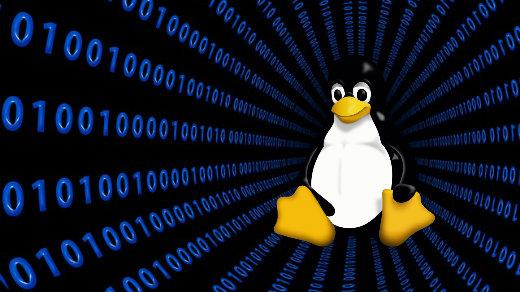


**Faculdade de Ciências Exatas e da Engenharia**

**Sistemas Operativos**

**Projeto Prático, 2019/2020, 1o Semestre**

**Simulação de um Metro**



**Alunos:** João Baptista Coelho Freitas (2035217), David Roberto Cravo da Mata (2035417) e Ricardo David da Silva Briceño (2032917)

**Docentes:** Eduardo Marques e Luís Gaspar

**Introdução / Discussão**

Foi-nos pedido pelos docentes que fosse implementada para a Segunda Fase do Projeto, a comunicação entre as aplicações Simulador e Monitor de modo a poderem interagir e trocar informações e/ou dados das funcionalidades que serão desenvolvidas para a Terceira Fase.

Foi também proposto que fosse fundamentada a encriptação das mensagens e a forma como irão ser tratadas a nível do Monitor e Simulador. Também iremos descrever a forma como vamos abordar a Sincronização das tarefas para que não haja interferência das mesmas.

# **Implementação**

Inicialmente houve problemas em “arrancar” com o projeto, pois o nível de abstração das funcionalidades oferecidas pelos **Sockets** são muitas variadas e não tão claras na hora de implementação. Contudo, conseguiu-se desenvolver a **comunicação entre as aplicações** e permitir a interação entre as mesmas.

O **formato das mensagens** foi definido de uma maneira simples, que vai seguir o seguinte formato: **XN.Y.Z** (Sendo **X** a estrutura de dados e **N** a instância sobre o qual ocorreu um acontecimento, **Y** representa qual o atributo da instância que irá ser afetado, **Z** será um parâmetro adicional caso seja necessário passar mais alguma informação). Consideremos a seguinte estrutura:

struct bilheteira {

bool VIP;

int estacao;

int pessoasFila;

int pessoasVIPFila;

int tempoMedioEspera;

int pessoasDesistentes;

};

A mensagem enviada do simulador para o monitor do tipo “**B1.3.X”** traduz-se em “no aumento do número de pessoas VIP na fila de espera da instância 1, da estrutura de dados bilheteira”. Segue-se na tabela um exemplo do que se pretende.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **X** | **N** | **Y** | **Z** | **Simulador** | **Monitor** | Monitor |
| E | 1 | 2 | X | Chegou uma pessoa à estação 1 | estacao1.numeroPassageirosSemBilhete++ | estacao1.numeroPassageirosSemBilhete++ |
| E | 2 | 2 | X | Chegou uma pessoa à estação 2 | estacao2.numeroPassageirosSemBilhete++ | estacao2.numeroPassageirosSemBilhete++ |
| E | 1 | 3 | 0 | Saiu um passageiro do comboio para a estação 1 | estacao1.numeroPassageirosComBilhete++ | estacao1.numeroPassageirosComBilhete++ |
| E | 2 | 3 | 0 | Saiu um passageiro do comboio para a estação 2 | estacao2.numeroPassageirosComBilhete++ | estacao2.numeroPassageirosComBilhete++ |
| E | 1 | 3 | 1 | Um passageiro comprou um bilhete na estação 1 | estacao1.numeroPassageirosComBilhete++ | estacao1.numeroPassageirosComBilhete++ |
| E | 2 | 3 | 1 | Um passageiro comprou um bilhete na estação 2 | Estacao2.numeroPassageirosComBilhete++ | estacao2.numeroPassageirosComBilhete++ |

Desta forma pretende-se que o Monitor mostre a informação estatística relevante do que está a ocorrer, como por exemplo as pessoas que estão no Metro, nas Estações, na Bilheteira, etc. Por outro lado, no Simulador irá ser mostrado mensagens dos acontecimentos que as threads realizaram de modo a poder haver um melhor controlo da execução.

Tendo sido pensado em grupo, diferentes maneiras de como abordar o Projeto num todo, decidiu-se que deveriam ser implementadas as seguintes **funcionalidades**:

* Função para criar pessoas (indicar se é VIP, estação inicial, o seu destino) e definir o seu comportamento, função esta que será associada a uma thread.
* Função que faz a tradução das mensagens encriptadas e provoca um acontecimento
* Função que controla a entrada no metro
* Função que controla a entrada na estação
* Função que controla a saída no metro
* Função que controla a saída na estação
* Função que controla a compra de bilhetes
* Função que determina os tempos médios de espera nas filas
* Função que controla o tempo da simulação
* Função que controla a viagem em si, isto é, as estações em que o passageiro pode sair (Se sair na estação errada, aguarda por um novo metro, caso contrário, a pessoa sai da estação e a thread é eliminada)

É sabido que estas funcionalidades, apesar de nos fazerem sentido e serem, numa primeira vista, fundamentais para o projeto podem vir a sofrer alterações sendo que algumas delas são capazes de nem serem implementadas ou, num outro ponto de vista, termos falta de funcionalidades fundamentais para o Projeto. Este assunto só irá ser bem fundamentado na Terceira Fase, tendo-se previsto ter o Projeto a funcionar na totalidade.

A grande parte destas funcionalidades têm de ser executadas em **exclusão mútua**, ou seja, iremos optar por implementar **Trincos/Semáforos** em todas as variáveis globais de forma a garantir a **Sincronização**. Um exemplo simples de se perceber é a entrada/saída de passageiros numa carruagem do metro, em que iriamos ter uma variável que incrementasse/decrementasse respetivamente em exclusão mútua pois iria-se obter informações erróneas caso dois ou mais passageiros entrassem/saíssem ao mesmo tempo.

Relativamente ao funcionamento geral, cada pessoa irá ser um thread. Na fila de compra de bilhetes, será implementado uma fila com prioridades, sendo que os utilizadores VIP passam sempre à frente dos utilizadores comuns. Caso haja muitas pessoas, os utilizadores podem desistir da fila de comprar de bilhetes. Após a compra do bilhete, é possível a desistência ou a entrada no metro, que será efetuada por ordem de chegada. De notar que a entrada no metro só pode ser efetuada caso o metro esteja na estação.