Universidade do Minho

Licenciatura em Ciências da Computação

Sistemas Operativos - Trabalho Prático

Grupo 17

Relatório

Inês Pires Presa (A90355)

Ivo Miguel Gomes Lima (A90214)

Tiago dos Santos Silva Peixoto Carriço (A91695)

16 de junho de 2021

Conteúdo

Capítulo 1

Introdução

Foi-nos proposto, no âmbito da unidade curricular de Sistemas Operativos, o desenvolvimento de um sistema capaz de transformar vários ficheiros de áudio concurrentemente, consultar as tarefas em execução, mostrando ainda o número de filtros disponiveis e em uso nessas transformações. Este tipo de sistema utiliza um conjunto de executáveis utilizados para filtrar e modificar um fluxo de áudios, permitindo tratar grandes quantidades de dados explorando a concorrência entre os diferentes processos.

Neste relatório,vamos explicitar a nossa abordagem ao problema, justificando a estrutura do nosso sistema, demonstrando os conhecimentos adquiridos durante as aulas, tais como a utilização/criação de processos, duplicação de descritores, criação de *pipes* sem/com nome, execução de processos, aplicação de sinais e *system calls*.

Capítulo 2

Funcionalidades

2.1 Aplicação de Filtros

Submetendo um comando do tipo *./aurras transform*, seguido dos nomes do ficheiro original assim como do ficheiro processado e de uma sequência de identificadores de filtros, o programa aplica-os ao primeiro ficheiro, guardando o resultado final no segundo. Esta ação só não será efetuada caso seja pedido que algum dos filtros seja aplicado um número de vezes superior ao máximo definido no ficheiro de configuração. Quando esse valor é alcançado devido à pré-existência de pedidos, este fica em lista de espera para que possa ser posteriormente processado.

2.2 Verifição do Estado

Caso o comando seja do tipo *./aurras status*, o cliente recebe do servidor uma descrição dos pedidos que estão a ser processados, assim como uma lista dos filtros em utilização e a identificação do *pid* do servidor.

2.3 Informação de Utilização

Na eventualidade de o utilizador pretender obter informação relativa às funcionalidades suportadas pelo cliente (servidor), este terá de digitar *./aurras* (*./aurrasd*).

Capítulo 3

Estrutura de implementação

3.1 Cliente (*aurras.c*)

Para a implementação das funcionalidades referidas anteriormente, foi necessário a criação de duas funções *status* e *transform* que serão brevemente descritas de seguida.

- **Status:** São criados dois *pipes* com nome que fazem a ligação com o servidor. O *pipe_escrever* é empregue para o envio do pedido de verificação do estado conduzido pelo cliente, em contrapartida o *pipe_ler* recebe a informação solicitada que será imprimida no *standard output*. O *pid* do cliente é enviado através do *pipePrincipal* para o servidor.
- **Transform:** Nesta função só é criado um *pipe_escrever*, uma vez que o método de comunicação que definimos no sentido servidor ->cliente é a utilização de sinais. Desta forma, o *pipe* é utilizado para enviar a informação referente ao pedido de transformação do ficheiro, quando o processo é iniciado o cliente recebe um sinal (*SIGUSR1*).

3.2 Servidor (*aurrasd.c*)

3.2.1 Variáveis Globais

- **struct lligada:** Esta *struct* serve para armazenar as informações presentes no ficheiro de configuração, nomeadamente o nome do filtro, o nome do executável e o número máximo de uso simultâneo desse filtro, para além disso guarda também o número de utilizações no momento do mesmo.
- **struct quantidade_filtro:** Esta estrutura de dados tem a funcionalidade de manter o nome de um filtro assim como a quantidade de utilizações presentes num certo comando.

- **struct tasks:** Neste caso a *struct* desempenha o papel de guardar os dados mais relevantes de um pedido, acrescentando ainda

```
typedef struct lligada {
    char *nome_filtro;
    char *nome_executavel;
    int maximo;
    int atual;
    struct lligada *prox;
} *Filtro;
```

Na *struct lligada*

```
struct quantidade_filtro {
    char *nome_filtro;
    int utilizacoes;
};
```

Na *struct quantidade_filtro*

```
typedef struct tasks {
    char *comando;
    char **ordem_filtros;
    struct quantidade_filtro *nomes_filtros;
    int numero;
    int processamento;
    int pid;
    char *input_file;
    char *output_file;
    struct tasks *prox;
} *Task;
```

Na *struct tasks*

- **Filtro filtros :**
- **Task tasks = NULL :**
- **char *pasta_filtros :**
- **int numero_tasks = 0 :**

Capítulo 4

Conclusão

Durante a realização deste trabalho prático embora o mesmo tenha sido apelidado de simples e até mesmo básico por parte da equipa docente, sentimos que exigiu uma grande organização da nossa parte para que não fosse perdido o foco do problema que estavamos a enfrentar. Tendo a aplicação dos conceitos teóricos revelado-se, de uma maneira geral, bastante interessante e desafiante pois tal como ***Ward Cunningham*** disse ***It's all talk until the code runs***.

Uma das peças fundamentais para a concretização do projeto foram as resoluções dos guiões práticos apresentadas pela equipa de docentes e até mesmo as nossas, pois serviram de apoio para o esclarecimento de dúvidas que surgiram, o que nos permitiu finalizar o mesmo com todas as funcionalidades solicitadas. Consideramos que a maior dificuldade foi sentida no momento inicial pois não conseguimos visualizar o produto final da nossa abordagem ao problema, mas conforme o estabelecimento de *mini checkpoints* conseguimos ultrapassar este impasse.