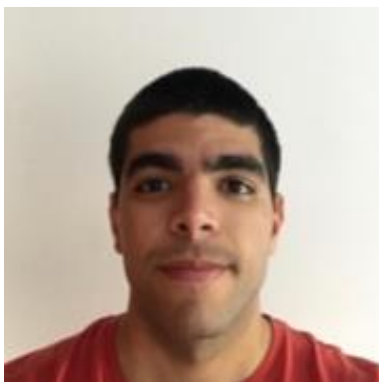




Universidade do Minho

# Aurras: Processamento de Ficheiros de Áudio

Trabalho prático da UC Sistemas Operativos



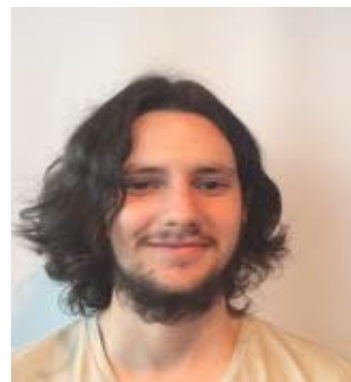
94166

Samuel de Almeida Simões  
Lira



93253

David Alexandre Ferreira  
Duarte



81667

Daniel José Coutinho  
Gonçalves de Faria

## Índice

Introdução.....	3
Descrição da solução.....	4
Arquitetura.....	4
Cliente .....	4
Servidor .....	4
Conclusões .....	6

## Introdução

No âmbito da unidade curricular Sistemas Operativos foi proposto aos alunos que implementassem um serviço capaz de transformar ficheiros de áudio por meio de filtros.

A aplicação desenvolvida permite, de acordo com o solicitado no enunciado, aplicar um ou um conjunto de filtros a um ficheiro de áudio determinado pelo utilizador através de um comando.

Eis a lista de implementações do programa:

- Arrancar o servidor com o ficheiro de configuração e a pasta com os executáveis (o conteúdo do ficheiro de configuração é guardado em memória)
- Permitir a aplicação de diferentes tipos de filtros que são determinados pelo ficheiro de configuração lido no arranque do servidor
- Concorrência de pedidos

Ao longo do presente relatório será explicado todo o processo de desenvolvimento e todas as decisões tomadas para a realização deste projeto.

## Descrição da solução

### Arquitetura

A aplicação desenvolvida foi baseada num sistema cliente – servidor.

### Cliente

No que diz respeito ao cliente este será um processo capaz de interpretar um conjunto de comandos passados como argumento e consoante o número de argumentos ele decide como tratar a resposta.

```
bin$ ./aurras ../samples/sample-1-so.m4a out.mp3 eco alto rapido
```

Figura 1- Exemplo de execução de um cliente

A comunicação do cliente para o servidor é feita com base em *pipes* com nome (no programa será o *file descriptor cs\_fd*). Após algum tratamento da informação passada na execução é enviado o buffer com a função *write* e os campos *cs\_fd*, *buffer* e *strlen(buffer)*.

### Servidor

O servidor é iniciado e fica à espera de um pedido vindo do cliente.

Depois de receber o pedido, o servidor cria um processo filho, tornando possível a execução paralela, para analisar os argumentos recebidos e, assim, perceber que processo deve executar. O pedido pode ser um de dois: *status* ou *transform*.

No primeiro caso, *status*, com base no estado do servidor será impressa na parte do cliente uma lista com os processos executados e em execução (um log) e ainda uma lista com os filtros em uso dos disponíveis.

No segundo caso, *transform*, o processo de transformação de um ficheiro será iniciado.

Do pedido recebido devem fazer parte um conjunto de argumentos que são o ficheiro ao qual se deve aplicar a transformação, o nome do ficheiro destino onde guardar o ficheiro origem transformado e o ou os filtros a aplicar ao ficheiro de modo a efetuar a transformação pretendida.

O encadeamento dos inputs e outputs dos processos é feito com a ajuda de pipes anónimos. Assuma-se a seguinte execução com comandos na bash:

```
bash$ bin/aurrasd-filters/aurrasd-echo < samples/samples-1.m4a | bin/aurrasd-filters/aurrasd-tempo-half > output.mp3
```

Traduzindo esta linha temos o seguinte processo de execução:

- `bin/aurrasd-filters/aurrasd-echo < samples/samples-1.m4a` : esta parte do comando representa a execução do programa *aurrasd-echo* que recebe como input o ficheiro

*samples-1.m4a*. Isto traduz-se na aplicação num *open* do ficheiro e de seguida um *dup2* para que o standard input do processo em questão passe a ser o *file descriptor* do ficheiro aberto.

- | bin/aurrasd-filters/aurrasd-tempo-half : Aqui tem-se a utilização de um pipe e o que esta parte da linha significa é que o output da primeira execução terá de ser o input desta última. Na aplicação isto traduz-se na utilização de pipes anónimos em que o processo com a execução de *aurrasd-echo* terá de escrever o output num pipe anónimo e o processo com a execução de *aurrasd-tempo-half* terá de ler do mesmo pipe anónimo.
- > output.mp3 : Esta será a parte final em que o output do ultimo processo terá de ser especificamente para um ficheiro com a assinatura apresentada. Na aplicação traduz-se novamente numa utilização de um *open* com a flag de criação de ficheiro e ainda no uso de *dup2* para que o output do processo seja para o *file descriptor* deste último *open*.

## Conclusões

O desenvolvimento deste trabalho prático foi de grande ajuda para consolidar os conhecimentos adquiridos nas aulas desta unidade curricular, principalmente sobre o uso de *pipes anónimos e com nome*, gestão de processos, utilização de `dup` e `fork`;

Foram encontradas ao longo do desenvolvimento algumas dificuldades, umas maiores que outras, mas que com o esclarecer de dúvidas e alguma pesquisa se viram resolvidas como por exemplo o encadeamento de pipes anónimos.

No entanto, algumas das funcionalidades pedidas no enunciado não foram implementadas, por alguma falta de compreensão e má gestão de tempo. Por exemplo: a comunicação entre filho e pai para agilizar a atualização de variáveis de estado. Embora isto não tenha sido implementado o grupo tem noção que o uso de sinais seria uma opção válida e compreende que isso resolveria o problema.

Em suma, foi um trabalho bastante elucidativo e que trouxe mais conhecimento e prática aos elementos do grupo.