
# Sistemas Operativos

## Programação em C para UNIX

2021 - 2022

**Nuno Santos**
**Francisco**
**Mendes**

Licenciatura de Engenharia Informática
14 de janeiro de 2022

# Conteúdo

# 1 Introdução

O trabalho prático consiste na implementação de um sistema para gerir atendimento de clientes em estabelecimentos médicos. O sistema, denominado MEDICALso irá mediar a interação entre cliente, médico e balcão.

O trabalho prático foi concretizado em linguagem C, para plataforma UNIX (Linux), usando os mecanismos do sistema operativo abordados. No que respeita à manipulação de recursos do sistema foi dada prioridade ao uso de chamadas ao sistema operativo face ao uso de funções biblioteca.

# 2 Arquitetura Geral

Neste trabalho optámos por usar o modelo Cliente – Servidor, onde o balcão será considerado o nosso servidor e clientes(programa) e médicos serão considerados clientes (modelo).

Quando executamos o balcão este cria de imediato o seu pipe que servirá para receber pacotes de dado de clientes e de médicos e cria um outro pipe só para estar atento ao sinal de vida do médico, este também corre em background o programa Classificador.

Quando executamos o cliente este também cria de imediato o seu pipe que servirá para receber pacotes do balcão ou do médico. Assim que um cliente chegue á MEDICALso este envia as suas informações ao balcão e recebe do balcão a sua especialidade e prioridade pois foi implementado um mecanismo de comunicação entre o balcão e o Classificador usando pipes anónimos ficando assim aguardar pela ordem do balcão para iniciar a consulta.

Quando executamos o médico á semelhança do cliente este também cria o seu pipe que servirá para receber pacotes do balcão ou do cliente. Assim que um médico chegue á MEDICALso este também envia as suas informações e recebe do balcão uma confirmação de registo. Assim que esteja em runtime um cliente cuja a especialidade atribuída é igual é especialidade do médico é enviado uma “ordem” do balcão para iniciarem a consulta.

# 3 Estruturas de Dados

## 3.1 Estrutura balcao

```
// ESTRUTAR QUE MEDICO E UTENTE ENVIAM PARA O BALCAO

typedef struct {
  char nome_utente[NOME_MAX], nome_medico[NOME_MAX],msg[256];
  char sintoma[SINTOMA_MAX];
  char especialidade[ESPECIALIDADE_MAX];
  char classificao[ESPECIALIDADE_MAX];
  int filas[MAX_FILAS],prioridade,sair,temp;
  pid_t id_utente, id_medico; //PID do processo
  int remove,encerra,flagOcupado,flagN;
  int  cliente; //Saber se e cliente ou medico | 1 -> Cliente | 0 -> Medico
  int cheio; // 1 - livre 0- cheio
  int registo_medico,registo_utente;
  int consulta; // 0 -> não esta  1 -> está
  int flagB; // 1 balcao 0 nao e balcao
  int med_ocupado,cli_ocupado; // 0-> esta | 1-> n esta
  pthread_mutex_t *trinco;  // Partilhado

}balcao;
```

Código 1: Estrutura balcao

Esta é a estrutura principal de comunicação o cliente o médico e o balcão comunicam através dela.

De modo a facilitar o nosso trabalho optámos por usar uma só estrutura para toda a comunicação, no entanto, temos noção de que desta forma temos algumas vantagens como desvantagens.

### 3.2 Estrutura thbalcao

```
typedef struct THbalcao{
   int continua,tempo;
   int maxClientes, maxMedicos;
   balcao p_cli[5];
   balcao p_med[5];
   int ite_cli , ite_med;//"iteradores"
   pthread_mutex_t *trinco;
   int tempo;
}thbalcao;
```

Código 2: Estrutura thbalcao

Esta é a estrutura associada ao "mostraListas". Responsável por mostrar se os clientes ou médicos estão ou não em consultas.

# 4 Balcao

O árbitro é o cérebro da MEDICALso. Este tem a responsabilidade de garantir toda a comunicação.

Primeiramente dizer que optámos por usar Select's não por acharmos que seja propriamente mais fácil mas sim pelo facto da ordem cronológica em que a matéria foi abordada nos laboratórios, as Thread's iriam ficar para muito perto da data de entrega do TP e poderíamos não ter tempo suficiente para consolidar bem essa matéria. No entanto importa referir que temos noção das vantagens e desvantagens quer usássemos o mecanismo Select's quer usássemos o mecanismo Thread's.

Sempre que lançamos o programa balcão são criados dois pipes, um chamado o "server_fifo" que servirá para receber todos os pacotes enviados pelo cliente e médico, outro pipe chamado de "sinal" que estará somente atento ao sinal de vida vindo do médico.

De seguida irá ler as variáveis Ambientes onde irá ficar com um máximo de clientes e máximo de médicos associado, para isso temos o script "varAmbiente.sh" que deve ser lançado da forma ". ./varAmbiente.sh" antes de lançar o balcão se não irá dar erro, depois irá criar um novo processo onde é feita a comunicação por pipes anónimos com o Classificador, responsável por classificar os sintomas com uma especialidade e prioridade. Já no processo pai é criada a Thread "mostraListas" que irá fazer o que acima foi descrito, depois entrará no ciclo "principal" onde começará por preparar o Select e depois irá ficar á espera de receber um pacote no seu pipe, após receber esse pacote irá verificar se ele veio de um cliente ou médico através de uma flag ("cliente") caso seja um cliente irá mandar os sintomas para o Classificador que depois de receber a resposta irá separá-la devidamente e enviar essa informação para o cliente. Logo de seguida irá verificar se tem algum médico com a especialidade igual á especialidade que o Classificador atribuído a esse cliente.

Caso o pacote que receba venha de um médico então irá enviar uma confirmação de registo ao médico, caso corra tudo bem, e depois procurará utentes com uma especialidade igual á especialidade do médico e informará o médico disso.

## 4.1 Clientes

O Cliente é responsável por permitir um utente ter uma consulta na MEDI CALso.

Numa fase inicial este começa por verificar se o balcão existe verificando se consegue abrir o seu pipe. Se efetivamente o balcão não existir este fecha. Se existir, abre dois pipes, o do servidor (balcão) e o dele (cliente).

Se o balcão estiver então aberto e a funcionar , o cliente pede ao utente ou seus sintomas.

Um cliente que pretenda ter uma consulta começa por enviar o seu PID (id de processo) e os seus sintomas e recebe do balcãoa sua especialidade e prioridade. Esta informação é enviada ao balcão pelo pipe deste ("server_fifo"). Depois de enviada essa informação o balcão irá verificar se está ligado algum especialista com a especialidade igual á classificação do cliente, caso esteja o balcão informa o cliente disso e passa á consulta com o especialista, caso não esteja nenhum médico ligado para aquela especialidade então o cliente fica a aguardar que se ligue algum.

De modo a receber a informação o cliente encontra-se num do while loop onde está constantemente a verificar se recebeu informação ou do STDIN ou do seu pipe.

O cliente tem dois estados possíves, o estado 1 em que não está em consulta e todo o stdin irá ser para o balcão e o estado 2 em que está em consulta e todo o stdin irá ser para o médico.

## 4.2 Médico

Um médico sempre que é lançado começa por criar o seu pipe "medico_fifo_%d" em que no %d irá ser substituído pelo PID daquele médico.

O médico tem 3 estados possíveis estado 0 em que não está registado devido a vários fatores como por exemplo o balcão não ter capacidade para mais médicos, ocorrer algum erro entre a comunicação entre médico e balcão, o estado 1 significa que recebeu do balcão uma confirmação de registo e passa assim a estar no estado 1 em que todo o stdin vai para o balcão ficando a aguardar do mesmo uma confirmação para iniciar a consulta por fim o estado 2 que significa que está em consulta e todo o stdin é enviado para o cliente.

Após o lançamento do programa e a criação do seu pipe, este envia o seu PID e especialidade ao balcão de onde recebe uma confirmação de registo (se reunir todas as condições para isso) e passa assim a estar á espera de clientes, se já houver algum cliente pendurado com a mesma classificação o balcão informa o médico e passam a iniciar a consulta propriamente dita, consulta essa que termina a qualquer altura introduzindo a palavra "adeus" e caso isto aconteça o médico informa o balcão que a consulta terminou e passa a estar no estado 1 em que aguarda por mais clientes.

O médico também é responsável por enviar um sinal de vida ao balcão a cada 20 segundos, para isso criou-se uma thread associada ao temporizador que enviará um sinal de vida para o pipe "sinal" do balcão.

## 4.3 Comandos ADMIN

O balcão e, deste modo, a MEDICALso, são controlados pelo ADMIN. Este envia, a partir da linha de comandos, ordens que devem ser executadas pelo balcão. Estas podem ser as seguintes:

- **utentes**: lista os utentes em fila de espera indicando qual a especialidade e qual a prioridade, e também os utentes atualmente a serem atendidos, em que especialidade e por qual especialista;
- **especialiastas:** indica a lista de especialistas conhecidos e estado atual (à espera, a atender o utente X);
- **delut X:** remove o utente em espera X, informando-o através do seu programa cliente (que depois encerra). Esse utente sai do sistema. Válido apenas para utentes que ainda não começaram a ser atendidos;
- **delesp X:** remove o especialista X, informando-o através do seu programa médico (que depois encerra). Válido apenas para especialistas que não estejam a atender nenhum utente no momento;
- **freqN:** passa a apresentar a ocupação das filas de espera de N em N segundos;
- **encerra:** termina o sistema, avisando os clientes, os médicos e o classificador.

# 5 Makefile

O nosso makefile apresenta as seguintes rules:

| Rule | Dependencies | O que faz |
|---|---|---|
| all | Medico cliente balcao | Compila todos os programas associados ao trabalho |
| balcao | balcao.c func_balcao.o utils.h | Compila o programa balcao |
| medico | medico.c utils.h | Compila o programa medico |
| cliente | cliente.c utils.h | Compila o programa cliente |
| func_balcao | func_balcao.o balcao.c utils.h | Compila o objeto func_balcao |
| clean | rm medico balcão cliente func_balcao.o server_fifo cliente_fifo* medico_fifo* sinal | Limpa todos os ficheiros gerados pelo makefile |

# 6 Conclusão

Ao longo deste trabalho, deparámo-nos e fomos resolvendo vários desafios que não esperávamos ter, tratando-se de uma excelente oportunidade para consolidação de matéria das aulas teóricas e práticas de Sistemas Operativos. Este permitiu-nos colocar em prática conceitos importantes sobre a arquitetura, criacão e desenvolvimento de programas para sistemas UNIX, dando-nos uma visão para os vários detalhes dessas atividades.

**NOTA:** O TP infelizmente tem alguns bugs/erros na implementação que foi descrita ao longo do relatório que não conseguimos resolver em tempo útil , esses mesmo bugs serão explicados durante a defesa.

# Pedaços de Código