Defesa de **Código**

Resolvendo o problema da Alice

Apresentado por:

Marjorie Ap. Daenecke, Rafael C. Alves, Beatriz R. Nahas, André L. Cecato

Sumário

1 Entendendo o problema

2 tcp.servidor.c

3 tcp.cliente.c

4 Execução dos códigos

5 Conclusões e reflexões

Entendendo **o problema**

- Toda vez que Alice liga seu computador, a **data** e a **hora** são ***resetadas*** para 01/01/2000 às 00:00:00.
- A **bateria do CMOS**, que mantém as configurações da BIOS, está ***sem carga***.
- A **placa-mãe do computador** é de um **modelo antigo** e os fornecedores ***não fabricam*** mais a peça necessária.
- **Ajustar manualmente** a data e a hora seria uma ***solução temporária***. No entanto, Alice não possui **permissões administrativas** para fazer essa alteração.

tcp.**servidor**.c

```c
void DieWithUserMessage(const char *msg, const char *detail) {
    perror(msg);
    fprintf(stderr, "%s\n", detail);
    exit(1);
}
```

```c
void HandleTCPClient(int clientSocket) {
    char buffer[BUFSIZE];
    // Pega a data e a hora do servidor(computador que está rodando o server)
    time_t currentTime = time(NULL);
    struct tm *localTime = localtime(&currentTime);
    strftime(buffer, BUFSIZE, "%Y-%m-%d %H:%M:%S", localTime);

    // Envia a data e hora para o cliente socket com o send()
    ssize_t numBytesSent = send(clientSocket, buffer, strlen(buffer), 0);

    if (numBytesSent < 0) {
        DieWithUserMessage("send() failed", "Unable to send data");
    }
    close(clientSocket);
}
```

```c
int main(int argc, char *argv[]) {
    if (argc != 2) {
        DieWithUserMessage("Parameter(s)", "<Server Port>");
    }
    in_port_t serverPort = atoi(argv[1]);

    // A função socket cria e retorna um descritor de arquivo
    int serverSocket;
    if ((serverSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0) {
        DieWithUserMessage("socket() failed", "Unable to create socket");
    }
    struct sockaddr_in serverAddress;                    //Endereço local
    memset(&serverAddress, 0, sizeof(serverAddress));    //Limpa a memória para ter certeza que não tem nada antes de iniciar
    serverAddress.sin_family = AF_INET;                  //Especifica o tipo de endereço, nesse caso é IPV4
    serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);   //Define o endereço de IP que o servidor vai usar
    serverAddress.sin_port = htons(serverPort);          //Define o numero da porta

    //Bind associa um endereço a um socket criado
    if (bind(serverSocket, (struct sockaddr*)&serverAddress, sizeof(serverAddress)) < 0) {
        DieWithUserMessage("bind() failed", "Unable to bind to port");
    }
    //prepara as conexões de entrada em uma fila de conexões pendentes
    if (listen(serverSocket, MAXPENDING) < 0) {
        DieWithUserMessage("listen() failed", "Unable to listen for connections");
    }
```

```
    for (;;) {
        struct sockaddr_in clientAddress;
        socklen_t clientAddressLen = sizeof(clientAddress);
        //o accept() extrai a primeira conexão da fila de pendentes criada pelo listen()
        int clientSocket = accept(serverSocket, (struct sockaddr*)&clientAddress, &clientAddressLen);
        if (clientSocket < 0) {
            DieWithUserMessage("accept() failed", "Unable to accept client connection");
        }
        char clientName[INET_ADDRSTRLEN];
        //o Ntop transforma os dados de binário para string
        if (inet_ntop(AF_INET, &clientAddress.sin_addr.s_addr, clientName, sizeof(clientName)) != NULL) {
            printf("Handling client %s/%d\n", clientName, ntohs(clientAddress.sin_port));
        } else {
            puts("Unable to get client address");
        }
        HandleTCPClient(clientSocket);
    }
    close(serverSocket);
    return 0;
}
```

tcp.**cliente**.c

```c
void DieWithSystemMessage(const char *msg) {
    perror(msg);
    exit(1);
}
```

```c
void DieWithUserMessage(const char *msg, const char *detail) {
    fprintf(stderr, "%s: %s\n", msg, detail);
    exit(1);
}
```

```c
int main(int argc, char *argv[]) {
    if (argc < 2 || argc > 3)
        DieWithUserMessage("Parameter(s)",
            "<Server Address> [<Server Port>]");
    char *serverIP = argv[1];
    in_port_t serverPort = (argc == 3) ? atoi(argv[2]) : 7;
    // A função socket cria e retorna um descritor de arquivo
    int sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sock < 0)
        DieWithSystemMessage("socket() failed");

    struct sockaddr_in serverAddress;                   //Endereço local
    memset(&serverAddress, 0, sizeof(serverAddress));   //Limpa a memória antes de rodar
    serverAddress.sin_family = AF_INET;                 //Especifica o tipo de endereço, nesse caso é IPV4
```

```
    int returnVal = inet_pton(AF_INET, serverIP, &serverAddress.sin_addr.s_addr);
    if (returnVal == 0)
        DieWithUserMessage("inet_pton() failed", "Invalid address string");
    else if (returnVal < 0)
        DieWithSystemMessage("inet_pton() failed");

    serverAddress.sin_port = htons(serverPort);
    //conecta o socket do cliente com o socket do servidor
    if (connect(sock, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) < 0)
        DieWithSystemMessage("connect() failed");

    char buffer[BUFSIZE];
    //o recv recebe a mensagem de um socket nesse caso do servidor
    ssize_t numBytes = recv(sock, buffer, BUFSIZE - 1, 0);
    if (numBytes < 0)
        DieWithSystemMessage("recv() failed");
    else if (numBytes == 0)
        DieWithUserMessage("recv()", "connection closed prematurely");

    buffer[numBytes] = '\0';
    printf("Recebemos a data e a hora do servidor: %s\n", buffer);

    close(sock);
    exit(0);
}
```

Execução **dos códigos**

```
beatriznahas@linux:~/BCC/Semestre_3-4/RedesComp$ gcc -o tcp_server tcp_server.c
beatriznahas@linux:~/BCC/Semestre_3-4/RedesComp$ ./tcp_server 8080
Handling client 127.0.0.1/40744
```

```
beatriznahas@linux:~/BCC/Semestre_3-4/RedesComp$ gcc -o tcp_cliente tcp_cliente.c
beatriznahas@linux:~/BCC/Semestre_3-4/RedesComp$ ./tcp_cliente 127.0.0.1 8080
Recebemos a data e a hora do servidor: 2024-08-18 13:48:42
beatriznahas@linux:~/BCC/Semestre_3-4/RedesComp$
```

Conclusões **e reflexões**

- A implementação de um sistema cliente-servidor foi eficaz em **resolver o problema** de ***data e hora*** no computador de Alice, permitindo que ela **continue a usar o computador para assinar documentos digitalmente**.

- Este projeto destaca como a **inovação** e a **aplicação** de conhecimentos técnicos podem ***minimizar limitações impostas por falhas de hardware***. Além disso, reforça a **importância** de estar preparado para lidar com a **obsolescência tecnológica**.

Obrigado **pela atenção**

Perguntas ou feedbacks?