RELATÓRIO TRABALHO PRÁTICO SISTEMAS OPERATIVOS

Trabalho realizador por:
Rafael Gil
a2020136741 @isec.pt
Pedro Nogueira
a2020136533@isec.pt

Índice

Implementação de Funcionalidades

Este trabalho é constituído por quatro programas distintos, sendo eles o classificador, um programa fornecido pelo professor que fornece informação, neste caso uma classificação composta por uma especialidade e um grau de prioridade, dependendo do *input* que tiver. O balcão, que vai agir como programa principal e intermédio entre todos os outros e que serve como "interface" ao administrador; o cliente, que vai servir de "interface" para o utente e o médico, que vai servir de "interface" para o especialista.

O classificador é o único programa que não tem que comunicar com todos os outros, uma vez que apenas o balcão comunica com ele e ele com o balcão. Sendo assim, foi usado um *named pipe* para proceder à comunicação entre os dois programas, uma vez que, para além de não ser possível aceder ao código-fonte deste, o classificador vai estar "alojado" num processo filho do programa balcão.

Em relação à comunicação entre os outros três programas, foram usados *FIFOS* para assim ser possível partilhar informação entre eles.

Estruturas

Foram usados três estruturas de dados distintas, a estrutura pedido, a estrutura medico e a estrutura balcao, sendo a estrutura medico e balcao usadas exclusivamente pelos programas com os mesmos nomes. Já a estrutura pedido, é usada pelos programas balcão, médico e cliente, sendo a estrutura que vai ser enviada entre os programas, através dos *FIFOS*.

Pedido

Uma vez que é usada por todos os programas, permitindo assim a comunicação entre eles, esta estrutura vai ter que conter dados suficientes para que seja possível ir ao encontro de todos os requisitos necessários de cumprir por cada aspecto da comunicação entre programas. Assim, a estrutura é composta por sete inteiros e quatro strings.

Dois dos inteiros têm o propósito de guardar o PID do cliente e do especialista, pid_cli e pid_med respetivamente, sendo o resto dos inteiros um conjunto de *flags:* "cli_med" que representa se a estrutura foi enviada por um médico ou por um cliente, sendo que se encontra a 0 se for um cliente e a 1 se for um médico; a *flag* "com" que representa se já se encontra numa consulta ou não, contendo o valor 0 se não estiver e 1 caso contrário; "temp" que apenas vai ser usada pelo médico e que indicada o tempo que este se encontrou ser mandar um sinal de vida para o balcão; e por último, a flag "prio" que apenas vais er usada pelo cliente e que vai conter a prioridade que lhe foi atribuída pelo classificador.

Em relação às strings, a "especialidade" apenas vai ser usada pelo médico e vai conter a sua especialidade; a "classificação" apenas vai ser

usada pelo cliente e vai conter a classificação que lhe foi atribuída pelo classificador; a string "sintomas" também vai ser usada apenas pelo cliente e vai conter os sintomas que ele indicou que tinha; por último, a string "msg" vai servir para transportar as mensagens que tanto o cliente como o médico enviarem um ao outro.

```
typedef struct{
    int pid_cli, pid_med, cli_med, com, sair, temp, prio;
    char sintomas[40], classificacao[40], msg[100], especialidade[40];
} pedido;
```

balcao

Esta estrutura tem como função ser transportada dentro do balcão, contendo assim um conjunto de dados essenciais para que tal seja possível, que vai ser o caso do *mutex* trinco, pois vai ser necessário proteger a informação de maneira a que as threads lançadas dentro do balcão não interfiram umas com as outras.

Tem dois arrays de estruturas do tipo pedido, um para os clientes e outro para os médicos, p_cli e p_med respetivamente. Estes arrays têm um tamanho fixo de 5, o que significa que apenas vai poder armazenar 5 estruturas em cada array. Estes arrays vão servir para armazenar a informação de cada programa cliente e médico que se ligar ao balcão, podendo assim fazer uma gestão das filas de espera.

Tem também um conjunto de inteiros, sendo dois deles para guardar a informação do número máximo de clientes e médicos que pode haver, maxclientes e maxmedicos respetivamente. Tem mais dois inteiros que vão servir o propósito de "iteradores" dos arrays, o ite_cli para os clientes e o ite_med para os médicos, uma vez que os arrays têm tamanho fixo e não convém aceder a espaços do array que estejam vazios. Tem também um inteiro que vai guardar o tempo a que vai ser listadas as filas de espera e, por último, tem um inteiro que vai servir de *flag* para as threads saberem quando encerrar.

```
typedef struct{
    int continua;
    int maxclientes, maxmedicos;
    pedido p_cli[5];
    pedido p_med[5];
    int ite_cli , ite_med;
    pthread_mutex_t *trinco;
    int tempo;
}balcao;
```

medico

Esta estrutura, à semelhança da estrutura balcao, vai ter um *mutex* de maneira a proteger a informação devido à thread que é lançada no programa médico.

Esta estrutura tem também uma estrutura do tipo pedido, que vai servir para guardar toda a informação referente ao médico e à ligação com os outros programas

```
typedef struct {
    pedido m;
    pthread_mutex_t *trinco;
} medico;
```

Programas

Balcão

Este programa é o mais complexo dos programas que compõem este projeto, uma vez que usufrui de dois FIFOS, um processo filho e duas threads.

Um dos FIFOS serve para receber informação dos restantes programas, enquanto que o segundo FIFO, apesar de também receber informação dos restantes programas, vai assumir que a função de eliminar o remetente da informação caso este seja um cliente ou vai tratar o sinal de vida caso o remetente seja um médico, ou seja, sempre que for um cliente a enviar informação para este FIFO, é porque se foi embora, então o balcão vai eliminar a informação do cliente das listas de espera; e caso tenha sido um médico a enviar informação por este FIFO, é porque foi um sinal de vida, logo o balcão vai repor o temporizador do médico em questão. É usado um mecanismo select para "tomar conta" de ambos estes FIFOS e ainda do *stdin*.

Quanto ao processo filho, como já foi brevemente explicado em cima, serve para ter a possibilidade de correr o classificador em *background* comunicando entre si através de um *named pipe*.

Em relação às *threads*, uma delas tem a função de fazer a listagem das listas de espera em X em X segundos, sendo este tempo de 20 em 20 segundos, mas que é reconfigurável em *run-time*, havendo um comando do administrador que faz isso. A segunda thread, tem como função apagar os médicos que estiveram X segundos sem mandar um sinal de vida, sendo esta cronometragem feita da seguinte maneira: de 5 em 5 segundos a *thread* vai correr o array que contém a lista dos médicos à procura de algum que tenha a variável do tempo a zero ou inferior. Caso encontre algum, vai proceder a eliminar esse médico do array. Sempre que passa por um médico e que não tenho a variável do tempo a zero ou inferior, a *thread* vai decrementar essa

variável. Esta variável é reposta ao seu valor de origem sempre que um médico envia um sinal de vida.

Médico

Este programa é iniciado ao introduzir o nome e a especialidade do médico, através da linha de comandos.

Este programa é composto por um FIFO e por uma thread, sendo o FIFO usado para receber informação do balcão e do cliente. É feita a distinção entre estes, sendo que o programa assume diferentes comportamentos dependendo de qual deles enviou a informação. Caso tenha sido o cliente a enviar a informação, então tudo o que for introduzido no *stdin* vai ser enviado para o cliente, caso contrário vai ser enviado para o balcão. É usado um mecanismo select para "tomar conta" do FIFO e do *stdin*.

Em relação à *thread*, esta vai ter a funcionalidade de 20 em 20 segundos, mandar um "sinal de vida" ao balcão, escrevendo o sinal no FIFO apropriado do balcão. Se ficar 20 segundos sem enviar um sinal de vida ao balcão, este médico vai ser eliminado das listas do balcão.

Cliente

Este programa é iniciado ao indicar o nome do cliente através da linha de comandos.

Este programa apenas vai fazer uso de um FIFO, que vai servir para receber informação do balcão e do médico. À semelhança do médico, este também vai ter comportamentos diferentes consoante o remetente da informação recebida, sendo que se tiver sido o médico a enviar a informação, o que for introduzido no *stdin* vai ser enviado para o médico remetente. Caso contrário, vai enviar para o balcão.

Quando o cliente é encerrado, vai enviar uma mensagem para o FIFO do balcão, levando o balcão a apagar este cliente da sua lista de espera.

Funcionamento

O programa balcão tem que ser sempre o primeiro a ser lançado, pois os outros dependem dele para funcionar. Assim sendo, sempre que o programa cliente ou o programa médico iniciam, é verificado se o FIFO do balcão existe, simbolizando assim que o balcão já foi inicializado.

Assim que o balcão inicia, cria os FIFOS e dá início ao processo filho, para correr o classificador, e fica à espera de receber informação em qualquer um dos FIFOS ou do *stdin*.

Quando um médico é iniciado, depois de verificar se o balcão já se encontra ligado, procede a criar o seu FIFO e a enviar a sua informação ao balcão, ficando a aguardar uma resposta do balcão, o que vai indicar que

este foi registado nas listas do balcão. Assim que receber a resposta, vai iniciar o select e vai ficar à espera de receber informação em qualquer um dos seus meios.

Quando um cliente é inicializado, após verificar se o balcão está disponível, vai ser criado o seu FIFO e vai ser lançada uma pergunta ao utilizador de maneira a que este introduza os seus sintomas. Após receber esta informação pelo *stdin*, esta vai ser enviada para o balcão e fica à espera de receber informação com a sua classificação do balcão. Assim que receber a resposta, vai iniciar o select e vai ficar à espera de receber informação em qualquer um dos seus meios.

No programa balcão, assim que receber uma ligação no seu FIFO, vai fazer a distinção se é um médico ou se é um cliente, agindo em conformidade com os requerimentos que ambos exigem. Assim, se detetar um cliente, vai verificar se este já existe na lista e caso exista vai saltar em frente ignorando assim a informação que recebeu; caso ainda não exista nas listas, vai percorrer as listas dos médicos à procura de um médico que esteja disponível e que seja da especialidade que ele precisa, se encontrar regista o cliente nas listas e vai enviar a informação do cliente ao médico, fazendo-se passar por um cliente, fazendo assim com que o médica desencadeia o conjunto de ações necessárias à comunicação com o cliente. Caso não encontre nenhum médico que corresponda às necessidades do cliente, então simplesmente vai guardar a informação do cliente na lista, que vai ficar à espera que chegue algum médico que corresponda às suas necessidades. A mesma coisa acontece para o médico, verificando se este já existe na lista e caso não exista vai proceder à ligação entre o médico e o cliente, ao enviar uma mensagem ao cliente, fazendo-se passar por um médico, o que vai despoletar o conjunto de ações necessárias ao cliente para estabelecer ligação com o médico. Caso já esteja registado na lista e esteja a comunicar para este FIFO, quer dizer que o médico está a informar que já acabou a consulta, o que vai levar a que o balcão vá alterar o estado deste médico, o que faz com que o médico fique novamente disponível para atender clientes.

Uma vez em consulta, o cliente e o médico podem ter uma conserva por um tempo indefinido, acabando apenas quando o médico digitar "acabou", o que vai fazer com que o programa cliente com que estava a comunicar encerre, o que faz com que este mande a informar ao balcão de que já encerrou. Após acabar uma consulta, o médico vai mandar uma mensagem ao balcão, a informar que já acabou a consulta e que está novamente disponível para atender clientes.

Conclusão

Concluímos assim que os *pipes*, os *forks,* as threads e os selects são uma peça muito importante na comunicação entre ficheiros executáveis e, com a realização deste projeto, aprendemos a usar e a manipular estas peças vitais à construção de programas.

Aprendemos também a usar as variáveis de ambiente em *Unix* que facilitam a alteração de valores globais.