Sistemas Operativos

Trabalho Prático – Meta 1

Trabalho Realizado por:
Daniel Fernandes (LEI-PL) - a2020116565@isec.pt
Hugo Jorge (LEI-PL) - a2020116988@isec.pt

Introdução

O trabalho prático foi desenvolvido no âmbito da unidade curricular de sistemas operativos. Este consiste na implementação de um sistema de gestão de atendimento de clientes, denominado de MEDICALso, em estabelecimentos médicos.

Este sistema é composto por três processos principais distintos, sendo eles o balcão, o qual só existe uma instância deste processo, e os processos cliente e médico, que poderão existir tantos quantos definidos inicialmente nas variáveis ambiente do sistema Unix linha de comandos, no qual correm estes processos. Existe ainda um quarto processo associado ao funcionamento do sistema, que foi fornecido inicialmente já desenvolvido e que será executado pelo balcão. Este quarto processo trata-se de um programa classificador e tem como objetivo classificar os sintomas apresentados pelos utentes.

No processo do balcão o administrador do sistema poderá intervir a qualquer momento e executar uma série de comandos descritos e tendo também o controlo sobre a informação de todos os utentes e médicos registados no sistema.

1. Estruturas

As estruturas medico e cliente, foram definidas de forma a ser guardada toda a informação relevante, tanto para a identificação, como para a comunicação entre processos.

Ao estarem definidas sob a forma de lista ligada, possibilita desde o início a sua organização pelos critérios definidos como, por exemplo, o grau de urgência ou ordem de "chegada". A remoção de posições individuais é também facilitada pois evita de se movimentar toda a estrutura cada vez que se faz uma alteração à sua constituição.

A lista ligada dos médicos é de struct medico em que cada nó da lista guarda o nome do medico, a especialidade e o namepipe do medico em arrays de caracteres, o PID do medico em um inteiro, e um ponteiro para uma struct cliente que aponta para o nó da lista ligada dos clientes mais precisamente para o utente que está a atender. Caso não esteja a atender este campo está a NULL.

A lista ligada dos clientes é de struct medico em que cada nó da lista guarda o nome do utente, a especialidade e o namepipe do medico em arrays de caracteres, a urgência e o PID em dois inteiros e um ponteiro para uma struct medico que aponta para o nó da lista ligada dos médicos mais precisamente para o médico que está a atender o utente. Caso não esteja a ser atendido este campo está a NULL.

Foi também criada uma estrutura uniformizada de forma que ambos clientes e médicos, consigam comunicar com o balcão da mesma forma, fornecendo-lhe os dados que necessita para manter a interação (struct MSG). Esta estrutura de mensagem contém um array de 256 caracteres para guardar o nome, um array de 256 caracteres para guardar mensagens, um array de 50 caracteres para guardar o nome do pipe de quem envia, um array de 50 caracteres para guardar o nome do pipe para quem há-de começar a enviar mensagem (serve para o cliente receber o name pipe do medico que vai ser atendido e vice-versa) e um inteiro para guardar o pid de quem envia a mensagem.

As estruturas mencionadas anteriormente estão definidas no servico.h onde são importadas para o balcão e serão importadas para os respetivos Clientes e Médicos. No balcaofunc.c estão as funções do balcão e os seus protótipos no balcao.h.

No balcão também são usadas duas estruturas que são enviadas para as threads. A sthread armazena os pipes de comunicação com o classificador, o file descriptor do pipe do balcão, um ponteiro para a variável de saída, um ponteiro para a variável que representa o tempo do alarme, um ponteiro para o início da lista ligada dos médicos, um ponteiro para a lista ligada dos utentes e 3 ponteiros para as variáveis mutex criadas anteriormente. A estrutura structTimes armazena também 3 ponteiros para variáveis mutex, os ponteiros para o início das listas de medicos e utentes, o ponteiro para a variável de saída e ainda guarda um ponteiro para uma estrutura sinal. Esse ponteiro é o início de uma lista ligada de estruturas sinais. Essa lista vai servir armazenar em cada nó o tempo e o PID em que são recebidos os sinais de vida dos médicos.

2. Programa Balcão

2.1. Validações Iniciais do Balcão

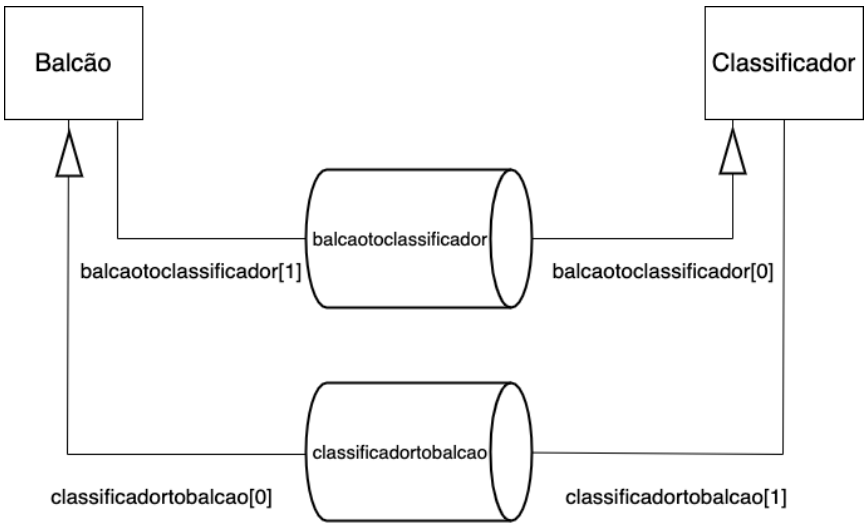
O programa balcão começa obtendo as variáveis de ambiente MAXCLIENTES e MAXMEDICOS. Se uma delas não existir ou se o conteúdo não for um inteiro maior que 0, o programa encerra alertando o administrador que houve um problema com as variáveis de ambiente.

2.2. Named Pipes e Pipes Anónimos

Para o funcionamento do balcão foi criado dois pipes anónimos para comunicação com o programa classificador (balcaotoclassificador e classificadortobalcao).

Também foram criados dois named pipes. O "PIPEBALCAO" usado para comunicação entre o balcão e os clientes e o balcão e os médicos. O "PIPEALERTA" é usado para a comunicação dos sinais de vida por parte dos médicos.

Os nomes dos named pipes estão especificados no ficheiro "serviço.h".



2.3. Funcionamento (main)

Depois de obtidas e validadas as variáveis de ambiente, o balcão tenta criar o pipe do balcão a partir da função "mkfifo". Caso haja algum problema ao criar o pipe verifica-se se é por já existir esse pipe. Sendo assim, significa que existe outro balcão a correr ao mesmo tempo. Algo que não é permitido e sendo assim é terminado o programa.

Após passar todas as validações referidas anteriormente, o balcão faz dois pipes anônimos a partir da função "pipe". De seguida é feito um fork em que no filho é fechado o stdin e colocado na sua posição o balcaotoclassificador[0] e fechado o stdout e

colocado na sua posição o classificadortobalcao[1]. Depois de fechados todos os pipes desnecessários é executado o classificador a partir do execl. Assim o classificador fica sempre a correr.

No processo pai abre-se o pipe do balcão para escrita e leitura (Modo: O_RDWR). Se existir problema ao abrir o pipe, o programa é terminado.

De seguida são criados e inicializadas quatro variáveis mutex, uma estrutura sthread e uma estrutura structTimes que serão passadas por agumento para as 4 threads criadas posteriormente.

No main, após serem criadas as threads, mostra ao administrador os comandos possíveis de inserir e entra num ciclo em que pede ao administrador um comando. Os comandos disponíveis são:

- "utentes" que mostra os utentes em espera e os utentes a serem atendidos e por quem.

- "especialistas" faz o mesmo que o comando anterior, mas para os especialistas.

- "delut" elimina o cliente e envia uma mensagem para terminar. Caso esteja em consulta manda para o medico uma mensagem a dizer adeus que ele interpreta e envia uma mensagem para a thread "ptid" a informar que a consulta terminou e que se encontra disponível. Assim a thread pode atribuir ao medico um novo utente caso exista. (procedimento pormenorizado na secção as threads).

- "delesp" elimina o medico e envia uma mensagem para o terminar e para o cliente que estiver a atender caso exista.

- "encerra" envia mensagem para thread "ptid" com o seu pid que termina as outras threads, os clientes e os médicos, e faz sair do ciclo na função main.

- "freq" modifica o tempo em que é mostrada ao utilizador as listas de espera.

- "sinais" mostra no ecrã a lista dos sinais enviados

Após sair do ciclo do main espera que todas as threads terminam para libertar toda a memória das listas dos clientes e médicos e fazer o unlink do pipe do balcão terminando assim o programa.

A qualquer momento o administrador pode terminar o balcão com as teclas CTRL+C, ou seja, enviando um sinal "SIGINT", este sinal é tratado pela função "trataControlC" em que coloca a variável global de término "sair" com o valor 1 e envia mensagem para terminar para o pipe do balcão, ou seja, para a thread "ptid" que atende os clientes e os médicos. Este procedimento tem um comportamento idêntico ao comando "encerra".

Ao enviar o sinal SIGINT o programa fica preso na leitura do teclado. Para o libertar é fechado o stdin inicialmente. Isto desperta um sinal SIGPIPE pelo sistema operativo que é tratado na função "trataSigPipe".

2.4. THREADS

Foram criadas 4 threads no balcão. A “ptidAlarm”, “ptidVAlerta”, “ptidRAlerta”, “ptid”.

A thread “ptidAlarm” executa a função “alarmPrint” que entra num ciclo que imprime de 30 em 30 segundos as listas de esperas por especialidade. Este valor é alterável pelo administrador.

A thread “ptidRAlerta” chama a função “recebeAlertas” que entra num ciclo em que cria um pipe para receber os sinais de vida dos médicos. Sempre que é recebido um sinal de alerta verifica se já existe esse médico e caso não exista adiciona a uma lista ligada ou se já existir incrementa 20 unidades na variável “time” do seu nó na lista ligada.

A thread “ptidVAlerta” executa a função “verificaAlertas” que entra num ciclo que de 2 em 2 segundos vai decrementar a cada nó da lista ligada dos sinais de vida dos médicos as suas variáveis “time”. Verifica também em cada médico se já excedeu os 20 segundos desde que enviou a última mensagem. Caso isso aconteça, procura-o na lista ligada dos médicos e remove-o. Se estiver em consulta envia uma mensagem ao utente para terminar. Também o remove da lista ligada dos sinais de vida. Caso o especialista tenha terminado por sua vontade, apenas o remove da lista ligada dos sinais de vida. Esta thread antes de terminar liberta a memória da lista ligada dos sinais de vida.

Por fim, a thread “ptid” executa a função “thrd_func” que vai receber do pipe do balcão as mensagens dos clientes e dos médicos. Recebe mensagens de struct MSG. Sempre que recebe uma mensagem, analisa o named pipe de quem enviou e se é um médico ou um cliente, pois o pipe de cada cliente é a junção da abreviatura “cli_” com o pid do processo e o pipe de cada médico é a junção da abreviatura “esp_” com o pid do processo.

Após receber a mensagem e verificar se é de um cliente ou médico, analisa se é um medico/cliente novo e insere na lista de médicos ou de clientes. E verifica se existe um médico ou cliente livre e com a mesma especialidade. Quando isto acontece envia para cada um, o name pipe com quem vai conversar no array de caracteres “pipeaEnviar” de uma struct MSG.

Esta thread quando recebe uma mensagem em que no campo do pid está o seu próprio PID, envia uma mensagem para todos os clientes e médicos para terminarem e termina a thread.

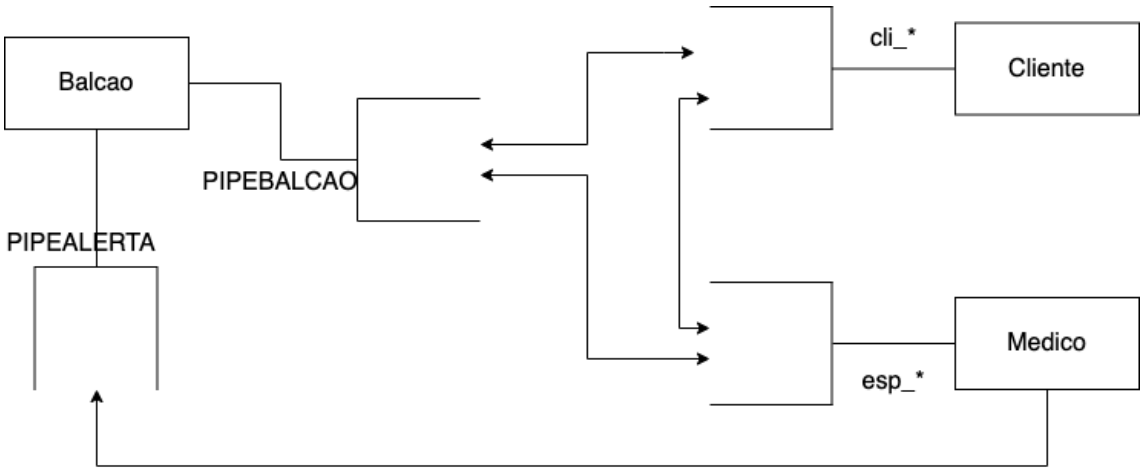
Esta thread quando recebe uma mensagem “sair” de um cliente elimina-o da lista e se for de um médico elimina-o da lista dos médicos. Se estiver em consulta, elimina o utente. O balcão não necessita de avisar o utente, pois o médico já o fez antes de avisar o balcão. Se for uma mensagem “adeus” enviada por parte do médico acontece algo semelhante ao referido anteriormente, porém não elimina o médico e caso exista um utente da sua especialidade, coloca-o a atender esse cliente em espera.

2.5. MUTEX

Para efeitos de sincronização foram criadas 4 variáveis mutex.

A variável mutex “listas” que serve para controlar o acesso às listas ligadas dos médicos e clientes. A “vSinal” serve para controlar o acesso à lista ligada de sinais de vida. A “vSair” para o acesso à variável global de termino(“sair”) de 3 das threads e do ciclo while do main. Estas acedem periodicamente a esta variável.

Por fim, a variável “vAlarme” serve para controlar o acesso à variável da quantidade de tempo em que é impresso as listas de espera dos utentes visto que esta quantidade pode ser modificada pelo administrador.



3. Programa Cliente

3.1. Validações Iniciais do Cliente

Quando o cliente é executado é verificado o número de argumentos. Se o número de argumentos for diferente de dois, o programa termina e imprime no ecrã a sintaxe de execução.

3.2. Named Pipes

No cliente, apenas é usado e criado um named pipe para receber mensagens. O named pipe é criado juntando a abreviatura “cli_” com o pid do processo. E comunica com o balcão a partir do named pipe do Balcão.

3.3. Funcionamento (main)

Após a validação inicial é criado o named pipe do cliente e aberto para escrita e leitura. De seguida é aberto o named pipe do balcão. Se algum dos procedimentos anteriores der erro, o programa é terminado.

De seguida é pedido ao utilizador para inserir os seus sintomas que são enviados para o balcão e fica a aguardar a resposta que indica a especialidade em que foi inserido e a quantidade de utentes à sua frente.

Depois de receber a mensagem do balcão, o main entra num ciclo while em que é programado um select.

O programa está preparado para receber um CTRL+C, ou seja, um sinal SIGINT que é tratado pela função “trataSinalSelect” que envia uma mensagem para com quem está a comunicar.

3.4. Select

No cliente existe um select que fica a aguardar a inserção de informação no teclado ou no named pipe.

Se for detectado informação no teclado é chamada a função “trataTeclado” que lê do teclado e envia para o file descriptor “fd_enviar”. Se for detectado informação no named pipe é chamada a função “leitura2” que lê do named pipe e processa a informação. Esta função lê mensagens de struct MSG.

Se receber uma mensagem com a mensagem “sair” ou “adeus”, termina o programa. Se receber uma mensagem em que na variável “pipeaEnviar” tenha caracteres tenta abrir esse pipe e começa a falar com esse pipe, pois significa que vai ser atendido por um médico.

4. Programa Medico

4.1. Validação da execução do programa médico

O programa médico tem de ser executado fornecendo dois parâmetros, correspondendo o primeiro ao nome do especialista, e o segundo à sua especialidade. Caso não aconteça, o utilizador é informado do erro de execução e é indicada qual a sintaxe correta a adotar.

4.2. Named Pipe

No programa médico, à semelhança do programa cliente, é chamada uma função para criar o named pipe, que irá juntar a abreviatura “esp_” ao pid do processo, para cada processo médico que seja executado. Este named pipe servirá para o médico receber mensagens provenientes tanto do balcão, como do cliente ao qual estará a dar consulta.

4.3. Funcionamento (main)

Após ser feita a verificação de execução, o processo cria o seu named pipe e abre-o no modo de leitura e escrita, por forma a não ficar bloqueado, à espera de receber uma mensagem. É então preenchida a estrutura de mensagem para enviar ao balcão, por forma a identificar a presença do médico, através da abertura do pipe do balcão no modo de escrita. Se, em algum destes procedimentos falhar, ocorrerá uma mensagem de erro e o programa terminará após libertar o seu named pipe criado.

Depois de comunicar a sua informação ao balcão, o médico ficará à espera que lhe seja atribuído um utente compatível com a sua especialidade, iniciando um mecanismo de select (explicado em maior detalhe no ponto 4.4).

Assim que seja atribuído um utente pelo balcão, o processo médico automaticamente ficará em contacto com o cliente do utente atribuído, pois receberá da struct MSG na variável “pipeaEnviar” que deixou de estar a null e passou a conter a informação do pipe do cliente, deixando assim de comunicar diretamente com o balcão. A conversação com o utente é feita de modo a que, tanto o especialista como o utente, possam escrever várias vezes seguidas, sem ter que esperar alguma resposta.

A consulta irá terminar quando algum dos dois intervenientes escrever a palavra “adeus”. Quando tal acontece, o processo médico informa então o balcão através do PIPEBALCAO, quem terminou a consulta e que o especialista está então livre para receber um novo utente.

O especialista poderá também a qualquer momento interromper a sua consulta e terminar o seu processo medico, escrevendo a palavra “sair” que irá desencadear a mudança de uma variável sair, que permitira sair do ciclo while do processo, terminar a sua thread e eliminar o seu named pipe criado.

O processo, tal como os anteriores, está também preparado para receber um CTRL+C, ou seja, um sinal SIGINT, que será processado pela função “trataCntrlC” que executa um procedimento idêntico ao de o especialista escrever a palavra “sair”, permitindo assim que o processo seja terminado corretamente.

4.4. Select

O mecanismo de select criado no programa médico, irá alternar entre a leitura da informação proveniente do teclado do especialista e a leitura do seu named pipe. No caso da primeira, é chamada a função “escreve2”, a qual trata e envia para o cliente o conteúdo escrito no teclado ou para o balcão caso o conteúdo escrito seja “sair” ou “adeus”. No segundo caso, é chamada a função “leitura2”, a qual lê e processa a mensagem recebida no named pipe, quer seja proveniente do cliente ou do balcão.

4.5. Threads

No instante em que o médico está à espera de um utente, é iniciada uma thread que será responsável pelo envio de um sinal ao balcão, por forma a identificar a presença ativa do médico. Este sinal é enviado de 20 em 20 segundos para um named pipe do balcão, específico para este efeito (PIPEALERTA), criado numa thread própria. Caso o processo do médico seja terminado sem que o balcão seja informado, a ausência deste sinal, fará com que o balcão apague o médico em questão da sua lista de médicos, como já explicado anteriormente no processo balcão.