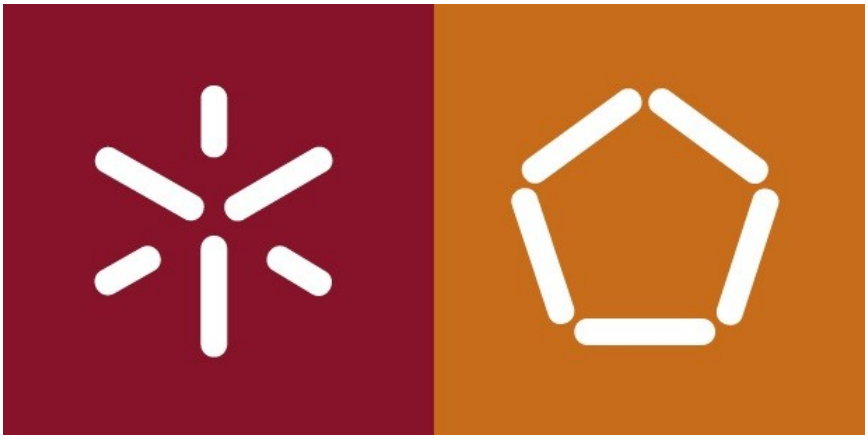

Mestrado Integrado em Telecomunicações e Informatica

2018 / 2019

Sistemas Operativos

Grupo Nº 12

Trabalho realizado

Tiago Pimenta Marques A76869

João Matos Costas A79859

Iago González Soto E8709

INTRODUÇÃO

Para estarmos em conformidade com o que foi pedido no relatório tivemos de elaborar dois programas, o **servidor**, que é suposto ficar em execução no *background*, mandar executar os programas em questão a devido tempo e hora e efetuar todas as outras funções pedidas e o **cliente**, que apenas serve para interação com o utilizador.

SERVIDOR

A primeira coisa que o servidor faz é declarar um array bidirecional chamado Argumentos3, é um array de 100 por 15, suportando assim 100 programas diferentes com 11 argumentos no máximo, cada linha do Argumentos3 possui os argumentos organizados da forma “ID DATA HORA EXECUTÁVEL ARGUMENTOS”, este array é declarado fora das funções para poder ser acedido por todas elas.

Main

De seguida passamos para o main(), o main requer um argumento, que será o nome do ficheiro onde ficará guardada a lista dos programas, abre o ficheiro e um fifo com o nome “*fifo1”*. Chamamos a função CarregaStruct() que, explicando de forma breve, carrega os dados do ficheiro para o Argumentos3 e retorna o número do ID seguinte. Achamos necessário fazer isso agora para retomar uma sessão anterior, caso ela exista.
Começa agora um *loop* infinito gerado por um while(1). Neste ciclo acontecem maioritariamente duas coisas: o servidor verifica se a hora e a data são as corretas para proceder à execução de um determinado programa e se o cliente “pediu” alguma coisa.
Para verificar se o cliente enviou algo, ele tenta ler o fifo na condição de um ciclo while e, se isto for verdade, preenche um array unidirecional com os campos separados por espaços usando a função *strtok_r()*.
Compara a primeira posição do array com “-l” “-c” ou “-a”usando a função *strcmp*, se não for igual a nada disto significa que o argumento não era valido e isso é apresentado na consola. Se alguma das comparações se mostrar verdadeira o programa segue para esse comando, no caso do listar(-l) e do cancelar(-c) chama funções próprias criadas por nós que serão explicadas em detalhe mais à frente no relatório, no caso do adicionar(-a) adiciona diretamente. Para adicionar abre o ficheiro com a opção de escrita e escreve nele os conteúdos de Argumentos(menos a instrução do comando a executar) precedidos do id e envia esse id para o cliente apresentar escrevendo-o no fifo2, criado também por ele, concluindo, chama a função CarregaStruct().

CarregaStruct

A função CarregaStruct() serve para carregar o array bidirecional Argumentos3 com os conteúdos do ficheiro, preenchendo-o assim, com os programas que foram agendados e ainda não foram executados juntamente com a sua data e a sua hora.
Faz isto apagando todo o conteúdo já presente no Argumentos3, de seguida utilizando lerficheiro(), que preenche um buffer que recebe como argumento com os conteúdos do ficheiro que também recebe como argumento, de seguida usa a função strtok_r para dividir o buffer por “\n” e outra vez para dividir os buffers obtidos por espaços obtendo assim cada campo do ficheiro e preenchendo o Argumentos3 com esses campos. Se depois deste processo a primeira posição do Argumentos3 contiver NULL significa que o ficheiro não tinha quaisquer dados e essa mensagem é mostrada na consola.
Além de tudo isto, a função também retorna o próximo número do ID a inserir.

CompararDataHora

Como já foi mencionado anteriormente, no main() o servidor está infinitamente a verificar se a hora e data atual corresponde a qualquer hora e data dos executáveis no ficheiro, para isto usamos a função compararDataHora()
Esta função começa por obter os valores atuais de data e hora e guardar em variáveis, seguidamente começa a percorrer o Argumentos3 linha a linha e para cada linha traduz a data e hora, que estão em formato *string*, para o formato *int.* Isto tem de ser feito para a comparação ser entre dois inteiros, se comparássemos strings e a data atual tivesse um zero à esquerda a comparação iria dar sempre falso.
Caso a comparação tenha sucesso, a função retorna o número da linha do programa a executar guardada no Argumentos3, senão, retorna -1.

Exectuar

Caso a função compararDataHora() retorne um valor diferente de -1, significa que temos um programa para executar neste momento e chama-se a função executar().
A função executar faz um fork() e o processo filho executa o programa usando o exevp() tendo como argumentos o nome do executável e um apontador com os argumentos necessários para esse executável. Tanto o nome como os argumentos tem de estar num Array unidirecional novo chamado Argumentos4 porque, no processo de executar um programa, o Argumentos3 é alterado para já não conter a linha correspondente a esse programa.
Além de executar, também chama a função executado().

Executado

A função executado() serve para escrever a linha do programa executado num ficheiro novo chamado “executados.txt” e para apagar essa linha do ficheiro que contém os programas agendados. Para apagar a linha chama a função cancelarEvento() e fornece-lhe como argumentos o nome do ficheiro e o id da linha a apagar.

CancelarEvento

A função cancelarEvento() serve para apagar uma linha especifica do ficheiro, linha correspondente ao id que recebe como argumento.
Isto acontece percorrendo o ficheiro existente, copiando todas as linhas menos a que contem o id fornecido e escrevendo-as num ficheiro novo chamado “aux.txt”.
De seguida apaga-se o ficheiro antigo e renomeia-se o “aux.txt” com o nome do ficheiro antigo, fazendo assim uma operação de *overwrite*
Caso o id não exista a função escreve isso na consola.
Esta função também pode ser chamada se o cliente pedir.

Listar

Também temos a função listar(), função esta serve para enviar de volta para o cliente uma lista dos programas agendados e dos executados usando o fifo2.
Para os agendados a função percorre o Argumentos3 e escreve-os no fifo2.
Para os executados a função usa lerficheiros() para verificar se o ficheiro “executados.txt” existe, caso exista a função lerficheiros() carregou os seus conteúdos para o buffer bufE e esse buffer é escrito no fifo2.
Finalmente a função fecha o fifo2.

Cliente

O programa cliente é responsável por receber as instruções e os argumentos dados pelo o utilizador e escrevê-los no fifo1. Este programa também garante que apenas argumentos válidos sejam transmitidos para o fifo1 e futuramente para o servidor, se tal não acontecer, o utilizador recebe uma mensagem de texto na linha de comandos, de como introduzir os argumentos, e não escreve no fifo1 até receber argumentos válidos.
Sempre que o utilizador agenda um programa para ser executado ou seja ”-a”, ele recebe no seu terminal o ID associado. Esse ID é diferente para todos os tipos de programas agendados, e é gerado no servidor e posteriormente recebido no programa Cliente utilizando o um fifo chamado “fifo2”.
O programa cliente também utiliza o “fifo2” para receber a lista de todos os programas agendados com o seu respetivo ID e os programas que já foram realizados, quando recebe como primeiro argumento o “-l”. Uma outra funcionalidade do cliente é cancelar um agendamento, para tal o utilizador tem de escrever “-c” e indicar o ID do programa a cancelar.