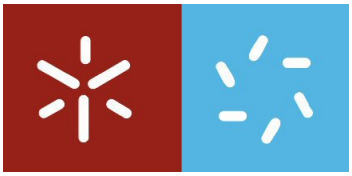

Universidade do Minho

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)

17 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 concorrentemente, 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 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 e demonstrar 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*.

Primeiramente, descreveremos as funcionalidades que o nosso programa suporta e, de seguida, faremos uma breve caracterização do nosso código, referindo a arquitectura de processos e os mecanismos de comunicação utilizados.

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 e quando o processo é iniciado o cliente recebe um sinal (*SIGUSR1*).

3.2 Servidor (*aurrasd.c*)

3.2.1 Variáveis Globais

- **Filtro filtros:** Esta variável dá acesso à *struct lligada* que 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.
- **Task tasks:** Esta variável tem uma estrutura de dados que lhe é associada, sendo esta responsável por guardar uma lista de *tasks*, armazenando em cada *nodo* o conjunto de dados necessários para processar uma determinada *task* (comando usado para a invocar, nome

dos filtros pela ordem que devem ser aplicados, nome de cada filtro e respetiva quantidade de vezes que deve ser executado, número da task, estado do processameno, *pid* do processo filho responsável pelo processamento do pedido, nome do ficheiro original e nome do ficheiro processado).

- **char* pasta_filtros:** Nome da diretória onde se encontram os ficheiros de filtros.
- **int numero_tasks:** Número de tasks recebidas até ao momento.

3.2.2 Descrição do Funcionamento

O servidor começa por testar se na sua inicialização recebeu os três argumetos necessários para o seu funcionamento. Caso isso se verifique, passa então à criação de um pipe que estabelecerá a comunicação entre o servidor e os clientes.

De seguida, é informado, através desse *pipe* do *pid* do cliente, para que possa tomar conhecimento do *pipe_lercliente* que traz a indicação da tarefa que deve ser efetuada. Caso esta tarefa seja um pedido de verificação do estado, será chamada a função *status*, caso seja uma requisição de transformação de ficheiro de áudio, será invocada a função *transform*. Este processo será repetido até que seja recebido um sinal de *SIGINT* ou *SIGTERM* que fechará de forma elegante o servidor. A seguir serão descritas as funções referidas.

- **Status:** Começa por criar um processo filho, que enviará através do *pipe_escrever* a informação do estado atual do programa ao cliente.
- **Transform:** No momento inicial armazena a informação referente ao pedido na estrutura de dados criada para o efeito (*tasks*), de seguida, verifica se o pedido pode ser executado utilizando a função *disponibilidade*. Em função do *output* da função referida, será tomado um de três caminhos:
  - O pedido é descartado (-1).
  - O pedido é colocado em espera (0).
  - O pedido é executado (1).

  Passamos agora a explicar a arquitetura de processos adotada para a execução da transformação de um ficheiro: é criado um processo filho para monitorizar as operações, sendo que de seguida cria-se um outro processo filho dentro do *monitor* que será responsável pela criação dos processos que executarão a aplicação dos filtros.

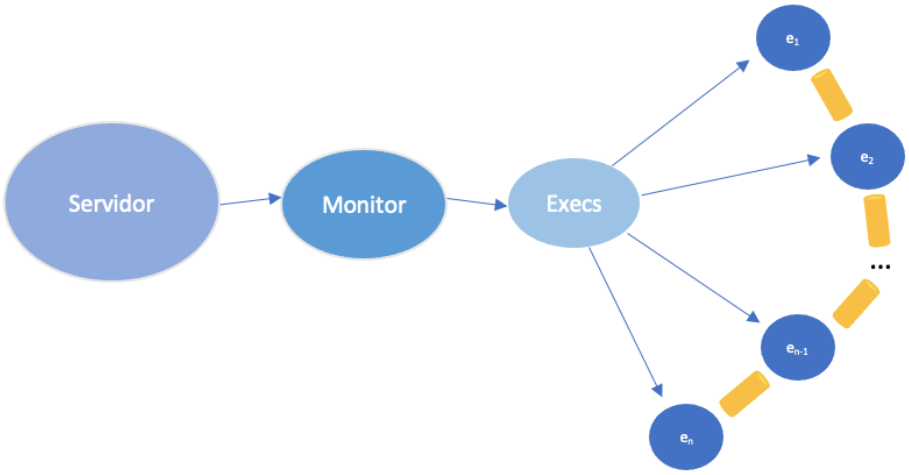

Figura 3.1: Diagram de Processos da *Transform*

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 a necessidade de uma grande organização da nossa parte para que não fosse perdido o foco do problema que estavamos a enfrentar. A aplicação dos conceitos teóricos revelou-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 docente 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, sendo o principal obstáculo a visualização do produto final para a nossa abordagem ao problema, mas conforme foram estabelecidos *mini checkpoints* conseguimos ultrapassar este impasse e suceder com uma boa implementação.