen(last_modified));
```

ওয়েবপেজটি সর্বশেষ কবে আপডেট করা হয়েছে তা জানাই।

```
char *content_length;
content_length = (char *)malloc(100);
strcpy(content_length, "Content-Length: ");
int decpnt, sign;
char *p;
p = (char *)malloc(20);
p = ecvt(rply.content_length, 15, &decpnt, &sign);
//printf("\nConversion-- %s %d %d", p, decpnt, sign);

strncat(content_length, p, decpnt);
strcat(content_length, "\r\n");
write(conn, content_length, strlen(content_length));
```

ওয়েবপেজ কত বাইট সেটি জানাই।

```
char *content_type;
content_type = (char *)malloc(100);
strcpy(content_type, "Content-Type: ");
strcat(content_type, rply.content_type);
strcat(content_type, "\r\n");
write(conn, content_type, strlen(content_type));
```

এটি ছবি, ভিডিও না শুধু লেখা সেই তথ্যটি জানাই।

```
write(conn, "\r\n", 2);

int b;
char c;
FILE *fp;
fp = (FILE *) malloc(sizeof(FILE));
fp = fopen(rply.file_path, "rb");
while(!feof(fp))
{
   c = getc(fp);
   if(!feof(fp))
   {
      //printf("%c", c);
      write(conn, &c, 1);
   }
}
```

এবার পেজটি পাঠাই।

```
//pthread_mutex_unlock(&emitMutex2);
return;
}
```

ফাংশন শেষ!

```
else if(rply.code == 400)
{
   char *ok_code = "HTTP/1.1 400 Bad Request\r\n";
   write(conn, ok_code, strlen(ok_code));

   char *date;
   date = (char *)malloc(100);
   strcpy(date, "Date: ");
   strcat(date, rply.date);
   strcat(date, "\r\n");
   write(conn, date, strlen(date));
   char *server;
   server = (char *)malloc(100);
   strcpy(server, "Server: ");
   strcat(server, rply.server);
   strcat(server, "\r\n");
   write(conn, server, strlen(server));
   write(conn, "\r\n", 2);

   char *error_400;
   error_400 = (char *)malloc(300);
   strcpy(error_400, "<html><head><title>Error No: 400. Bad Request.</title>
</head><body><h2>The server cannot parse the request.</h2><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><hr>For further info mail to shehab_sust@yahoo.com.</body><html>");
   write(conn, error_400, strlen(error_400));
}
```

যদি পেজটি পাওয়া না যায় তাহলে দু:সংবাদ জানিয়ে দেই।

```
else if(rply.code == 501)
{
   char *ok_code = "HTTP/1.1 501 Method Not Implemented\r\n";
   write(conn, ok_code, strlen(ok_code));

   char *date;
   date = (char *)malloc(100);
   strcpy(date, "Date: ");
   strcat(date, rply.date);
   strcat(date, "\r\n");
   write(conn, date, strlen(date));

   char *server;
   server = (char *)malloc(100);
   strcpy(server, "Server: ");
   strcat(server, rply.server);
```

```
   strcat(server, "\r\n");
   write(conn, server, strlen(server));
   write(conn, "\r\n", 2);

   char *error_501;
   error_501 = (char *)malloc(300);
   strcpy(error_501, "<html><head><title>Error No: 501. Method Not Implemented.
</title></head><body><h2>The server only resolves the GET method.</h2><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><hr>For further info mail to shehab_sust@yahoo.com.</body>
<html>");
   write(conn, error_501, strlen(error_501));
}
```

এর মানে হল রিকোয়েস্টে কোন সমস্যা নেই কিন্তু যে মেথডে চাওয়া হয়েছে সেটির সাপোর্ট আমি এখনও দেই নি। দোষ আমারই।

```
else if(rply.code == 404)
{
   char *ok_code = "HTTP/1.1 404 File Not Found\r\n";
   write(conn, ok_code, strlen(ok_code));

   char *date;
   date = (char *)malloc(100);
   strcpy(date, "Date: ");
   strcat(date, rply.date);
   strcat(date, "\r\n");
   write(conn, date, strlen(date));

   char *server;
   server = (char *)malloc(100);
   strcpy(server, "Server: ");
   strcat(server, rply.server);
   strcat(server, "\r\n");
   write(conn, server, strlen(server));

   write(conn, "\r\n", 2);

   char *error_404;
   error_404 = (char *)malloc(300);
   strcpy(error_404, "<html><head><title>Error No: 404. File Not Found.</title>
</head><body><h2>The server could not found the requested url.</h2><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><hr>For further info mail to shehab_sust@yahoo.com.</body>
<html>");
   write(conn, error_404, strlen(error_404));
}
```

এর মানে হল ওয়েবসার্ভারে অনুরোধকৃত পেজের কোন অস্তিত্বই নেই।

```
else if(rply.code == 403)
{
   char *ok_code = "HTTP/1.1 403 Forbidden\r\n";
```

```
   write(conn, ok_code, strlen(ok_code));

   char *date;
   date = (char *)malloc(100);
   strcpy(date, "Date: ");
   strcat(date, rply.date);
   strcat(date, "\r\n");
   write(conn, date, strlen(date));

   char *server;
   server = (char *)malloc(100);
   strcpy(server, "Server: ");
   strcat(server, rply.server);
   strcat(server, "\r\n");
   write(conn, server, strlen(server));

   write(conn, "\r\n", 2);

   char *error_403;
   error_403 = (char *)malloc(300);
   strcpy(error_403, "<html><head><title>Error No: 403. Forbidden.</title>
</head><body><h2>You are not authorized to access this url.</h2><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><hr>For further info mail to shehab_sust@yahoo.com.</body><html>");

   write(conn, error_403, strlen(error_403));
}
```

এর মানে হল পেজটি আছে কিন্তু দেখানো যাবেনা।

```
} //end of emit_reply
```

ফাংশন শেষ!

ওয়েবসার্ভারে যে শুধুমাত্র পেজের রিকোয়েস্টই আসে তা নয়। এটি একটি ছবির রিকোয়েস্ট হতে পারে, ভিডিওর রিকোয়েস্ট হতে পারে বা অন্যকিছুও হতে পারে। রিকোয়েস্ট থেকে সেটি বুঝে নিয়ে আবার রিপ্লাইতে বলে দিতে হয় কি ধরনের ডেটা পাঠানো হচ্ছে। গেটমাইমটাইপ নামে একটি ফাংশন এটি সামলায়।

```
char *getMimeType(char *str)
{
   printf("\nentered mime type");
   int i;
   //defined here the file extension
   char *fileExt[MAX];
   fileExt[0] = "html";
   fileExt[1] = "htm";
   fileExt[2] = "txt";
   fileExt[3] = "gif";
   fileExt[4] = "jpg";
   fileExt[5] = "qt";

   //defined here the mime type
```

```
   char *mType[MAX];
   mType[0] = "text/html";
   mType[1] = "text/html";
   mType[2] = "text/plain";
   mType[3] = "image/gif";
   mType[4] = "image/jpeg";
   mType[5] = "video/quicktime";

   for(i = 0; i < 6; i++)
   {
      if(strcmp(str, fileExt[i]) == 0)
      {
         printf("\nmatched");
         return mType[i];
      }
   } //end of for loop
   printf("\nreturning ERROR");
   char *er = "ERROR";
   return er;
} //end of getMimetype
```

ফাংশনটির আইডিয়া খুব সহজ। রিকোয়েস্ট থেকে মাইমটাইপ পড়ে সে অনুযায়ী রিপ্লাইয়ের মাইমটাইপ ঠিক করে দিবে।

রিপ্লাই হিসেবে যে বাইটগুলো পাঠানো হয় সেগুলো গুছানোর জন্য প্রিপেয়ার রিপ্লাই নামে একটি ফাংশন ব্যবহার করা হয়।

```
Reply prepare_reply(int code, Request rq)
{
```

ফাংশন শুরু হল।

```
// extracting the file extension
char *drive, *direc, *fname, *ext;

drive = (char *) malloc (3);
direc = (char *) malloc (66);
fname = (char *) malloc (9);
ext = (char *) malloc (5);

Reply rpl;
```

শুরুতে মেমোরীতে কিছু জায়গা গুছিয়ে নেই।

```
if(code == 200)
{
```

এর মানে হল আমরা ওয়েবসাইট ভিজিটরকে জানাতে যাচ্ছে যে পেজটি সার্ভারে পাওয়া গেছে।

```
rpl.code = 200;

time_t lt;
lt = time(NULL);
strcpy(rpl.date, ctime(&lt));
rpl.date[strlen(rpl.date) - 1] = '\0';

strcpy(rpl.server, "TinyThreadedServer/0.1a");
```

কিছু খুচরো তথ্য গুছিয়ে নেই।

```
char dir[100];
getcwd(dir, 100);
//printf("\ncurrent dir is %s", dir);
//printf("\nfile path is %s", rq.filePath);
strcat(dir, "/");
strcat(dir, rq.filePath);
//printf("\nabsolute file path is %s", dir);
char *path, *p;
path = (char *) malloc (80);
p = (char *) malloc (80);
path = strtok(dir, "/");
strcpy(p, "/");
strcat(p, path);
do{
   path = strtok('\0', "/");
   if(path)
   {
      strcpy(ext, path);
      strcat(p, "/");
      strcat(p, path);
   }
} while(path);
//printf("\nthe c++ file path %s", p);
```

হার্ডডিস্কে কোথায় ফাইলটি আছে খুঁজে নেই।

```
char *e;
e = (char *)malloc(10);
e = strtok(ext, ".");
e = strtok('\0', ".");
//printf("\nfile extension is %s", e);
```

ফাইলের এক্সটেনশনটি জেনে নেই।

```
FILE *fp;
fp = (FILE *) malloc(sizeof(FILE));
if((fp = fopen(p, "r")) == NULL)
{
   printf("\nFILE NOT FOUND.\nERROR CODE 404");
```

```
    rpl.code = 404;
    return rpl;
}
```

ফাইলটি পড়া যায় কিনা দেখি।

```
struct stat buff;
stat(p, &buff);
//printf("\nSize: %ld\ntime of last modification: %s", buff.st_size,
ctime(&buff.st_mtime));

strcpy(rpl.last_modified, ctime(&buff.st_mtime));
rpl.last_modified[strlen(rpl.last_modified) - 1] = '\0';
```

ফাইলটি সর্বশেষ কবে আপডেট করা হয়েছে সেই তথ্যটি টুকে নেই।

```
rpl.content_length = buff.st_size;
```

ফাইলের সাইজ কত সেই তথ্যটি নেই।

```
if(strcmp(getMimeType(e), "ERROR") == 0)
{
    rpl.code = 403;
    fclose(fp);
    return rpl;
}
```

ফাইলটি পড়ার পারমিশন আছে কিনা দেখি।

```
    strcpy(rpl.content_type, getMimeType(e));
    strcpy(rpl.file_path, p);
    fclose(fp);
}
```

আরো কিছু তথ্য নেই।

```
else if (code == 400)
{
    rpl.code = 400;

    time_t lt;
    lt = time(NULL);
    strcpy(rpl.date, ctime(&lt));
    rpl.date[strlen(rpl.date) - 1] = '\0';

    strcpy(rpl.server, "TinyThreadedServer/0.1a");
```

```
}
```

এর মানে হল ফাইলটি পাওয়া যায় নি।

```
else if (code == 501)
{
   rpl.code = 501;

   time_t lt;
   lt = time(NULL);
   strcpy(rpl.date, ctime(&lt));
   rpl.date[strlen(rpl.date) - 1] = '\0';

   strcpy(rpl.server, "TinyThreadedServer/0.1a");
}
```

এর মানে হল ফাইলটি আছে কিন্তু পড়ার পারমিশন নেই।

```
   return rpl;
} //end of prepare_reply
```

ফাংশন শেষ।

কোন ব্রাউজার যখন রিকোয়েস্ট পাঠায় তখন সেখানে অনেক ধরণের তথ্য থাকে। সেগুলো ধরে ধরে বুঝার জন্য পারস রিকোয়েস্ট ফাংশনটি ব্যবহার করা হয়।

```
Request parse_request(char *rbuff)
{
   char *fileName;
   char *hostName;
   int j, k, l, m, n;
   Request r;
```

ফাংশন শুরু হল।

```
fileName = (char *) malloc (100);
hostName = (char *) malloc (100);
```

মেমোরীতে একটু জায়গা নেই।

```
for(j = 5, k = 0; ;j++, k++)
{
   if(rbuff[j] == ' ') break;
   fileName[k] = rbuff[j];
} //end of for loop
```

```
fileName[k]= '\0';
//printf("\nThe valid file path is %s", fileName);
strcpy(r.filePath, fileName);
```

কোন ওয়েবপেজ দেখতে চায় সেটি জেনে নেই।

```
//finding the host
char *p;
p = strstr(rbuff, "Host: ");

for(l = 0; l < 6; l++)
{
   p++;
} //end of for loop

for(m = 0, n = 0; ; m++, n++)
{
   if(*p == '\r' || *p == '\n') break;
   hostName[n] = *p;
   p++;
} //end of for loop
hostName[n]= '\0';

fflush(stdout);
strcpy(r.hostName, hostName);
```

রিকোয়েস্টটি কোন কম্পিউটার থেকে এসেছে জেনে নেই।

```
   return r;
} // end of parse_request
```

ফাংশন শেষ।

একটি ওয়েব পেজ দেখার রিকোয়েস্টে অনেক ধরণের মেথড থাকতে পারে, যেমন, গেট, পুট ইত্যাদি। আমার এই সার্ভার শুধু মাত্র গেট সাপোর্ট করে। এই ব্যাপারগুলো সামলানোর জন্য ফাইন্ড মেথড ফাংশনটি ব্যবহার করা হয়।

```
int find_method (char *req)
{
   pthread_mutex_lock(&srvrMutex2);
   //printf("\nin the find_method function\n%s\nthe size of the request is: %d",
req, strlen(req));

   char *method_list[6];
   method_list[0] = (char *) malloc (10);
   strcpy(method_list[0], "PUT");
   method_list[1] = (char *) malloc (10);
   strcpy(method_list[1], "HEAD");
   method_list[2] = (char *) malloc (10);
   strcpy(method_list[2], "POST");
   method_list[3] = (char *) malloc (10);
```

```
    strcpy(method_list[3], "DELETE");
    method_list[4] = (char *) malloc (10);
    strcpy(method_list[4], "TRACE");
    method_list[5] = (char *) malloc (10);
    strcpy(method_list[5], "CONNECT");

    int i, j = 0;
    char *r;
    r = (char *) malloc (10);
    for(i = 0; i < strlen(req); i++)
    {
       if(req[i] == ' ') break;
       r[i] = req[i];
    } //end of for loop
    r[i] = '\0';
    //printf("\nExtracted method is %s", r);

    if(strcmp(r, "GET") == 0)
    {
       //printf("\nit's a GET method !!!");
       pthread_mutex_unlock(&srvrMutex2);
       return 0;
    }

    for(i = 0; i < 7; i++)
    {
       if(strcmp(r, method_list[i]) == 0)
       {
          printf("\n%s matched with %s", r, method_list[i]);
          j = 1;
          break;
       }
    } //end of for loop

    if(j == 1)
    {
       pthread_mutex_unlock(&srvrMutex2);
       return 1;
    }
    else
    {
       pthread_mutex_unlock(&srvrMutex2);
       return -1;
    }
 } // end of find_method
```

গেট মেথড হলে আমি আগাবো না হলে স্যরি বলে দিব।

কোড এখানেই শেষ! পড়ার জন্য ধন্যবাদ! এখন একটু মাথা খাটিয়ে বের করা আমার ওয়েব সার্ভারের সোর্স কোডের ফাইলের নাম allizom রেখেছি কেন?

July 4, 2015 subeen টিউটোরিয়াল ওয়েব সার্ভার, থ্রেড, প্রোগ্রামিং, সি