Universidade Federal de Minas Gerais

Departamento de Ciência da Computação - DCC

Trabalho Prático 01: Servidor de Mensagens - Sistema Supervisório

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

1. INTRODUÇÃO:

O sistema supervisório foi implementado para coletar e armazenar dados dos equipamentos de um processo de produção através de sensores de modo a auxiliar no gerenciamento e monitoramento da linha de produção de uma fábrica. Existem ao todo 4 equipamentos na planta dessa indústria: Esteira, Guindaste, Ponte Rolante e Empilhadeira**.**

Em cada um desses equipamentos podem ser instalados quatro tipos de sensores: temperatura, pressão, velocidade e corrente, atuando-se esses sensores podemos então obter valores correspondentes às grandezas aferidas**.** Por questões de simplificação de código e execução, os sensores sempre assumem valores randômicos no intervalo de 0 a 10.

Ademais, é importante mencionar que o sistema possui uma limitação no número de sensores que podem ser instalados simultaneamente nos equipamentos, de modo que apenas 15 dos 16 sensores podem ser utilizados de maneira síncrona. Por fim, cada sensor e cada equipamento possui um número de identificação único como é possível verificar abaixo:

Equipamento	Id
Esteira	01
Guindaste	02
Ponte Rolante	03
Empilhadeira	04

Sensor	Id
Temperatura	01
Pressão	02
Velocidade	03
Corrente	04

O sistema funciona no modelo cliente-servidor através da interação entre uma Central de Monitoramento (cliente) e uma Estação Remota (servidor)**.** Sendo que o servidor pode se conectar ao cliente por meio de um protocolo de internet, seja ele IPV4 ou IPV6. O estabelecimento deste protocolo de comunicação se dá por meio de instruções enviadas pelo operário diretamente da planta para a Central de monitoramento sendo exibidas e recebidas pela Estação Remota.

Para a implementação desse protocolo são dadas as seguintes instruções de comunicação:

- **Instalar Sensor:** adiciona um sensor em um equipamento, através do comando:

  `add sensor [sensor_id] in equipment_id`

  em que também é possível instalar até três sensores a partir de uma única instrução. Em caso de sucesso, a Estação Remota retorna a mensagem: `sensor [sensor_id] added`**.** Caso o(s) sensor(es) já esteja(m) instalado(s) é exibida a mensagem `sensor [sensor_id] already exists in [equipment_id]` e em caso do número de sensores instalados no equipamento ou na planta de um modo geral atingir seu limite de alocação é exibida a mensagem `limit exceeded`**.**

- **Remover Sensor:** solicita a remoção de um sensor em um equipamento por meio do comando:

  `remove sensor [sensor_id] in [equipmento_id]`

  também sendo possível remover até três sensores a partir de uma única instrução. Em caso de sucesso, essa instrução retornará a mensagem retorna `sensor [sensor_id] removed` e caso o sensor não exista no equipamento requisitado a mensagem exibida será `sensor [Sensor_id] does not exist in [equipment_id]`.
- **Consultar equipamento:** exibe a lista de sensores instalados em um equipamento por meio do comando:

  `list sensors in [equipment_id]`

  que retorna uma lista de identificadores dos sensores instalados em caso de sucesso ou então a mensagem de erro `there are no sensors installed` caso nenhum sensor tenha sido alocado.
- **Consultar variável de processo:** solicita os dados da leitura dos sensores de um dado equipamento mediante o comando:

  `read [sensor_id]1 … [sensor_id]n in [equipment_id]`

  o retorno são os valores na ordem que foi requerida a leitura em caso de sucesso ou `not installed` caso algum sensor ainda não tenha sido instalado e consequentemente realizado sua leitura.

2. IMPLEMENTAÇÃO:

A primeira parte da implementação do trabalho corresponde ao estudo e análise dos conceitos presentes na programação em redes, aprendendo como utilizar os recursos das bibliotecas e os conceitos relacionados a comunicação cliente-servidor. Em seguida foi feita a estruturação do cliente do sistema implementado, que nesse caso corresponde à Central de Monitoramento da fábrica.

Desse modo, com o auxílio da biblioteca Socket.h doi feita a inicialização do cliente a partir do método `socket()` e a conexão com o servidor via diretiva `connect()`, quando feita essa conexão é enviada então a instrução por meio do método `send()` obtendo-se a resposta via `receiv()`. Em seguida, o servidor, que no caso corresponde à Estação Remota foi também implementado utilizando-se uma lógica semelhante.

Logo no início da aplicação a matriz que armazena os dados dos sensores foi inicializada com valores default (-1) como podemos ver abaixo. É importante ressaltar que as linhas da matriz simbolizam os sensores e as colunas, os equipamentos.

	Esteira	Guindaste	Ponte Rolante	Empilhadeira
Temperatura	–1.00	–1.00	–1.00	–1.00
Pressão	–1.00	–1.00	–1.00	–1.00
Velocidade	–1.00	–1.00	–1.00	–1.00
Corrente	–1.00	–1.00	–1.00	–1.00

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 chama a função `handle()`. Após os dados serem manipulados, o servidor envia um buffer contendo a informação da solicitação pelo método `send()` para o cliente e conclui a conexão.

Ademais, para a estruturação dessa funcionalidade também foi implementada a biblioteca handlers.c que, além de conter funções que processam as instruções principais (add, read, list e remove) também contém funções auxiliares que atuam na execução dos processos:

a. **`countEntryNumbers():`** conta quantas entradas foram dadas na execução, separando cada entrada por um espaço de modo a facilitar a leitura dos identificadores..
b. **`validateEntry():`** verifica se o valor das entradas relacionadas ao identificador dos sensores e equipamentos compreende o intervalo de 1 a 4.
c. **`cleanString():`** facilita o tratamento de instruções com múltiplas entradas.

3. DESAFIOS:

Foram encontrados diversos desafios ao longo do desenvolvimento do trabalho, o primeiro deles foi justamente o primeiro contato com a programação em redes e também com a biblioteca socket.h. Segundamente, outro empecilho encontrado foi 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.

Por esse motivo, a execução deste trabalho foi possível graças a pesquisas em sites, fóruns e documentações da linguagem C, especialmente a comunidade StackOverflow. Além disso, como dito anteriormente, o processamento de strings em C de um modo geral se mostrou como um dos maiores desafios de modo que algumas partes do código não puderam ser totalmente otimizadas utilizando-se de variáveis e chamadas desnecessárias.

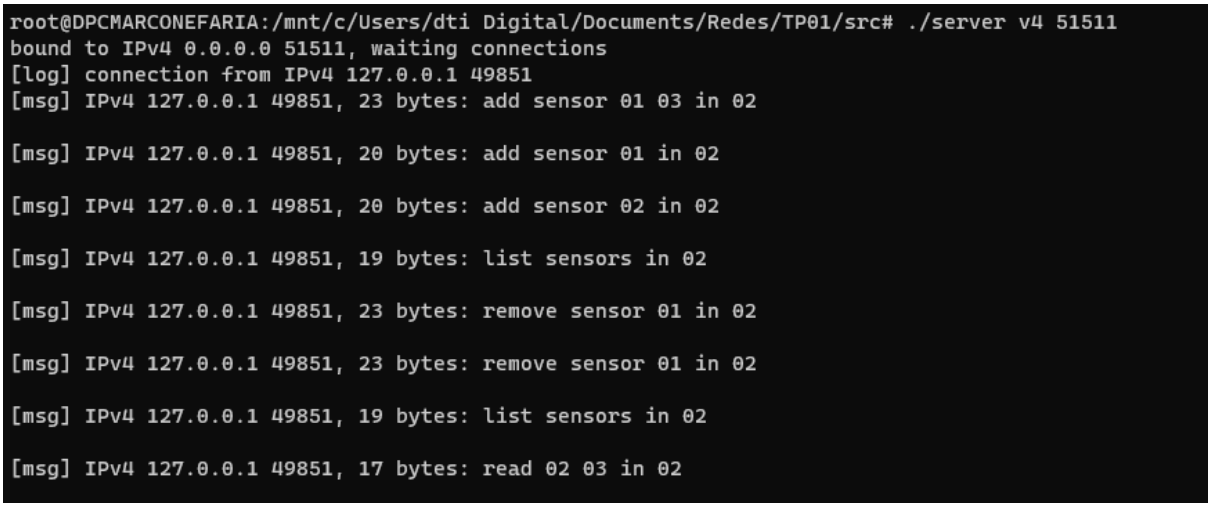
4. RESULTADOS E CONCLUSÃO:

Realizada a implementação da solução, foram realizados alguns testes de execução como é possível verificar abaixo:

```
root@DPCMARCONEFARIA:/mnt/c/Users/dti Digital/Documents/Redes/TP01/src# ./client 127.0.0.1 51511
connected to IPv4 127.0.0.1 51511
>>>add sensor 01 03 in 02
sensor 01 03 added
>>>add sensor 01 in 02
sensor 01 already exists in 02

>>>add sensor 02 in 02
sensor 02 added
>>>list sensors in 02
01 02 03
>>>remove sensor 01 in 02
sensor 01 removed
>>>remove sensor 01 in 02
sensor 01 does not exist in 02

>>>list sensors in 02
02 03
>>>read 02 03 in 02
7.77 8.86
```



Como é possível verificar alguns requisitos são cumpridos de maneira satisfatória, como a conexão por IPV4 ou IPV6 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 de maneira satisfatória, como o `read` de sensores que não foram instalados, sendo retornado algum erro que não foi possível identificar, além da ordem correta de listagem dos sensores de acordo com a ordem de instalação dos mesmos.