



Nome do Aluno: Luiz Fernando Postingel Quirino		RGA: 2016.0743.003-7
Assunto: Atividade 03	Data(s): 25-04-2021	

Atividade 03 - Programação com Sockets C (TCP) - 14/04/2021

Implementação

Foi implementado um HTTP Server simples, que serve um arquivo `".html"`.

Para testes e utilização

Compilar com os parâmetros padrão

```
1 $ gcc -ansi -Wall -pedantic server.c -o server
```

Executando com as definições padrão:

```
1 $ ./server
```

Executando com as definições (arquivo e porta) customizados:

```
1 $ ./server arquivo.html 8001
```

Ambiente de desenvolvimento

- Arch Linux x86_64
- Kernel 5.11.16-arch1-1 SMP
- GCC 10.2.0

Dificuldades encontradas

Demorei para encontrar a documentação que queria em uma linguagem acessível aos meus conhecimentos, apesar de ter um acesso tranquilo ao inglês, as terminologias que se utilizava em alguns materiais, como o código que usei de base, disponível neste vídeo[1] do canal *Eduonix Learning Solutions*, por exemplo, me pareciam muito soltos, apesar de funcionarem nos testes.

Depois de alguma pesquisa, encontrei uma cópia gratuita e em PDF pesquisável do livro *Programando em C. Para linux Unix e Windows*[2], que no capítulo 23 explica de forma bem clara e objetiva as terminologias e funções que haviam me causado confusão na literatura/exemplos que havia encontrado, inclusive do código usei como guia para compreender compreender e desenvolver uma solução ao problema apresentado.

Há algumas ressalvas e melhorias possíveis, entre elas, o arquivo `.html` lido necessita ter apenas uma linha, dado que é feita a leitura com `fgets`, esta é uma problemática que poderia ter sido resolvida em uma implementação com mais tempo.

1 Anexo I - Código fonte utilizado

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <sys/socket.h>
6 #include <sys/types.h>
7 #include <netinet/in.h>
8
9
10 int isNumeric(const char *str)
11 {
12     while(*str != '\0')
13     {
14         if(*str < '0' || *str > '9')
15             return 0;
16         str++;
17     }
18     return 1;
19 }
20
21
22 int main(int argc, char const *argv[])
23 {
24
25     FILE *html_data;
26     char response_data[4096], input[255], filename[255];
27     char http_header[4096] = "HTTP/1.1 200 OK\r\nContent-Type: text/html; charset=
    UTF-8\r\n\r\n";
28     char http_content[8192] = "";
29     int server_socket, client_socket, port;
30     struct sockaddr_in server_address;
31
32     /* Verifica parametros passados e permite customizar algumas
33      * op es de acesso
34      */
35     if (argc < 3 )
36     {
37         port=8080;
38         strcpy(filename, "index.html");
39
40         printf("\n\tEntre com um nome para o arquivo html(sem a extens o)");
41         printf("\n\tdeixe vazio para index.html\n\t");
42         fgets (input, 255, stdin);
43         input[strlen(input) - 1] = '\0';
44         if(strlen(input))
45         {
46             strcpy(filename, input);
47             strcat(filename, ".html");
48         }
49
50         printf("\n\tEntre com um numero para a porta");
51         printf("\n\tdeixe em branco para 8080\n\t");
52         fgets (input, 255, stdin);
53         input[strlen(input) - 1] = '\0';
54         if(strlen(input) && isNumeric(input))
55         {
56             port = atoi(input);
57         }
58     }
```

```

59 } else {
60     strcpy(filename, argv[1]);
61     port = atoi(argv[2]);
62 }
63
64
65 /* cria o socket
66 */
67 server_socket = socket(AF_INET, SOCK_STREAM, 0);
68
69 /* define o endereço do server
70 */
71 server_address.sin_family = AF_INET;
72 server_address.sin_port = htons(port);
73 server_address.sin_addr.s_addr = INADDR_ANY;
74
75 bind(server_socket, (struct sockaddr *)&server_address, sizeof(server_address))
76 ;
77 listen(server_socket, 5);
78
79 /* executa enquanto não recebe um sinal de cancelamento
80 */
81 printf("\n\t http://localhost:%d", port);
82 printf("\n\t Pressione Ctrl+C para finalizar\n");
83 while (1)
84 {
85     /* Limpa e reseta o cabeçalho
86     */
87     memset(http_content, '\0', sizeof(http_content));
88     strcat(http_content, http_header);
89
90     /* abre um arquivo para servir, e lê seu conteúdo
91     * ao usar o fgets, o html precisa estar formatado em apenas uma linha
92     * com um pouco mais de tempo, teria sido interessante implementar a
93     melhoria
94     */
95     html_data = fopen(filename, "r");
96     fgets(response_data, 4096, html_data);
97     fclose(html_data);
98
99     strcat(http_content, response_data);
100
101     client_socket = accept(server_socket, NULL, NULL);
102     send(client_socket, http_content, sizeof(http_content), 0);
103     close(client_socket);
104 }
105
106 return 0;
107 }

```

Observação: preferi não desmembrar o código e não criar bibliotecas em arquivos separados para facilitar os testes.

Referências

- [1] Eduonix Learning Solutions. *Socket Programming Tutorials In C For Beginners / Part 2 / Eduonix*. URL: <https://www.youtube.com/watch?v=mStnzIEprH8>.
- [2] Marcos Aurelio Pchek Laureano. *Programando em C. Para Linux, Unix e Windows*. URL: http://www.mlaureano.org/livro/Programando_C_conta.pdf.



**UNIVERSIDADE FEDERAL
DE MATO GROSSO DO SUL**