**Universidade Federal de Minas Gerais**

**Departamento de Ciência da Computação - DCC**

Trabalho Prático 02: Sistema de múltiplas conexões

**Aluno:** Marcone Márcio da Silva Faria **Matrícula:** 2019021573

## 1. INTRODUÇÃO:

O sistema de múltiplas conexões foi implementado para coordenar múltiplas conexões simultâneas entre os equipamentos de um processo de produção permitindo a comunicação entre eles, facilitando com que a troca de informações em tempo real seja facilitada. Para essa implementação foi desenvolvido um servidor responsável por coordenar essas conexões à medida em que os equipamentos entram e saem da rede e equipamentos que solicitam informações de outros equipamentos por intermédio do servidor**.**

Nesse sistema podem ser estabelecidas até 15 conexões simultâneas gerenciadas pelo servidor da aplicação por meio da utilização de múltiplas threads. Ademais, o protocolo de aplicação funciona sobre o protocolo TCP, sendo as mensagens entregues sobre um canal de bytes com garantia de entrega em ordem, assim as mensagens são entregues de forma sequencial do servidor para o cliente e vice-versa.

O estabelecimento deste protocolo de comunicação se dá por meio de instruções enviadas pelo usuário a partir de um dos equipamentos previamente conectados com o servidor. Para a implementação do protocolo são dadas as seguintes instruções de comunicação:

- **Requisição de entrada de equipamento:** requisita para o servidor a adição de mais um equipamento na rede, a instrução de execução é dada por:

  ```
  ./equipment <server IP> <server port>
  ```

  em que é solicitada a inserção de um equipamento na rede, nesse caso, havendo alguma conexão disponível, o servidor retorna para o terminal específico do equipamento qual o ID atribuído a ele naquela conexão em específico, e, em caso do número de conexões estabelecidas dor superior ao limite, o servidor envia uma mensagem para o equipamento e finaliza a conexão**.**
- **Encerrar conexão:** solicita o encerramento da conexão:

  ```
  close connection
  ```

  envia uma mensagem ao servidor informando o encerramento da conexão entre aquele equipamento em específico com o servidor, bem como limpa o valor da informação captada pelo equipamento possibilitando que aquele ID possa ser utilizado novamente por outro equipamento.
- **Listar equipamentos:** exibe a lista de equipamentos conectados ao servidor naquele momento por meio do comando:

  ```
  list equipment
  ```

  assim, esse comando retorna somente o ID dos equipamentos que estão conectados com o servidor naquele momento, não listando aqueles que tiveram sua conexão encerrada previamente.
- **Requisitar informação:** solicita a informação lida pelo equipamento mediante o comando:

  ```
  request information from [equipment_id]
  ```

  requisita para o servidor a informação de um equipamento passando-se o ID do mesmo, caso o equipamento esteja conectado o servidor retorna esse valor para o equipamento que solicitou a informação ou uma mensagem de erro caso contrário.

## 2. IMPLEMENTAÇÃO:

A primeira parte da implementação do trabalho corresponde ao estudo e análise dos conceitos presentes na programação em redes, em especial a conexão multithread entre servidor e clientes, aprendendo como utilizar os recursos das bibliotecas e os conceitos relacionados a esse tipo de comunicação. Em seguida foi feita a estruturação do cliente do sistema implementado.

Desse modo, com o auxílio da biblioteca Socket.h foi feita a inicialização do cliente a partir do método `socket()` e a espera dos dados do servidor via diretiva `recv()`, quando feita essa conexão é enviada então a instrução por meio do método `send()` obtendo-se a resposta novamente via `recv()`. Em seguida, o servidor, também implementado utilizando-se uma lógica semelhante.

Logo no início da aplicação o vetor de 15 posições que armazena as informações dos equipamentos e controla a fila de espera foi inicializada com valores default (-1), sendo esse valor modificado a cada vez que uma nova conexão é estabelecida.

O servidor aguarda a conexão do cliente e quando ela finalmente chega, é aceita prontamente pelo método `accept()`**.** Após receber uma conexão do cliente e aceitá-la, o servidor recebe a solicitação pelo método `recv()` e processa os comandos digitados pelo usuário verificando se a instrução foi passada corretamente**.** Após os dados serem manipulados, o servidor envia um buffer contendo a informação da solicitação pelo método `send()` para o cliente correspondente e conclui a conexão, algo semelhante também ocorre com o cliente à medida que é necessário enviar alguma informação de encerramento de conexão ou adição de um novo equipamento.

Ademais, para a estruturação dessa funcionalidade também foi implementada a biblioteca util.c que, além de conter funções que auxiliam a manipulação de sockets também contém funções auxiliares e constantes que atuam na execução dos processos:

- **`countEntryNumbers():`** conta quantas entradas foram dadas na execução, separando cada entrada por um espaço de modo a facilitar a leitura dos identificadores.

## 3. DESAFIOS:

Foram encontrados diversos desafios ao longo do desenvolvimento do trabalho, o primeiro deles foi justamente o primeiro contato com a programação multithread em redes. Segundamente, outro empecilho encontrado foi novamente o baixo nível da linguagem C que não apresenta um tipo bem estruturado para strings e funções de manipulação com um nível de abstração maior, se tornando bem complexo de realizar todas as operações necessárias.

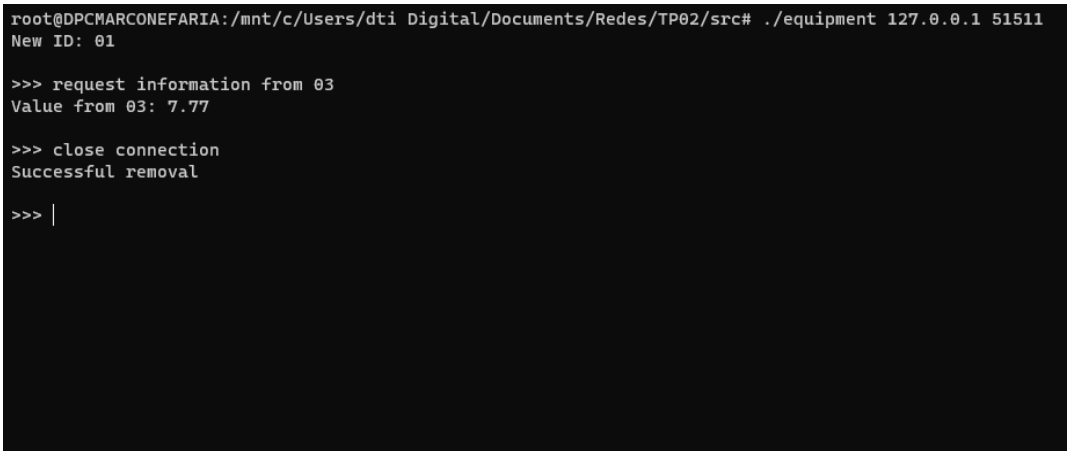
Além disso, foram encontradas dificuldades em transmitir as mensagens entre servidor e cliente, de modo a coordenar a resposta aos comandos e também entre os clientes. Esse último tipo de transmissão, por sua vez, não foi possível de ser implementado, foram pesquisados em diversos sites, fóruns e documentações da linguagem C, especialmente a comunidade StackOverflow a respeito da transmissão broadcast das mensagens mas não foi possível estruturar alguma função ou um socket que pudesse realizar o envio e recebimento dessas mensagens de uma maneira correta.

## 4. RESULTADOS E CONCLUSÃO:

Realizada a implementação da solução, foram realizados alguns testes de execução como é possível verificar abaixo, como dito anteriormente, não foi possível implementar a transmissão broadcast e por isso algumas mensagens não são mostradas nos terminais dos clientes.

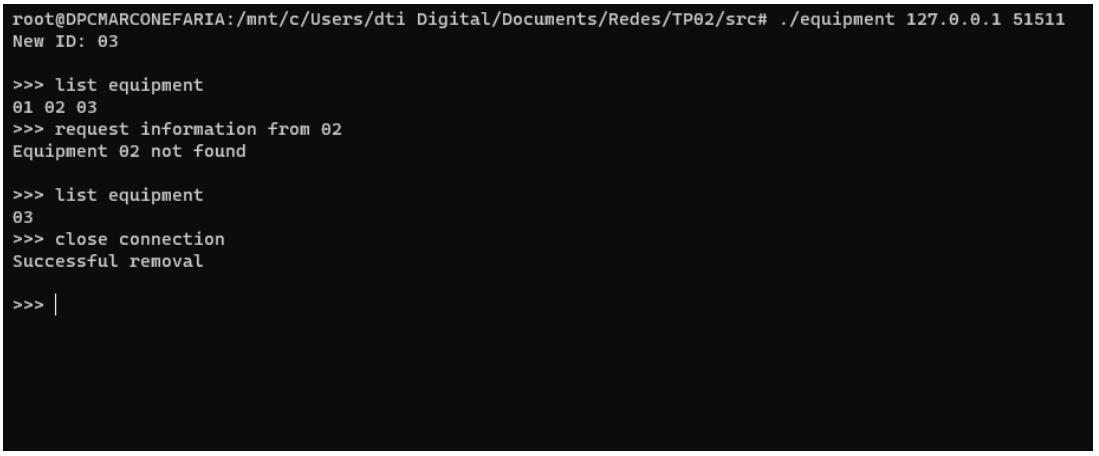
### a. Servidor:


### b. Equipamento 1:




### c. Equipamento 2:

**d. Equipamento 3:**




**e. Equipamento 16:**


Como é possível verificar alguns requisitos são cumpridos de maneira satisfatória, como a conexão multithread entre o servidor e clientes (até 15 simultaneamente) e a execução de todos os tipos de instrução, além do tratamento de significativa parte dos erros que podem ser ocasionados na execução do programa. Contudo, alguns resultados não foram obtidos, como a transmissão broadcast e a comunicação e transmissão de mensagens entre os clientes.