Universidade do Minho

Mestrado Integrado em Engenharia Informática

2º ANO, 2º Semestre, 2019/2020

Sistemas Operativos

Controlo e Monitorização de Processos e Comunicação

Ana Pereira A81712 | Adriana Meireles A82582 | Shahzod Yusupov A82617

Grupo 47

June 14, 2020

Índice

1 Introdução

No âmbito da Unidade Curricular de Sistemas Operativos foi nos proposto pelos docentes a implementação de um serviço de monitorização de execução e de comunicação entre processos. Para tal foi necessário implementar um cliente e um servidor.

Foi essencial para a execução deste projeto o conhecimento adquirido ao longo do semestre nas aulas práticas desta UC bem como a utilização dos manuais das chamadas ao sistema (*system calls*).

Neste relatório será descrito claramente o raciocínio e as estratégias utilizadas para a implementação das várias funcionalidades do projeto.

2 Implementação

2.1 Servidor

O Servidor comunica com o Cliente através de FIFOs. Para tal são criados 2 pipes com nome: `pipe1` e `pipe2`. Estes são abertos pelo Servidor para leitura e escrita, respetivamente. Assim, o Servidor lê o pedido que o Cliente escreve no `pipe1` e escreve a resposta a esse pedido no `pipe2`.

O Servidor do serviço implementado permite que o Cliente envie tarefas para o mesmo executar. Assim, este guarda o estado dessas tarefas, enquanto estão em execução e após terem terminado. Possibilitando a apresentação da listagem das tarefas bem como a sua terminação por ordem do Cliente.

2.2 Cliente

O Cliente pode comunicar com o servidor de 2 maneiras: pela linha de comandos ou através do **stdin**. É efetuado o *parsing* do comando pelo cliente, sendo de seguida escrito o mesmo no `pipe1`. A funcionalidade referente à ajuda ao utilizador, não é comunicada ao servidor, sendo efetuada pelo Cliente.

Posteriormente, após comunicar ao servidor a instrução, este lê do `pipe2` a resposta do servidor e apresenta-a no **stdout**.

2.3 Estrutura

Dado que este serviço se baseia na submissão e execução de tarefas, definimos uma pequena estrutura no ficheiro **def.h** que representa uma tarefa.

```
typedef struct{
    int num;
    char * commands;
    int pid;
}*Task;
```

O `num` refere-se ao número da tarefa, o `commands` consiste nos comandos a executar e o `pid` indica o *id* do processo que efetua a tarefa.

2.4 Auxiliares

De modo ao ficheiro do servidor não ficar "muito pesado" foi criado o ficheiro **aux.c** que contém funções auxiliares utilizadas. Tais como a `readLine` e a função que executa uma série de comandos separados por pipes anónimos, já com os respetivos redirecionamentos.

2.5 Funcionalidades

O nosso serviço suporta as seguintes funcionalidades:

- **Tempo máximo de Execução**

  Esta opção permite que o utilizador defina um tempo máximo para a execução de uma tarefa. Se esta opção for usada, o valor máximo é guardado numa variável global e em todas as tarefas posteriormente submetidas é efetuado um `alarm(maxExecucao)` antes de começar a execução dos comandos da tarefa.

  Caso o processo ainda esteja a executar aquando da receção do alarme, na função handler do mesmo, o processo termina, chamando o *_exit*, e escreve para o histórico.

  Funcionalidade disponível através dos comandos: `tempo-execucao segs` ou `-m segs`.

- **Executar uma Tarefa**

  Uma tarefa consiste numa sequência de comandos separados por pipes anónimos. Quando uma tarefa é submetida é lhe atribuído um número, retornando o mesmo ao Cliente. De seguida, após a criação da nova **Task**, adiciona-se a tarefa ao array global `currentTasks` que guarda as tarefas em execução. É chamada a função auxiliar, `mysystem` que faz *parse* dos comandos e os executa.

  Ao terminar a execução da tarefa, escreve-se para o histórico. Quando o processo filho termina, o processo pai recebe o sinal **SIGCHLD**, sendo este sinal tratado num handler. Neste, é efetuado *wait* e, tendo o *pid* do processo que terminou, é removido a tarefa do array das tarefas em execução com o *pid* correspondente.

  Funcionalidade disponível através dos comandos: `executar "c1|...|cn"` ou `-e "c1|...|cn"`.

- **Listar tarefas em Execução**

  Para apresentar a lista de tarefas em execução num dado momento ao Cliente é necessário consultar o array das tarefas em execução e escrever a informação num buffer que por sua vez é escrito no FIFO de onde o Cliente irá ler.

  Funcionalidade disponível através dos comandos: `listar` ou `-l`.

- **Terminar uma Tarefa**

  Uma tarefa em execução pode ser terminada forçadamente, se assim o Cliente desejar. Para tal, busca-se a tarefa em execução que se pretende terminar e, através da *system call kill*, envia-se o sinal **SIGKILL** ao *pid* responsável por executar a tarefa. Esta operação também é registada no histórico, com a indicação de que foi a pedido do cliente.

  Funcionalidade disponível através dos comandos: `terminar numTarefa` ou `-t numTarefa`.

- **Listar Histórico de Tarefas**

  Como referido previamente, quando uma tarefa termina é guardada a sua informação no histórico. Para guardar este histórico foi criado um ficheiro **historico.txt**, onde é colocada cada tarefa terminada e a razão de término da mesma.

Cada entrada do histórico tem o formato `#numTarefa, razãoTérmino : comandos`. Existem 3 opções para a razão que terminou a tarefa: `concluida`, `terminada por cliente` ou `max execução`.

Assim, para consultar o histórico é apenas necessário ler do ficheiro e escrever os seus conteúdos no FIFO que o Cliente irá ler.

Funcionalidade disponível através dos comandos: `historico` ou `-r`.

- **Apresentar Ajuda**

  Para apresentar o Menu de Ajuda ao Cliente foi criada a função `helpGuide` que mostra as opções de comandos disponíveis no nosso serviço.

  Funcionalidade disponível através dos comandos: `ajuda` ou `-h`.

- **Consultar Standard Output de uma Tarefa**

  De modo a ser possível a consulta do *output* de uma tarefa já concluída, foi adicionado na função `mysystem` um redirecionamento do **stdout** para o ficheiro **log** no último comando da tarefa. Também foi guardado no ficheiro **log.idx**, o número da tarefa e a posição do ficheiro **log** onde começa o *output* da mesma.

  Então, quando o Cliente deseja consultar o *output*, na função `logs` acedemos ao ficheiro **log.idx** e buscamos a posição de início e fim do *output* no ficheiro **log**. É reposicionado o *offset* do ficheiro para a posição de início e, de seguida, lemos linha a linha até à posição final, guardando o *output* e escrevendo-o, posteriormente, no FIFO.

  Funcionalidade disponível através dos comandos: `output numTarefa` ou `-o numTarefa`.

Foram implementadas todas as funcionalidades propostas exceto a referente ao tempo máximo de inatividade de comunicação entre os pipes anónimos. O Cliente tem disponível a opção que define o tempo de inatividade, no entanto esse valor não está a ser utilizado.

3 Makefile

Para correr o nosso serviço foi criada uma pequena Makefile, abaixo apresentada. Esta trata de compilar o Cliente e o Servidor, gerando os executáveis *argus* e *argusd* respetivamente. O comando **make clean** remove os executáveis, os dois FIFOs utilizados, bem como os ficheiros referentes ao histórico e ao output das tarefas.

```
CC=gcc

all: def.c aux.c
    $(CC) -Wall -o argusd argusd.c
    $(CC) -Wall -o argus argus.c

clean:
    rm -f argus argusd historico.txt fifo1 fifo2 log log.idx
```

4 Conclusão

A realização deste trabalho permitiu, de certa forma, consolidar e, ao mesmo tempo, pôr em prática os conteúdos lecionados ao longo do semestre.

Apesar de termos conseguido implementar a maioria das funcionalidades, o grupo sentiu dificuldade em alguns aspetos como, por exemplo, onde colocar a remoção de uma tarefa do array global aquando do término de uma tarefa em execução, dado que o processo pai não espera que o filho termine a tarefa. A principal dificuldade surgiu na tentativa de implementação da funcionalidade "tempo de inatividade" e, apesar de algumas tentativas, esta não foi conseguida a tempo, ficando assim para reflexão e talvez para trabalho futuro.

Assim, de um modo geral, os objetivos foram cumpridos com sucesso, estando o grupo satisfeito com os resultados obtidos e pronto para aplicar o conhecimento adquirido em trabalhos futuros.