Instituto Superior de Engenharia de Coimbra

Licenciatura em Eng. Informática

Relatório do Trabalho Prático de SO 21/22
Sistema MEDICALso

Meta 1

Diogo Miguel Pinto Pascoal, 2018019825

Nuno Gabriel Tavares Honório, 2019126457

1. Introdução

O trabalho prático de Sistemas Operativos consiste na implementação de um sistema de gestão de atendimento de clientes em estabelecimentos médicos denominado MEDICALso. Este sistema encarrega-se de encaminhar e mediar a interação entre clientes, médicos e o balcão de atendimento.

Este trabalho é concretizado em linguagem C para a plataforma Unix (Linux), sendo que nesta primeira meta apenas é explicado levemente as estruturas de dados planeadas além da interação balcão - classificador.

2. Estruturas de Dados / Headers

Em vez de inserir todas as estruturas de dados num único ficheiro, foi decidido inserir cada estrutura num header correspondente, sendo que as estruturas de dados associadas ao balcão se encontram no balcao.h, as que estão associadas ao cliente no cliente.h e assim seguindo.

2.1 Balcão

```
struct Balcao{
    int N; //valor maximo de clientes EM SIMULTANEO
    int M; //valor maximo de medicos EM SIMULTANEO

    int nClienteLigados;
    int nMedicosLigados;
     // ortopedia, geral,
     //  3       ,   1  ,
    int filaDeEspera[ESPECIALIDADES];
};
```

Figura 1- Estrutura Balcao

A estrutura Balcao irá conter informação relativa à gestão geral do nosso programa. Guarda o número máximo de Clientes e Especialistas (Médicos) após a leitura das variáveis ambientes MAXCLIENTES e MAXMEDICOS que devem ser definidas anteriormente.

Armazena também duas variáveis inteiras que terão como valor o número de processos abertos tanto de Especialistas como Clientes. O array de inteiros filaDeEspera tem como objetivo guardar o número de clientes em espera para cada uma das especialidades existentes no hospital.

2.2 Cliente

```
typedef struct utente{
    char nomeUtente[MAX_STRING_SIZE];
    char especialidadeAtribuida[MAX_STRING_SIZE];
    int prioridadeAtribuida;
    int desistiu; // morreu ou desistiu do tratamento
    int fila; // se estiver na fila - 1 ; se estiver em tratamento - 0
} Utente, *pUtente;
```

Figura 2 – Estrutura Utente

De momento o cliente / utente é definido por uma estrutura utente que contém dois arrays, um para guardar o nome do mesmo e outra para guardar a especialidade atribuída ao utente, duas *flags* inteiras que permitem saber se o utente desistiu do tratamento ou se este se encontra na fila (variáveis que vão variar entre 0 e 1) e por fim uma variável inteira com a prioridade recebida pelo classificador.

2.3 Medico

```
typedef struct especialista{
    char nomeMedico[MAX_STRING_SIZE];
    char especialidade[MAX_STRING_SIZE];
    int isAlive;

}Especialista, *pEspecialista;
```

Figura 3 - Estrutura Especialista

O Especialista tem como objetivo conter a informação de cada Médico Especialista: dois arrays, um com o nome do médico e outro com a especialidade do mesmo, assim como uma *flag* inteira para saber se o mesmo ainda se encontra "vivo".

2.4 Utils.h

Apesar de não conter nenhuma estrutura, o header Utils.h será utilizado para manter variáveis constantes que são necessários por diversas partes do sistema, evitando assim duplicar estes valores entre diversos ficheiros e headers.

3. Comunicação Balcão – Classificador

Nesta primeira meta foi requerido implementar a parte do balcão relativa à classificação da especialidade e respetiva prioridade. Para o efeito, foi assumido que os dados dos sintomas eram obtidos diretamente pelo utilizador administrador do programa balcão.

Deste modo, após a confirmação de que as variáveis de ambiente necessárias para o balcão poder executar estavam definidas e válidas, são criados 2 arrays de inteiro para conterem os File Descriptors dos pipes que serão criados para comunicar com o Classificador: fd_balcao_classificador e fd_classificador_balcao, um para substituir o stdin e o outro para substituir o stdout do classificador

De seguida são efetivamente criados os pipes; caso a criação de um destes falhe, o balcão irá abortar a execução do sistema.

Por último, é feito um fork para poder executar o classificador à parte. No entanto, antes de invocar a função execl() para executar o classificador, é substituído o stdin e stdout desta thread para poder enviar informação para o classificador sem interagir diretamente com este.

Assim como indicado na documentação do programa Classificador, a execução do mesmo termina ao receber o input "#fim". Quando isto acontecer, o balcão irá enviar o input para o Classificador, terminando assim a execução do Classificador, e irá também interpretar esse input para si mesmo para terminar o loop onde se encontra à espera para receber sintomas e devolver a resposta.

Para evitar problemas de formatação e de leitura de "lixo", a cada ciclo de escrita e leitura nos pipes é realizada uma limpeza aos buffers que obtêm o input e output, assim como ao stdin e stdout.

4. Variáveis de ambiente

Para a execução do sistema e, mais em específico, do balcão, são necessárias duas variáveis de ambiente que têm que já estar definidas antes de executar o balcão. Essas duas variáveis são MAXCLIENTES e MAXMEDICOS que representam respetivamente o número máximo de clientes e médicos / especialistas que podem se encontrar conectados ao mesmo tempo ao servidor.

De acordo com o enunciado, caso alguma destas variáveis não se encontre definida ou apresente um valor inválido, o balcão irá interromper a sua execução, indicando o motivo em stderr. Deste modo, não existem valores por omissão para as variáveis de ambiente.