NOME: Jeferson Rodrigues Silva
DISCIPLINA: Sistemas Operacionais de Tempo Real
PROFESSOR: Osmar Marchi Dos Santos
DATA: 24/06/2019

TRABALHO FINAL

INTRODUÇÃO

O sistema tem como ideia principal simular o tempo de viagem de 3 diferentes tipos meios de transporte e calcular qual é o melhor baseado no menor tempo de viagem. São 3 tipos de meios de transporte disponíveis para o cálculo: carro, ônibus e metrô. Todos os cálculos são feitos no simulador, o monitor apenas envia os comandos para que os cálculos sejam realizados conforme a entrada de dados dada pelo usuário. Vale ressaltar que todos os cálculos e comparações feitas pelo simulador são realizados por funções periódicas, esses dados são salvos em um buffer e posteriormente é encaminhado para que o monitor exiba a informação para o usuário.

IMPLEMENTAÇÃO

Conforme descrição do trabalho, foram implementadas 3 tarefas periódicas.

- void *average_speed(void *arg)

Função que calcula a velocidade média do trajeto caso seja selecionado um meio de transporte válido. Ao final do cálculo a função converte o resultado da operação para char e salva as informações no buffer através de uma função auxiliar: ***save_b(char *arg)***, função essa que é responsável também por atualizar os valores de índice do buffer e da variável que indica se o buffer está vazio.
A periodicidade da tarefa é controlada pela função: **make_periodic(TIME_SPEED, &info)**, onde TIME_SPEED é definida no início do código através de um **#define**.

- void* delay(void *arg)

Função que calcula os tempos de trajeto de acordo com cada tipo de transporte. A função é setada a partir de comandos do usuário e calcula os valores de acordo com algumas condições de tempo e distância. Para fazer o cálculo foi implementada a fórmula básica da física que diz que $t = \Delta S / \Delta V$média. Para o cálculo do valor final foram utilizados valores médios de velocidade segundo dados dos agentes de trânsito do estado de São Paulo no ano de 2016. Os valores de velocidade são previamente obtidos pela função **average_speed** e posteriormente são utilizados nesse cálculo, por meio do uso de variáveis globais. Foram adicionados cantantes de atraso, considerando o tempo de espera que temos nas linhas de ônibus e metrôs da cidade de São Paulo. Há ainda um tempo de atraso adicional que é atribuído aos chamados horários de pico na cidade (entre 18 e 19hs). Esses valores foram

adicionados apenas para fins de maior similaridade com os valores obtidos na prática para estes cálculos.

- void* best_opt(void *arg)

Por fim já com as velocidades setadas e os tempos de atraso calculados, a função best_opt calcula qual é a melhor opção, ou qual a opção com o menor período de tempo. Aqui basicamente foi implementado um comparador que envia constantes para o monitor, indicando qual opção é a mais vantajosa para o usuário final.

COMANDOS

- 0, meio de transporte - Resulta na velocidade média do trajeto de acordo com o meio de transporte escolhido. São 3 opções de meios de transporte diferentes:

    1. Carro
    2. Ônibus
    3. Metrô

- 1, meio de transporte - Resulta no tempo de deslocamento do trajeto de acordo com o meio de transporte escolhido. São 3 opções de meios de transporte diferentes e elas seguem o mesmo padrão utilizado no comando anterior.

- 2, meio de transporte Retorna qual a melhor opção de meio de transporte baseada na comparação do tempo de deslocamento de cada uma, neste caso independe o meio de transporte selecionado, pois o cálculo leva em conta apenas os dados de tempo já calculados previamente por outras tarefas .

VARIÁVEIS DE CONDIÇÃO

É utilizada uma flag de controle (empty) indicando se o buffer contém ou não informações.
A variável de condição **cond** é utilizada dentro de um laço while, na função send_results, esperando até que o buffer contenha alguma informação.

MUTEX

No código foram utilizados dois mutexes: **protect** e **mutex**.
O primeiro é utilizado para modificações das variáveis globais assim como a para escrita e leitura do buffer de informações, utilizado nas funções average_speed, delay, best_opt e send_results. O segundo é utilizado para criação das threads das funções periódicas e para a escrita no socket, utilizado na função send_results.

COMUNICAÇÃO

A comunicação entre o monitor e o simulador ocorre por meio das seguintes funções:

Monitor recebe dados do simulador:	**recv(sockfd,buffer,50,0);**
Monitor envia dados para o simulador:	**send(sockfd,buffer,50,0);**
Simulador recebe dados do monitor:	**write(newsockfd,buffer,50);**
Simulador envia dados para o monitor	**read(newsockfd,buffer,50);**

CONCLUSÃO

Apesar de não englobar todos os cálculos reais necessários para a solução desse tipo de problema, as funções aqui implementadas simulam bem o funcionamento de um sistema embarcado, tendo como premissa o recebimento, processamento e a saída de dados. Conceitos de periodicidade, variáveis de condição, condição de corrida e comunicação em rede foram amplamente abordados na implementação dos códigos. Sendo assim o presente trabalho mostrou-se uma alternativa simplificada, mas completamente funcional de um sistema em tempo real englobando os principais conceitos teóricos e práticos referente ao tema.