Servidor web simplificado com uso do TCP

A. Título: Criando um servidor Web simplificado com uso de TCP

B. Objetivos: Compreender a estrutura interna de servidores web, nos moldes tradicionais cliente/servidor.

C. Roteiro: O conteúdo deste laboratório pode ser explorado, considerando as subdivisões apresentadas a seguir:

Parte I – Revisão sobre as funções open, read, write e close

É sabido que a função socket é uma abstração do comando open (ou fopen) utilizado para tratamento de arquivos normais em linguagem C. Execute o comando abaixo e relembre como os descritores de arquivo podem ser úteis para realizar cópias de arquivos:

1) Em um host Linux, crie um sub-diretório com o nome **web_server_simplificado**.

2) Copie o código teste01.c para o seu sub-diretório e compile-os utilizando o gcc (ex.: gcc teste01.c –o teste01). Este arquivo realiza uma cópia de um arquivo em outro pela escrita byte-a-byte de um arquivo de origem para um arquivo destino, conforme está descrito na tabela abaixo:

```
1  /* Fundamentos de Redes - Prof. Fernando W. Cruz    */
2  /* leituraescrita em arquivos*/
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <fcntl.h>
8  #include <sys/types.h>
9  #include <sys/stat.h>
10
11 int main(int argc,char **argv) {
12   int z, fd_orig, fd_dest;
13   char getbuf[1];/* GET buffer */
14
15   if (argc != 2) {
16      printf("SINTAXE: %s <Nome_arquivo_a_ser_criado>\n", argv[0]);
17      exit(0);
18   }
19   fd_orig = open("index.html", O_RDONLY, S_IRWXO );
20   fd_dest = open(argv[1], O_RDWR|O_CREAT, S_IRWXO );
21   while ( ( z = read (fd_orig, getbuf, 1 )) > 0 ) {
22         z = write(fd_dest, getbuf, 1);
23   }/* fim-while */
24   printf("Arquivo %s criado ...\n", argv[1]);
25   close(fd_orig); close(fd_dest);
26  return 0;
27 } /* fim-main */
```

3) Para realizar um teste, copie o arquivo index.html para o mesmo diretório onde você colocou o teste01.c

4) Execute o programa teste01.c, passando como parâmetro, o nome do arquivo destino onde será feita a cópia do arquivo index.html. (ex.: ./teste01 arq01.html). Perceba que o novo arquivo é exatamente igual ao index.html

Obs.:Execute um ***man*** no sistema e descreva os parâmetros da função read e write usadas neste programa.

Parte II – Utilizando um servidor Web simplificado com fdopen, fdclose, dup e setsockoptions

Verifique o código descrito na tabela a seguir:

```
1  /*************************************/
2  /* Fundamentos de Redes de Computadores  */
3  /* Prof. Fernando W. Cruz                    */
4  /* web80.c :Este eh um Web Server extremamente simples: */
5  #include <stdio.h>
6  #include <unistd.h>
7  #include <stdlib.h>
8  #include <errno.h>
9  #include <string.h>
10 #include <fcntl.h>
11 #include <time.h>
12 #include <sys/types.h>
13 #include <sys/socket.h>
14 #include <netinet/in.h>
15 #include <sys/un.h>
16 #include <sys/uio.h>
17 #define TRUE    1
18 int main(int argc,char **argv) {
19   int z, sd;                  /* Descritor de Socket do Web Server         */
20   int novo_sd;                /* Descritor com dados do cliente     */
21   int alen;                   /* Tamanho do Endereco                */
22   struct sockaddr_in end_web, end_cli;      /* End.do Web Server e do Cliente*/
23   int b = TRUE;               /* Reutilizacao do ender.SO_REUSEADDR        */
24   FILE *rx, *tx;                   /* Stream de Leitura e Escrita               */
25   char getbuf[2048];          /* GET buffer                         */
26   time_t td;                  /* Data e hora corrente                       */
27
28   sd = socket(AF_INET,SOCK_STREAM,0);
29   /* Web address on port 80: */
30   memset(&end_web, 0, sizeof end_web);
31   end_web.sin_family     = AF_INET;
32   end_web.sin_port       = ntohs(80);
33   end_web.sin_addr.s_addr  = ntohl(INADDR_ANY);
34
35   z = bind(sd,(struct sockaddr *)&end_web, sizeof end_web);
36   /* Ativa a opcao SO_REUSEADDR :  */
37   z = setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &b, sizeof b);
38   z = listen(sd,10);
39   /* Recebe uma msg e retorna um arquivo html */
40   for (;;) {
41    /* Wait for a connect from browser: */
42     alen = sizeof end_cli;
43     novo_sd = accept(sd,(struct sockaddr *)&end_cli, &alen);
44     rx = fdopen(novo_sd,"r"); /* cria stream de leitura associada a novo_sd*/
45     tx = fdopen(dup(novo_sd),"w"); /* cria stream de escrita associada a novo_sd */
46     fgets(getbuf, sizeof getbuf, rx);
47     printf("Msg de chegada = %s\n",getbuf);
48     /* Resposta com um documento HTML */
49     fputs("<HTML>\n"
50           "<HEAD>\n"
51           "<TITLE>Pagina de teste para este pequeno Web Server</TITLE>\n"
52           "</HEAD>\n"
53           "<BODY>\n"
54           "<H1>Servidor WWW no ar !!!</H1>\n",tx);
55     time(&td);
56     fprintf(tx,"<H2>PID desse Web Server: %ld <H2>",(long)getpid());
57     fprintf(tx,"<H5>Hora da requisicao:  %s</H5>\n", ctime(&td));
58     fputs("</BODY>\n" "</HTML>\n",tx);
59     fclose(tx); fclose(rx);
60  } /* fim-for */
 return 0;
} /* fim-main */
```

5) Copie o código web80.c (vide figura acima) para o sub-diretório do servidor Linux e compile-o (ex.: gcc web80.c –o web80).
6) Certifique-se de que o arquivo index.html está residente no mesmo diretório onde o web80.c está instalado. Coloque o código web80 para executar.
7) De um outro equipamento, coloque um browser para executar e coloque o endereço IP do *host* onde está executando o web80. Observe os comandos http que o navegador passa para o servidor Web no momento de requisição das páginas.