Projeto Prático

Sistemas Operativos

José Ferreira A96798

José Loureiro A96467

Projeto desenvolvido para a Licenciatura em Ciências da Computação

Universidade do Minho

Maio 2024

1 Introdução

O presente relatório refere-se ao trabalho prático proposto na unidade curricular de Sistemas Operativos na Licenciatura em Ciências da Computação Escola de Ciências da Universidade do Minho no ano de 2024.

O objetivo do trabalho consiste no desenvolvimento de uma aplicação cliente/servidor cujo intuito consiste em processar comandos de “bash” e consultar os programas em execução, o servidor é responsável por armazenar as informações e o estado de programas terminados.

A seguir, são expostas as propostas tomadas pelo grupo, onde é explicada a implementação de forma breve, aprofundamento apenas as partes que o grupo considera as mais relevantes para a compreensão do funcionamento do software desenvolvido.

2 Arquitetura

Como mencionado anteriormente, a aplicação segue o modelo Cliente/Servidor, pelo que existem dois programas a desenhar: o Cliente, responsável por comunicar Tarefas ao Servidor, que é responsável por executar comandos de “bash” e consultar os programas em execução enviado o resultado ao Cliente.

2.1 Servidor

Servidor é responsável por receber pedidos efetuados pelos clientes (execute ou status) guardá-los num ficheiro “Tarefas” com o correspondente pedido, argumentos, tempo de execução e processá-los de forma que execute cada pedido corresponde ao ID da tarefa fornecida pelo cliente ao servidor, mandando para o standard output do servidor a informação resultante desse mesmo processo com o seu tempo de execução e ID incluído da tarefa realizada

2.2 Cliente

O cliente é um processo independente do servidor - aliás é possível vários clientes estarem ligados ao servidor simultaneamente. O cliente deve receber o seu pedido através dos seus argumentos, processá-los para o formato utilizado pelo servidor, criar um canal de comunicação (FIFO) para o qual o servidor lê desse mesmo canal até o pedido ser terminado, imprimindo a resposta do cliente e do servidor para o standard output.

3 Implementação

Para facilitar o processo de comunicação deste projeto entre o nosso Cliente e Servidor, decidimos adotar a simplicidade de uma estrutura de dados capaz de salientar as nossas necessidades e de forma simples fazer a separação da informação recebida pelo Servidor do nosso Cliente, com isto, fizemos uma separação logica da nossa estrutura conforme as imposições apresentadas. Assim, dividimos a nossa estrutura em 3 partes, “OngoingTask”, “FinishedTask” e “TaskQueue” que serão brevemente apresentadas

3.1 Estrutura de dados

Macros “Client” e “Server” para facilitar o processo de identificação de abertura de FIFOS usados durante a comunicação do Cliente/Servidor.

```c
#define SERVER "fifo_server"
#define CLIENT "fifo_client"
```

Estrutura de dados para as Tarefas em execução no nosso programa quanto este comunica do Cliente para o Servidor.

```c
typedef struct OngoingTask{
    int type;                   // 0 -> exec, 1 -> status, 2 -> stop
    int time;                   // Time -> tempo fornecido pelo cliente
    int pid;                    // pid do processo a executar
    char prog[20];              // programa de bash a executar
    char args[300];             // argumentos do nosso programa de bash
    int argsSize;               // comprimento em inteiro dos argumentos do programa de bash
    char taskID[20];            // ID do nosso programa
    time_t start_time;          // tempo real do começo da execução do nosso programa
    time_t finish_time;         // tempo real do final da execução do nosso programa
    struct FinishedTask *next;  // indicar a execução proxima tarefa
} OngoingTask;
```

Estrutura de dados para as Tarefas já finalizadas no nosso programa quanto este comunica a finalização da execução do pedido efetuado pelo cliente.

```c
typedef struct FinishedTask{
    int pid;                    // pid do processo finalizado
    char taskID[20];            // ID da Tarefa finalizada
    char prog[20];              // programa de bash efetuado nesta tarefa
    time_t exec_time;           // calculo do tempo real da execução desta tarefa
} FinishedTask;
```

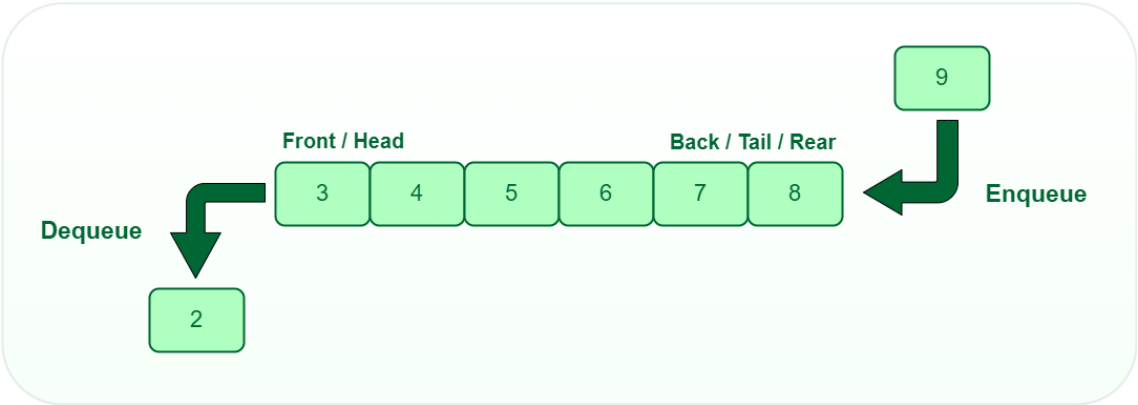
Estrutura de dados utilizada para execução do escalonamento de cada Tarefa escolhida pelo grupo, que será explicada a seguir.

```c
typedef struct TaskQueue{
    FinishedTask *front;
    FinishedTask *rear;
} TaskQueue;
```

3.2 Escalonamento

Das diversas políticas de escalonamento de execução de Tarefas existentes para este tipo de projetos, optamos por utilizar *first come first served*, sempre que o Servidor recebe uma tarefa por parte do Cliente, este guarda-a numa queue. Quando é passado o comando *done* o servidor executa todas as tarefas que estão na queue da primeira adicionada até à última. Para nos ajudar neste processo decidimos utilizar como estrutura de dados uma *Queue* (fila).

As tarefas em execução são guardadas na queue e depois de executadas são registadas, de forma persistente, num ficheiro.



4 Funcionalidades

4.1 Execute

Processo principal de execução de Tarefas em “bash” efetuadas pelo Cliente, com a possibilidade da sua utilização com diversos argumentos:

1. time: indicação do tempo estimado, em milissegundos, para a execução da tarefa;

2. “-u”: indicação que apenas vai executar um programa;

3. prog-a: o nome/caminho do programa a executar (p.ex., cat, grep, wc);

4. [args]: os argumentos do programa, caso existam (p.ex., arg-1 (...) arg-n).

Exemplo de pedido efetuado pelo Cliente com tempo 10 com flag “-u” do programa de “bash” ls -l

```
p@josep-VivoBook-ASUSLaptop-X571LI-F571LI:~/Desktop/2Ano/SO/Projeto/Sistemas-Operativos$ make
-Wall -g  -c src/orchestrator.c -o obj/orchestrator.o
-Wall -g  obj/orchestrator.o -o bin/orchestrator
-Wall -g  -c src/client.c -o obj/client.o
-Wall -g  obj/client.o -o bin/client
p@josep-VivoBook-ASUSLaptop-X571LI-F571LI:~/Desktop/2Ano/SO/Projeto/Sistemas-Operativos$ ./bin/client execute 10 -u ls -l
p@josep-VivoBook-ASUSLaptop-X571LI-F571LI:~/Desktop/2Ano/SO/Projeto/Sistemas-Operativos$ ./bin/client done
```

Exemplo da resposta efetuada pelo Servidor desta tarefa pelo Cliente

```
-Operativos$ ./bin/orchestrator


TASK 1 Received

Task ID: T1
PID: 23954
Program: ls


FINISHED TASK:
Task ID: T1
PID: 23954
Execution Time: 1 ms
```

4.2 Status

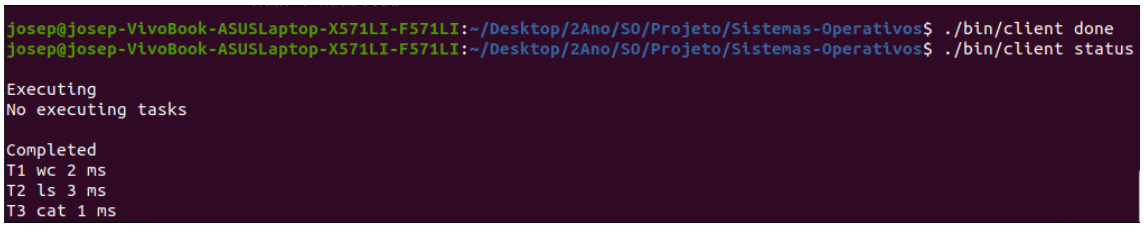
O Cliente permite aos utilizadores, via linha de comando, através da opção status, listar as tarefas em execução e as tarefas terminadas.

O processamento para dar resposta à interrogação anterior é ser realizado pelo servidor. O cliente apenas envia o pedido e recebe a resposta, apresentando-a ao utilizador.

No caso das tarefas em execução, ou à espera para executar, são listados os respetivos identificadores e programas. Para as tarefas terminadas, é imprimida informação relativamente aos seus identificadores, programas, e tempo de execução.

O Servidor não bloqueia Clientes que realizarem pedidos de execute devido a um status a ser atendido por outro Cliente.

Exemplo de um pedido de status efetuado pelo Cliente



5 Conclusão

Deste modo, e para concluir, o grupo considera ter cumprido todos os requisitos impostos relativamente as funcionalidades básicas propostas pelos docentes, implementando uma aplicação cliente/servidor de interação de processos através do paradigma da comunicação por FIFOS.

Como em qualquer projeto, existem aspetos que podiam ter sido melhorados. Um desses aspetos seria a implementação de processamento de várias tarefas em paralelo. Outro aspeto seria a melhoria da função status que não se encontra 100% funcional.

Assim sendo, o grupo considera o seu projeto como tendo sido bem-sucedido relativamente aos requisitos básicos.