EA872 Laboratório de Programação de Software Básico
Atividade 6

Vinícius Esperança Mantovani **RA 247395**
Entrega (limite): 13/09/2023, 13:00

OBSERVAÇÃO: Todo o relatório é feito de acordo com a forma imposta na Atividade 8, ou seja, explica e documenta todo o código e todas as saídas.
Os códigos serão todos disponibilizados junto com o relatório pelo envio no moodle.
No arquivo comprimido enviado, existe um ".sh" que compila automaticamente o parser.

Atividade 4)

Os códigos das funções se encontram a seguir:

```c
C/C++

HEAD:
/**
 * @brief Responde a requisicao do usuario por um recurso, devolvendo um
cabecalho e o recurso.
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int headRes(char *caminho, char *recurso, char *connection, int fd_reg){ //a
string caminho deve ser acompanhada de um / no final
       int fd1;
       char buf[1000];
       int fd2;

       struct stat statind; //usado para index.html
       struct stat statwel; //usado para welcome.html
       struct stat statbuf; //usado para o arquivo do caminho
```

```c
        char caminho_recurso[100] = "";

        //concatena o caminho e o nome do arquivo:
        strcpy(caminho_recurso, caminho);
        strcat(caminho_recurso, recurso);

        //guarda os status do recurso em statbuf
        int st_out = stat(caminho_recurso, &statbuf);

        //caso o arquivo nao exista
        if (st_out == -1 && errno == ENOENT) {
        //imprime a saida necessaria (resposta ao cliente)
        printsAnswer(404, connection, caminho_recurso,0, fd_reg);
        return 404;
        }

        //switch entre casos de arquivo regular ou diretorio
        switch (statbuf.st_mode & S_IFMT)
        {
        //caso seja arquivo regular:
        case S_IFREG :
               if((statbuf.st_mode & S_IRUSR) != 0){
               //imprime a saida necessaria (resposta ao cliente)
               printsAnswer(200, connection, caminho_recurso,0, fd_reg);
               return 0;
               } else{
               printsAnswer(403, connection, caminho_recurso,0, fd_reg);
               return 403;
               }
               break;

        //caso seja arquivo de diretorio
        case S_IFDIR :
               //caso seja permitida a varredura no diretorio
               if((statbuf.st_mode & S_IXUSR) != 0){

               char cpy1[100]; //copia de caminho_recurso com "index.html"
concatenado
               char cpy2[100]; //copia de caminho_recurso com "welcome.html"
concatenado
               int rBytes; //numero de bytes lido
               int existsInd; //flag para identificar existencia ou nao do
arquivo index.html
               int existsWel; //flag para identificar existencia ou nao do
arquivo welcome.html

               //faz o processo de copia e concatenacao de cpy1 e cpy2
               strcpy(cpy1, caminho_recurso);
```

```
                strcpy(cpy2, caminho_recurso);
                strcat(cpy1, "/index.html");
                strcat(cpy2, "/welcome.html");
                existsInd = stat(cpy1, &statind);
                existsWel = stat(cpy2, &statwel);

                //caso não existam index.html nem welcome.html
                if((existsInd == -1 && errno == ENOENT) &&
                        (existsWel == -1 && errno == ENOENT)){
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(404, connection, "",0, fd_reg);
                        return 404;
                }
                //caso exista e haja permissao para leitura index.html:
                else if(existsInd != -1 && (statind.st_mode & S_IRUSR) != 0){
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(200, connection, cpy1,0, fd_reg);
                        return 0;

                //caso exista e haja permissao para leitura welcome.html:
                } else if(existsWel != -1 && (statwel.st_mode & S_IRUSR) !=0){
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(200, connection, cpy2,0, fd_reg);
                        return 0;

                //caso em que pelo menos um existe mas nenhum tem permissao
para leitura:
                } else{
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(403, connection, caminho_recurso,0,
fd_reg);
                        return 403;
                }
                }
                //caso nao haja permissao de leitura
                else if((statbuf.st_mode & S_IXUSR) == 0){
                //imprime a saida necessaria (resposta ao cliente)
                printsAnswer(403, connection, caminho_recurso,0, fd_reg);
                return 403;
                }
                break;
        }
        printf("EOFunction_Test");
        return 0;
}


GET:
```

```
/**
 * @brief Responde a requisicao do usuario por um recurso, devolvendo um
cabecalho e o recurso.
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int getRes(char *caminho, char *recurso, char *connection, int fd_reg){ //a
string caminho deve ser acompanhada de um / no final
       int fd1;
       char buf[50];
       int fd2;

       struct stat statind; //usado para index.html
       struct stat statwel; //usado para welcome.html
       struct stat statbuf; //usado para o arquivo do caminho
       char caminho_recurso[100] = "";

       //concatena o caminho e o nome do arquivo:
       strcpy(caminho_recurso, caminho);
       strcat(caminho_recurso, recurso);

       //guarda os status do recurso em statbuf
       int st_out = stat(caminho_recurso, &statbuf);

       //caso o arquivo nao exista
       if (st_out == -1 && errno == ENOENT) {
       //imprime a saida necessaria (resposta ao cliente)
       printsAnswer(404, connection, caminho_recurso, 0, fd_reg);
       return 404;
       }


       //switch entre casos de arquivo regular ou diretorio
       switch (statbuf.st_mode & S_IFMT)
       {
       //caso seja arquivo regular:
       case S_IFREG :
              if((statbuf.st_mode & S_IRUSR) != 0){
              //imprime a saida necessaria (resposta ao cliente)
              printsAnswer(200, connection, caminho_recurso,0, fd_reg);
              fd1 = open(caminho_recurso, O_RDONLY, 0600);
              int rBytes; //numero de bytes lido

              while((rBytes = read(fd1, buf, sizeof(buf))) != 0){
                     write(1, buf, rBytes);
                     write(fd_reg, buf, rBytes);
```

```
            }
            write(1, "\n", 1);
            write(fd_reg, "\n", 1);

            return 0;
            } else{
            printsAnswer(403, connection, caminho_recurso,0, fd_reg);
            return 403;
            }
            break;

      //caso seja arquivo de diretorio
      case S_IFDIR :
            //caso seja permitida a varredura no diretorio
            if((statbuf.st_mode & S_IXUSR) != 0){

            char cpy1[100]; //copia de caminho_recurso com "index.html"
concatenado
            char cpy2[100]; //copia de caminho_recurso com "welcome.html"
concatenado
            int rBytes; //numero de bytes lido
            int existsInd; //flag para identificar existencia ou nao do
arquivo index.html
            int existsWel; //flag para identificar existencia ou nao do
arquivo welcome.html

            //faz o processo de copia e concatenacao de cpy1 e cpy2
            strcpy(cpy1, caminho_recurso);
            strcpy(cpy2, caminho_recurso);
            strcat(cpy1, "/index.html");
            strcat(cpy2, "/welcome.html");
            existsInd = stat(cpy1, &statind);
            existsWel = stat(cpy2, &statwel);

            //caso não existam index.html nem welcome.html
            if((existsInd == -1 && errno == ENOENT) &&
                  (existsWel == -1 && errno == ENOENT)){
                  //imprime a saida necessaria (resposta ao cliente)
                  printsAnswer(404, connection, "",0, fd_reg);
                  char webspace[100];
                  strcpy(webspace, caminho);
                  strcat(webspace, "/erro404.html");

                  fd2 = open(webspace, O_RDONLY, 0600);

                  while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                        write(1, buf, rBytes);
                  }
```

```
                        write(1, "\n", 1);
                        write(fd_reg, "\n", 1);

                        return 404;
                }
                //caso exista e haja permissao para leitura index.html:
                else if(existsInd != -1 && (statind.st_mode & S_IRUSR) != 0){
                        fd2 = open(cpy1, O_RDONLY, 0600);
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(200, connection, cpy1,0, fd_reg);

                        while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                                write(1, buf, rBytes);
                        }
                        write(1, "\n", 1);
                        write(fd_reg, "\n", 1);
                        return 0;

                //caso exista e haja permissao para leitura welcome.html:
                } else if(existsWel != -1 && (statwel.st_mode & S_IRUSR) !=0){
                        fd2 = open(cpy2, O_RDONLY, 0600);
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(200, connection, cpy2,0, fd_reg);

                        while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                                write(1, buf, rBytes);
                        }
                        write(1, "\n", 1);
                        write(fd_reg, "\n", 1);
                        return 0;

                //caso em que pelo menos um existe mas nenhum tem permissao
para leitura:
                } else{
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(403, connection, caminho_recurso,0,
fd_reg);

                        char webspace[100];
                        strcpy(webspace, caminho);
                        strcat(webspace, "/erro403.html");

                        fd2 = open(webspace, O_RDONLY, 0600);

                        while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                                write(1, buf, rBytes);
                        }
                        write(1, "\n", 1);
                        write(fd_reg, "\n", 1);

                        return 403;
                }
                }

                //caso nao haja permissao de varredura
                else if((statbuf.st_mode & S_IXUSR) == 0){
                //imprime a saida necessaria (resposta ao cliente)
                printsAnswer(403, connection, caminho_recurso,0, fd_reg);
                return 403;
                }
                break;
        }
        printf("EOFunction_Test");
        return 0;
}
```

O código será disponibilizado em sua integridade juntamente com o relatório enviado.

Atividade 5)

As funções são apresentadas abaixo:

```
C/C++

TRACE:

/**
 * @brief Responde a requisicao do usuario, devolvendo um 200 OK e um
cabecalho.
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int traceRes(char *caminho, char *recurso, char *connection, int fd_reg){
//a string caminho deve ser acompanhada de um / no final

        char caminho_recurso[100] = "";

        //concatena o caminho e o nome do arquivo:
        strcpy(caminho_recurso, caminho);
        strcat(caminho_recurso, recurso);
        printsAnswer(200, connection, caminho_recurso, 0, fd_reg);
        return 0;

}


OPTIONS:
/**
 * @brief Responde a requisicao do usuario por um recurso, devolvendo um
cabecalho e os comandos possiveis.
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int optionsRes(char *caminho, char *recurso, char *connection, int fd_reg){

        char caminho_recurso[100] = "";

        //concatena o caminho e o nome do arquivo:
        strcpy(caminho_recurso, caminho);
        strcat(caminho_recurso, recurso);

        write(1, "GET, HEAD, TRACE, OPTIONS\n", 27);
        write(fd_reg, "GET, HEAD, TRACE, OPTIONS\n", 27);

        printsAnswer(200, connection, caminho_recurso, 1, fd_reg);

        return 0;
}
```

Atividade 6)

A função de tratamento de erros se encontra abaixo:

```
C/C++
/**
 * @brief Imprime o codigo de saida e a mensagem associada a ele
 *
 * @param code int indicadora do codigo de retorno
*/
int printCode(int code, int fd_reg){
        int eh200 = 0;

        switch (code)
        {
        case 404:
```

```
        write(1, "HTTP/1.1 404 File Not Found\n", 28);
        write(fd_reg, "HTTP/1.1 404 File Not Found\n", 28);

        break;

        case 403:
        write(1, "HTTP/1.1 403 Forbidden\n", 23);
        write(fd_reg, "HTTP/1.1 403 Forbidden\n", 23);

        break;

        case 200:
        write(1, "HTTP/1.1 200 OK\n", 16);
        write(fd_reg, "HTTP/1.1 200 OK\n", 16);

        eh200++;
        break;

        default:
        break;
        }

        return eh200;
}
```

No entanto, a função responsável por imprimir as páginas html de erro é a seguinte, que também é responsável por imprimir todo o cabeçalho das respostas a requisições:

```
C/C++
/**
 * @brief Imprime a saida de acordo com o codigo passado como parametro
 *
 * @param code inteiro indicador do codigo passado do HTTP
 * @param connection string indicadora do tipo de conexao
 * @param file string indicadora do caminho do arquivo
 * @param options inteiro que indica se a funcao foi chamada por OPTIONS
 *
*/
void printsAnswer(int code, char *connection, char *file, int options, int
fd_reg){

        int eh200 = 0;

        // printa o codigo e a mensagem associada a ele
        if(options != 1){
```

```
        eh200 = printCode(code, fd_reg); //usado para verificar se o codigo eh
200 (nao houve erro)
        } else{
        eh200 = 1;
        }

        time_t time_now; //usado na aquisicao do tempo
        time(&time_now); //usado na aquisicao do tempo

        struct tm *info = localtime(&time_now); //usado na aquisicao do tempo
        struct tm *info_mod; //usado na aquisicao do tempo da ultima
modificacao
        struct stat file_stats; //usado para obter informacoes sobre o arquivo

        char date[30]; //armazena a data
        char last_mod[30]; //armazena a data da ultima modificacao
        char lenght[10]; //tamanho do arquivo

        strcpy(date, asctime(info)); //usado na aquisicao do tempo
        write(1, "Date: ", 6);
        write(fd_reg, "Date: ", 6);

        write(1, date, 25); //imprime a data
        write(fd_reg, date, 25); //imprime a data

        write(1, "Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca
Mantovani\n", 63);
        write(fd_reg, "Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca
Mantovani\n", 63);
        write(1, "Connection: ", 12);
        write(fd_reg, "Connection: ", 12);
        write(1, connection, 10); //imprime o tipo de conexao
        write(fd_reg, connection, 10); //imprime o tipo de conexao
        write(1, "\n", 1);
        write(fd_reg, "\n", 1);
        write(1, "Last-Modified: ", 15);
        write(fd_reg, "Last-Modified: ", 15);

        if(stat(file, &file_stats) == -1){
        write(1, "\n", 1);
        write(fd_reg, "\n", 1);
        } else{
        info_mod = localtime(&file_stats.st_mtime);
        strcpy(last_mod, asctime(info_mod));
        write(1, last_mod, 25);
        write(fd_reg, last_mod, 25);

        }
        write(1, "Content-Length: ", 16);
        write(fd_reg, "Content-Length: ", 16);

        sprintf(lenght, "%d", file_stats.st_size);

        write(1, lenght, strlen(lenght)); //imprime o tamanho do recurso
        write(fd_reg, lenght, strlen(lenght)); //imprime o tamanho do recurso

        write(1, "\n", 1);
        write(fd_reg, "\n", 1);

        write(1, "Content-Type: ", 14);
        write(fd_reg, "Content-Type: ", 14);

        char aux[20];

        if(eh200 == 1){
        strtok(file, ".");
        strcpy(aux, strtok(NULL, "."));
        if(strcmp(aux, "html") == 0){
        write(1, "text/html", 9); //imprime o tipo de dado do recurso
        write(fd_reg, "text/html", 9); //imprime o tipo de dado do recurso


        }
        } else{
        write(1, "text/html", 9);
        write(fd_reg, "text/html", 9);
        }

        write(1, "\n\n", 2);
        write(fd_reg, "\n\n", 2);


}
```

Atividade 7)

A seguir, estão os programas ".l" e ".y" escritos com as funções feitas integradas a eles:

```
C/C++

sepcomm.l:

%{
	#include "sepcomm.tab.h"
	#include <stdio.h>
	#include <string.h>

	struct Lista{ //define a estrutura da lista ligada
	struct Lista *proximo;
	char params[50][50];
	char comando[50];
	};
	int count_com = 0; //conta o numero de comandos na lista ate o momento
	int count_par = 0; //conta o numero de parametros na lista componente
do comando sendo analisado no momento
	char reqs[10]; //armazena o tipo de requisicao feito
	char ress[50]; //armazena o recurso requisitado
	char conn[50];

	struct Lista Comandos; //declaracao do primeiro elemento da lista de
comandos
	struct Lista *Atual_Comandos = &Comandos; //declaracao do ponteiro que
sempre aponta para o ultimo elemento adicionado na lista de comandos

	int verifica_comando = 0; //verifica se algum comando foi passado na
linha (1 se tem, 0 cc)
%}

%START       param
%x           comman
comments     ^#+.+$
command      ^([^:#]+)
doublepoints :
parameter    [^,\n#]+
space        [ ]+
comma        ,
req          ^([^ \n#]+)
res          [/]([^ \n#]+)
end_fst      "HTTP/1.1"

%%
{comments}   {
                   sscanf(yytext, "%s", &yylval.string_value);
                   return COMMENT;
             }

<comman>{command}  {
                   count_par = 0; //comeca a contar o numero de parametros
deste comando
                   struct Lista *i = &Comandos;
```

```
                    while(i->proximo != NULL){ //     teste para ver se foram
armazenados dcorretamente os comandos anteriores e o primeiro de seus
parametros
                            //printf("-------->comando anterior: %s\n",
i->comando);
                            //printf("-------->param 1: %s\n", i->params[0]);
                            i = i->proximo;
                    }

                    sscanf(yytext, "%s\n", &yylval.string_value);

                    if(count_com == 0){ //caso seja o primeiro comando, nao
cria pula para novo espaco da lista
                            strcpy(Atual_Comandos->comando, yytext);
                            Atual_Comandos->proximo = (struct Lista
*)malloc(sizeof(struct Lista));
                            count_com++;
                    } else{ //nao eh o primeiro comando, logo, pula para o
proximo espaco
                            Atual_Comandos = Atual_Comandos->proximo;
                            strcpy(Atual_Comandos->comando, yytext);
                            Atual_Comandos->proximo = (struct Lista
*)malloc(sizeof(struct Lista));
                            count_com++;
                    }
                    //printf("-------->comando atual: %s\n",
Atual_Comandos->comando); //teste para ver se o comando foi devidamente
adicionado
                    verifica_comando = 1;
                    return COMMAND;
             }

{req}        {
                    sscanf(yytext, "%s", &yylval.string_value);
                    strcpy(reqs, yytext);
                    return REQUISITION;
             }

{res}        {
                    sscanf(yytext, "%s", &yylval.string_value);
                    strcpy(ress, yytext);
                    return RESOURCE;
             }

{end_fst}           {
                    BEGIN comman;
                    sscanf(yytext, "%s", &yylval.string_value);
                    return ENDFST;
```

```
              }

<comman>{doublepoints}  {
                    BEGIN param; //entra no modo de recepcao de parametros
                    sscanf(yytext, "%s", &yylval.string_value);
                    return DOUBLEPNTS;
              }

\n            {
                    verifica_comando = 0;
                    BEGIN comman; //volta para o modo de recepcao de comandos
                    return LINEBREAK;
              }

<param>{parameter}          {
                            sscanf(yytext, "%s", &yylval.string_value);

                            if(verifica_comando == 1){ //verifica se a linha
contem realmente um comando
                                  strcpy(Atual_Comandos->params[count_par],
yytext);

                                  if(strcmp(yytext, "keep-alive")){
                                  strcpy(conn, "keep-alive");
                                  } else if(strcmp(yytext, "close")){
                                  strcpy(conn, "close");
                                  }

                                  count_par++;
                            }

                            return PARAMETER;
                            }

{comma}       {
                    sscanf(yytext, "%s", &yylval.string_value);
                    return COMMA;
              }

{space}       ;


%%


sepcomm.y:

%{
	#include <stdio.h>
	#include <string.h>
	#include <sys/types.h>
	#include <sys/stat.h>
	#include <unistd.h>
	#include <time.h>
	#include <stdlib.h>
	#include <errno.h>
	#include <fcntl.h>

	int yylex(void);
	int yyerror(char const *s);
	extern char reqs[10];
	extern char ress[50];
	extern char conn[50];
%}

%union {
	char *string_value;
	float float_value;
	char char_value;
}

%token <string_value> DOUBLEPNTS
%token <string_value> PARAMETER
%token <string_value> COMMA
%token <string_value> COMMAND
%token <string_value> LINEBREAK
%token <string_value> COMMENT
%token <string_value> REQUISITION
%token <string_value> RESOURCE
%token <string_value> ENDFST

%%

total         :      line
              |      total line
              |      fst_line
              ;

line          :      COMMAND DOUBLEPNTS parameters LINEBREAK

              |      DOUBLEPNTS parameters LINEBREAK   {
                                                      printf("ERRO: não foi
passado nenhum comando nesta linha\n");
                                                }
              |      COMMAND DOUBLEPNTS LINEBREAK
```

```
                |       COMMENT LINEBREAK
                ;

parameters      :       parameters COMMA PARAMETER
                |       PARAMETER
                ;

fst_line        :       fst_line LINEBREAK
                |       fst_line ENDFST
                |       fst_line RESOURCE
                |       REQUISITION
                ;



%%

/**
 * @brief Imprime o codigo de saida e a mensagem associada a ele
 *
 * @param code int indicadora do codigo de retorno
*/
int printCode(int code, int fd_reg){
        int eh200 = 0;

        switch (code)
        {
        case 404:
                write(1, "HTTP/1.1 404 File Not Found\n", 28);
                write(fd_reg, "HTTP/1.1 404 File Not Found\n", 28);

                break;

        case 403:
                write(1, "HTTP/1.1 403 Forbidden\n", 23);
                write(fd_reg, "HTTP/1.1 403 Forbidden\n", 23);

                break;

        case 200:
                write(1, "HTTP/1.1 200 OK\n", 16);
                write(fd_reg, "HTTP/1.1 200 OK\n", 16);

                eh200++;
                break;

        default:
                break;
```

```c
	}

	return eh200;
}

/**
 * @brief Imprime a saida de acordo com o codigo passado como parametro
 *
 * @param code inteiro indicador do codigo passado do HTTP
 * @param connection string indicadora do tipo de conexao
 * @param file string indicadora do caminho do arquivo
 * @param options inteiro que indica se a funcao foi chamada por OPTIONS
 *
*/
void printsAnswer(int code, char *connection, char *file, int options, int
fd_reg){

	int eh200 = 0;

	// printa o codigo e a mensagem associada a ele
	if(options != 1){
	eh200 = printCode(code, fd_reg); //usado para verificar se o codigo eh
200 (nao houve erro)
	} else{
	eh200 = 1;
	}

	time_t time_now; //usado na aquisicao do tempo
	time(&time_now); //usado na aquisicao do tempo

	struct tm *info = localtime(&time_now); //usado na aquisicao do tempo
	struct tm *info_mod; //usado na aquisicao do tempo da ultima
modificacao
	struct stat file_stats; //usado para obter informacoes sobre o arquivo

	char date[30]; //armazena a data
	char last_mod[30]; //armazena a data da ultima modificacao
	char lenght[10]; //tamanho do arquivo

	strcpy(date, asctime(info)); //usado na aquisicao do tempo
	write(1, "Date: ", 6);
	write(fd_reg, "Date: ", 6);

	write(1, date, 25); //imprime a data
	write(fd_reg, date, 25); //imprime a data

	write(1, "Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca
Mantovani\n", 63);
```

```
        write(fd_reg, "Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca
Mantovani\n", 63);
        write(1, "Connection: ", 12);
        write(fd_reg, "Connection: ", 12);
        write(1, connection, 10); //imprime o tipo de conexao
        write(fd_reg, connection, 10); //imprime o tipo de conexao
        write(1, "\n", 1);
        write(fd_reg, "\n", 1);
        write(1, "Last-Modified: ", 15);
        write(fd_reg, "Last-Modified: ", 15);

        if(stat(file, &file_stats) == -1){
        write(1, "\n", 1);
        write(fd_reg, "\n", 1);
        } else{
        info_mod = localtime(&file_stats.st_mtime);
        strcpy(last_mod, asctime(info_mod));
        write(1, last_mod, 25);
        write(fd_reg, last_mod, 25);

        }

        write(1, "Content-Length: ", 16);
        write(fd_reg, "Content-Length: ", 16);

        sprintf(lenght, "%d", file_stats.st_size);

        write(1, lenght, strlen(lenght)); //imprime o tamanho do recurso
        write(fd_reg, lenght, strlen(lenght)); //imprime o tamanho do recurso

        write(1, "\n", 1);
        write(fd_reg, "\n", 1);

        write(1, "Content-Type: ", 14);
        write(fd_reg, "Content-Type: ", 14);

        char aux[20];

        if(eh200 == 1){
        strtok(file, ".");
        strcpy(aux, strtok(NULL, "."));
        if(strcmp(aux, "html") == 0){
                write(1, "text/html", 9);  //imprime o tipo de dado do recurso

                write(fd_reg, "text/html", 9);  //imprime o tipo de dado do
recurso

        }
```

```c
        } else{
        write(1, "text/html", 9);
        write(fd_reg, "text/html", 9);
        }

        write(1, "\n\n", 2);
        write(fd_reg, "\n\n", 2);


}

/**
 * @brief Responde a requisicao do usuario por um recurso, devolvendo um
cabecalho e o recurso.
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int getRes(char *caminho, char *recurso, char *connection, int fd_reg){ //a
string caminho deve ser acompanhada de um / no final
        int fd1;
        char buf[50];
        int fd2;

        struct stat statind; //usado para index.html
        struct stat statwel; //usado para welcome.html
        struct stat statbuf; //usado para o arquivo do caminho
        char caminho_recurso[100] = "";

        //concatena o caminho e o nome do arquivo:
        strcpy(caminho_recurso, caminho);
        strcat(caminho_recurso, recurso);

        //guarda os status do recurso em statbuf
        int st_out = stat(caminho_recurso, &statbuf);

        //caso o arquivo nao exista
        if (st_out == -1 && errno == ENOENT) {
        //imprime a saida necessaria (resposta ao cliente)
        printsAnswer(404, connection, caminho_recurso, 0, fd_reg);
        return 404;
        }


        //switch entre casos de arquivo regular ou diretorio
        switch (statbuf.st_mode & S_IFMT)
        {
```

```
        //caso seja arquivo regular:
        case S_IFREG :
                if((statbuf.st_mode & S_IRUSR) != 0){
                //imprime a saida necessaria (resposta ao cliente)
                printsAnswer(200, connection, caminho_recurso,0, fd_reg);
                fd1 = open(caminho_recurso, O_RDONLY, 0600);
                int rBytes; //numero de bytes lido

                while((rBytes = read(fd1, buf, sizeof(buf))) != 0){
                        write(1, buf, rBytes);
                        write(fd_reg, buf, rBytes);
                }
                write(1, "\n", 1);
                write(fd_reg, "\n", 1);

                return 0;
                } else{
                printsAnswer(403, connection, caminho_recurso,0, fd_reg);
                return 403;
                }
                break;

        //caso seja arquivo de diretorio
        case S_IFDIR :
                //caso seja permitida a varredura no diretorio
                if((statbuf.st_mode & S_IXUSR) != 0){

                char cpy1[100]; //copia de caminho_recurso com "index.html"
concatenado
                char cpy2[100]; //copia de caminho_recurso com "welcome.html"
concatenado
                int rBytes; //numero de bytes lido
                int existsInd; //flag para identificar existencia ou nao do
arquivo index.html
                int existsWel; //flag para identificar existencia ou nao do
arquivo welcome.html

                //faz o processo de copia e concatenacao de cpy1 e cpy2
                strcpy(cpy1, caminho_recurso);
                strcpy(cpy2, caminho_recurso);
                strcat(cpy1, "/index.html");
                strcat(cpy2, "/welcome.html");
                existsInd = stat(cpy1, &statind);
                existsWel = stat(cpy2, &statwel);

                //caso não existam index.html nem welcome.html
                if((existsInd == -1 && errno == ENOENT) &&
                        (existsWel == -1 && errno == ENOENT)){
```

```
                    //imprime a saida necessaria (resposta ao cliente)
                    printsAnswer(404, connection, "",0, fd_reg);
                    char webspace[100];
                    strcpy(webspace, caminho);
                    strcat(webspace, "/erro404.html");

                    fd2 = open(webspace, O_RDONLY, 0600);

                    while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                          write(1, buf, rBytes);
                    }
                    write(1, "\n", 1);
                    write(fd_reg, "\n", 1);

                    return 404;
             }
             //caso exista e haja permissao para leitura index.html:
             else if(existsInd != -1 && (statind.st_mode & S_IRUSR) != 0){
                    fd2 = open(cpy1, O_RDONLY, 0600);
                    //imprime a saida necessaria (resposta ao cliente)
                    printsAnswer(200, connection, cpy1,0, fd_reg);

                    while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                          write(1, buf, rBytes);
                    }
                    write(1, "\n", 1);
                    write(fd_reg, "\n", 1);
                    return 0;

             //caso exista e haja permissao para leitura welcome.html:
             } else if(existsWel != -1 && (statwel.st_mode & S_IRUSR) !=0){
                    fd2 = open(cpy2, O_RDONLY, 0600);
                    //imprime a saida necessaria (resposta ao cliente)
                    printsAnswer(200, connection, cpy2,0, fd_reg);

                    while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                          write(1, buf, rBytes);
                    }
                    write(1, "\n", 1);
                    write(fd_reg, "\n", 1);
                    return 0;

             //caso em que pelo menos um existe mas nenhum tem permissao
para leitura:
             } else{
                    //imprime a saida necessaria (resposta ao cliente)
                    printsAnswer(403, connection, caminho_recurso,0,
fd_reg);
```

```
                    char webspace[100];
                    strcpy(webspace, caminho);
                    strcat(webspace, "/erro403.html");

                    fd2 = open(webspace, O_RDONLY, 0600);

                    while((rBytes = read(fd2, buf, sizeof(buf))) != 0){
                          write(1, buf, rBytes);
                    }
                    write(1, "\n", 1);
                    write(fd_reg, "\n", 1);

                    return 403;
              }
              }

              //caso nao haja permissao de varredura
              else if((statbuf.st_mode & S_IXUSR) == 0){
              //imprime a saida necessaria (resposta ao cliente)
              printsAnswer(403, connection, caminho_recurso,0, fd_reg);
              return 403;
              }
              break;
        }
        printf("EOFunction_Test");
        return 0;
}

/**
 * @brief Responde a requisicao do usuario por um recurso, devolvendo um
cabecalho e o recurso.
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int headRes(char *caminho, char *recurso, char *connection, int fd_reg){ //a
string caminho deve ser acompanhada de um / no final
        int fd1;
        char buf[1000];
        int fd2;

        struct stat statind; //usado para index.html
        struct stat statwel; //usado para welcome.html
        struct stat statbuf; //usado para o arquivo do caminho
        char caminho_recurso[100] = "";
```

```
       //concatena o caminho e o nome do arquivo:
       strcpy(caminho_recurso, caminho);
       strcat(caminho_recurso, recurso);

       //guarda os status do recurso em statbuf
       int st_out = stat(caminho_recurso, &statbuf);

       //caso o arquivo nao exista
       if (st_out == -1 && errno == ENOENT) {
       //imprime a saida necessaria (resposta ao cliente)
       printsAnswer(404, connection, caminho_recurso,0, fd_reg);
       return 404;
       }

       //switch entre casos de arquivo regular ou diretorio
       switch (statbuf.st_mode & S_IFMT)
       {
       //caso seja arquivo regular:
       case S_IFREG :
              if((statbuf.st_mode & S_IRUSR) != 0){
              //imprime a saida necessaria (resposta ao cliente)
              printsAnswer(200, connection, caminho_recurso,0, fd_reg);
              return 0;
              } else{
              printsAnswer(403, connection, caminho_recurso,0, fd_reg);
              return 403;
              }
              break;

       //caso seja arquivo de diretorio
       case S_IFDIR :
              //caso seja permitida a varredura no diretorio
              if((statbuf.st_mode & S_IXUSR) != 0){

              char cpy1[100]; //copia de caminho_recurso com "index.html"
concatenado
              char cpy2[100]; //copia de caminho_recurso com "welcome.html"
concatenado
              int rBytes; //numero de bytes lido
              int existsInd; //flag para identificar existencia ou nao do
arquivo index.html
              int existsWel; //flag para identificar existencia ou nao do
arquivo welcome.html

              //faz o processo de copia e concatenacao de cpy1 e cpy2
              strcpy(cpy1, caminho_recurso);
              strcpy(cpy2, caminho_recurso);
              strcat(cpy1, "/index.html");
```

```
                strcat(cpy2, "/welcome.html");
                existsInd = stat(cpy1, &statind);
                existsWel = stat(cpy2, &statwel);

                //caso não existam index.html nem welcome.html
                if((existsInd == -1 && errno == ENOENT) &&
                        (existsWel == -1 && errno == ENOENT)){
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(404, connection, "",0, fd_reg);
                        return 404;
                }
                //caso exista e haja permissao para leitura index.html:
                else if(existsInd != -1 && (statind.st_mode & S_IRUSR) != 0){
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(200, connection, cpy1,0, fd_reg);
                        return 0;

                //caso exista e haja permissao para leitura welcome.html:
                } else if(existsWel != -1 && (statwel.st_mode & S_IRUSR) !=0){
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(200, connection, cpy2,0, fd_reg);
                        return 0;

                //caso em que pelo menos um existe mas nenhum tem permissao
para leitura:
                } else{
                        //imprime a saida necessaria (resposta ao cliente)
                        printsAnswer(403, connection, caminho_recurso,0,
fd_reg);
                        return 403;
                }
                }
                //caso nao haja permissao de leitura
                else if((statbuf.st_mode & S_IXUSR) == 0){
                //imprime a saida necessaria (resposta ao cliente)
                printsAnswer(403, connection, caminho_recurso,0, fd_reg);
                return 403;
                }
                break;
        }
        printf("EOFunction_Test");
        return 0;
}


/**
 * @brief Responde a requisicao do usuario, devolvendo um 200 OK e um
cabecalho.
```

```c
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int traceRes(char *caminho, char *recurso, char *connection, int fd_reg){
//a string caminho deve ser acompanhada de um / no final

	char caminho_recurso[100] = "";

	//concatena o caminho e o nome do arquivo:
	strcpy(caminho_recurso, caminho);
	strcat(caminho_recurso, recurso);

	printsAnswer(200, connection, caminho_recurso, 0, fd_reg);
	return 0;

}

/**
 * @brief Responde a requisicao do usuario por um recurso, devolvendo um
cabecalho e os comandos possiveis.
 *
 * @param caminho string indicadora do caminho para o recurso;
 * @param recurso string indicadora do nome do recurso;
 * @param connection string indicadora do tipo de conexao;
*/
int optionsRes(char *caminho, char *recurso, char *connection, int fd_reg){

	char caminho_recurso[100] = "";

	//concatena o caminho e o nome do arquivo:
	strcpy(caminho_recurso, caminho);
	strcat(caminho_recurso, recurso);

	write(1, "GET, HEAD, TRACE, OPTIONS\n", 27);
	write(fd_reg, "GET, HEAD, TRACE, OPTIONS\n", 27);

	printsAnswer(200, connection, caminho_recurso, 1, fd_reg);

	return 0;
}

void main(int argc, char *argv[]){
	char buffer[50];
	int rBytess = 0;

	//printa a requisicao na tela
```

```c
int fd_in = open(argv[2], O_RDONLY, 0600);
while((rBytess = read(fd_in, buffer, sizeof(buffer))) != 0){
write(1, buffer, rBytess);
}
close(fd_in);

//altera stdin e stdout
close(0);
close(1);
fd_in = open(argv[2], O_RDONLY, 0600);
int fd_out = open(argv[3], O_WRONLY | O_CREAT, 0600);
int fd_reg = open(argv[4], O_WRONLY | O_CREAT | O_APPEND, 0600);
rBytess = 0;

//escreve a entrada em registro

write(fd_reg, "requisicao:\n\n", 13);
while((rBytess = read(fd_in, buffer, sizeof(buffer))) != 0){
write(fd_reg, buffer, rBytess);
}
write(fd_reg, "\n\n", 2);
close(fd_in);
fd_in = open(argv[2], O_RDONLY, 0600);

yyparse();

char recurso[20]; //armazena o nome do recurso
char caminho[50]; //armazena o caminho para o recurso (sem seu nome)

//distingue entre casos de requisicao e imprime a resposta na saida:

write(fd_reg, "resposta:\n\n", 11);

if(strcmp(reqs ,"GET") == 0){
getRes(argv[1], ress, conn, fd_reg);
} else if(strcmp(reqs, "HEAD") == 0){
headRes(argv[1], ress, conn, fd_reg);
} else if(strcmp(reqs, "TRACE") == 0){
traceRes(argv[1], ress, conn, fd_reg);
} else if(strcmp(reqs, "OPTIONS") == 0 ){
optionsRes(argv[1], ress, conn, fd_reg);
}

write(fd_reg, "\n", 1);

close(0);
close(1);
close(fd_reg);
```

```
}

int yyerror (char const *s){
        fprintf (stderr, "%s\n", s);
}
```

Atividade 8)

A seguir, encontram-se as saídas para cada um dos arquivos de requisições desenvolvidos em aula:

```
C/C++
Requisição 1:

GET /dir1 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Reposta 1:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:44:09 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<title>ERRO 404 FILE NOT FOUND</title>
```

```
</head>
<body>
Um erro foi identificado:
O arquivo requisitado não existe!
</body>
</html>


Requisição 2:

GET /dir2 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 2:

HTTP/1.1 403 Forbidden
Date: Thu Sep 14 17:45:35 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 14:59:25 2023
Content-Length: 4096
Content-Type: text/html



Requisição 3:

HEAD /dir1 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
```

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 3:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:46:44 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html


Requisição 4:

HEAD /dir2 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 4:

HTTP/1.1 403 Forbidden
Date: Thu Sep 14 17:50:30 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 14:59:25 2023
Content-Length: 4096
```

```
Content-Type: text/html


Requisição 5:

HEAD /dir1/dir11 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 5:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:52:29 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html


Requisição 6:

HEAD /dir1/texto1.html HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```

```
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 6:

HTTP/1.1 200 OK
Date: Thu Sep 14 17:54:08 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 22:24:38 2023
Content-Length: 283
Content-Type: text/html


Requisição 7:

HEAD /dir1/texto2.c HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 7:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:55:46 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html


Requisição 8:

TRACE / HTTP/1.1
```

```
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

Resposta 8:

```
HTTP/1.1 200 OK
Date: Thu Sep 14 17:57:03 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep 13 22:15:39 2023
Content-Length: 4096
Content-Type:
```

Requisição 9:

```
GET /dir1/texto1.html HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

```
Resposta 9:

HTTP/1.1 200 OK
Date: Thu Sep 14 17:57:54 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 22:24:38 2023
Content-Length: 283
Content-Type: text/html


Requisição 10:

GET /dir1/dir11 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

Resposta 10:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 20:36:50 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<title>ERRO 404 FILE NOT FOUND</title>
</head>
<body>
Um erro foi identificado:
O arquivo requisitado não existe!
</body>
```

```
</html>


Requisição 11:

GET /dir1/texto2.html HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 11:

HTTP/1.1 403 Forbidden
Date: Thu Sep 14 20:39:07 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 15:10:33 2023
Content-Length: 268
Content-Type: text/html


Requisição 12:

OPTIONS / HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```

```
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 12:

GET, HEAD, TRACE, OPTIONS
Date: Thu Sep 14 21:03:30 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep 13 22:15:39 2023
Content-Length: 4096
Content-Type:


Requisição 13:

GET / HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Resposta 13:

HTTP/1.1 200 OK
Date: Thu Sep 14 21:05:46 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 15:44:35 2023
Content-Length: 267
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<title>INDEX</title>
```

```
</head>
<body>
Aqui está o conteúdo da página. Altere-o de maneira a
facilitar a identificação de cada página. Ajudaria dizer aqui
qual é a página e onde ela está na estrutura de diretórios.
</body>
</html>
```

Abaixo, agora, encontra-se o resultado no “registro.txt”:

```
C/C++



requisicao:

GET /dir1 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:44:09 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html
```

```
requisicao:

GET /dir2 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 403 Forbidden
Date: Thu Sep 14 17:45:35 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 14:59:25 2023
Content-Length: 4096
Content-Type: text/html


requisicao:

HEAD /dir1 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

```
resposta:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:46:44 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html


requisicao:

HEAD /dir2 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 403 Forbidden
Date: Thu Sep 14 17:50:30 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 14:59:25 2023
Content-Length: 4096
Content-Type: text/html


requisicao:

HEAD /dir1/dir11 HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:52:29 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html


requisicao:

HEAD /dir1/texto1.html HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

resposta:

HTTP/1.1 200 OK
Date: Thu Sep 14 17:54:08 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
```

```
Last-Modified: Wed Sep  6 22:24:38 2023
Content-Length: 283
Content-Type: text/html


requisicao:

HEAD /dir1/texto2.c HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 17:55:46 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html


requisicao:

TRACE / HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 200 OK
Date: Thu Sep 14 17:57:03 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep 13 22:15:39 2023
Content-Length: 4096
Content-Type: requisicao:

GET /dir1/texto1.html HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 200 OK
Date: Thu Sep 14 17:57:54 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 22:24:38 2023
Content-Length: 283
Content-Type: text/html

requisicao:

GET /dir1/dir11 HTTP/1.1
Host: localhost:2020
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 404 File Not Found
Date: Thu Sep 14 20:36:50 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified:
Content-Length: 0
Content-Type: text/html



requisicao:

GET /dir1/texto2.html HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:
```

```
HTTP/1.1 403 Forbidden
Date: Thu Sep 14 20:39:07 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 15:10:33 2023
Content-Length: 268
Content-Type: text/html


requisicao:

OPTIONS / HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

GET, HEAD, TRACE, OPTIONS
Date: Thu Sep 14 21:03:30 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep 13 22:15:39 2023
Content-Length: 4096
Content-Type: requisicao:

GET / HTTP/1.1
Host: localhost:2020
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


resposta:

HTTP/1.1 200 OK
Date: Thu Sep 14 21:05:46 2023
Server: Servidor HTTP ver. 0.1 de Vinicius Esperanca Mantovani
Connection: keep-alive
Last-Modified: Wed Sep  6 15:44:35 2023
Content-Length: 267
Content-Type: text/html
```