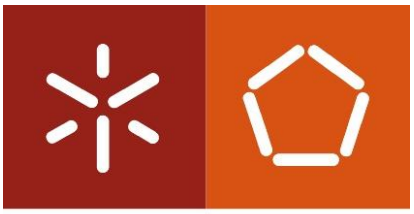

Universidade do Minho

Departamento de Informática

Controlo e Monitorização de Processos e Comunicação

Grupo Nº 69

Carlos Ferreira (A89509)

José Alves (A89563)

Luis Araújo (A86772)

15 de Junho de 2020

Conteúdo

Capítulo 1

Introdução

Foi nos proposto neste semestre a construção de um sistema de *Controlo e Monitorização de Processos e Comunicação*. Para tal, teriamos de produzir uma comunicação entre um cliente e um servidor, armazenar informação, nao só dos comandos executados pelo cliente, como também e os seus *outputs* e por fim controlar o tempo de execução de um processo.
Nos próximos capítulos deste relatório, iremos explicar com mais detalhe o problema proposto pelos docentes, as estratégias utilizadas para a resolução, as dificuldades do mesmo, e por fim uma breve conclusão do projeto.

Capítulo 2

Problema proposto

Como já foi referido anteriormente, foi nos proposto a implementação de um sistema *Controlo e Monitorização de Processos e Comunicação*. Para tal, foi nos indicado que teriamos de elaborar duas formas diferentes de comunicação com o utilizador, uma através da linha de comandos, e outro através de uma interface textual interpretada *(shell)*.Por fim, tivemos ainda de desenvolver algumas funcionalidades:

- **tempo-inactividade** [**-i**] tempo máximo de inactividade de comunicação num pipe anónimo
- **tempo-execucao** [**-m**] tempo máximo de execução de um tarefa
- **executar** [**-e**] executar uma tarefa
- **listar** [**-l**] listar as tarefas em execução
- **terminar** [**-t**] terminar uma tarefa em execução
- **historico** [**-r**] histórico de tarefas terminadas
- **ajuda** [**-h**] linhas de comandas da utilizaçao do sistema
- **output** [**-o**] *standard outputs* produzido por uma tarefa já executada

Capítulo 3

Resolução do problema

3.1 Argus

Na parte do cliente, começamos primeiro por interpretar o comando inserido, pela validação do mesmo e de seguida enviamos esse comando para execução no servidor. Para interpretação do comando, criamos uma função a qual analisa uma linha introduzida pelo cliente, e logo de seguida a esta linha ser captada, esta entra numa outra função que valida a mesma. Após este processo, enviamos a informação necessaria para o servidor através de um fifo.

3.2 Argusd

Na parte do servidor, este recebe a informação contida no fifo, e procede a execução do comando. Para que seja possivel conter um histórico e uma listagem dos comandos executados no servidor, optamos por construir uma estrutura que se comporta igual a um arraylist (*tasklist*), guardando neste (*task*) informações como, o estado do comando, (running, concluded, max_inactivity, max_execution, terminated) os pids que executaram ou estão a executar o comando, o número dos comandos, e por fim o comando em string.

3.3 Output

Para que a parte do output seja bem sucedida e com um bom tempo de execução, fizemos um sistema que contivesse um ficheiro com as respostas do comandos executados ("log") e outro com as informações necessárias para ir buscar um dado comando ("log.idx"), tal como foi indicado pelos docentes. À medida que os output's eram gerados pelos comandos exec, o output destes eram redirecionados para o input do ficheiro que contem as respostas, e para posteriormente fosse possivel buscar essas respostas, guardamos o local onde este foi escrito no ficheiro e o seu tamanho, numa estrutura (*logidx*), o qual é escrito num ficheiro diferente "log.idx".

Capítulo 4

Estruturas

```
typedef enum {running, concluded, max_inactivity, max_execution, terminated} STATUS;
typedef enum {false, true} bool;


typedef struct logidx {
    int offset;
    int size;
} LOGIDX;


typedef struct task {
    int ncmd;
    pid_t pid[64];
    STATUS status;
    char *task;
} TASK;


typedef struct tasklist {
    int used;
    TASK list[4096];
} TASKLIST;


typedef struct readln_buffer {
    int fd;
    ssize_t mem_size;
    char *line;
} ReadlnBuffer;


typedef struct parsed_line {
    char opt;
    char arg[4096];
} ParsedLine;
```

Capítulo 5

Conclusão

Ao longo do desenvolvimento do trabalho, o grupo deparou-se com várias decisões em termos de implementaçao e algumas dificuldades, sendo estas a utilização de memória dinâmica, utilização de sinais para que fosse possivel a ocorrência de tempo-execucao e tempo-inatividade. Apesar de não termos conseguido a introdução destes comandos, devido a não experiência de comunicação entre filho-pai, concluímos que o resto do projeto foi bem sucedido.