

Universidade do Minho  
Mestrado Integrado em Engenharia Informática

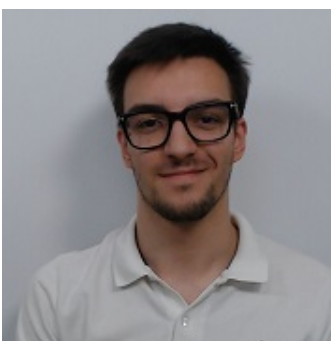
## Sistemas Operativos - Trabalho Prático

Grupo 51

André Silva - A87958

João Nunes - A87972

Junho 2021



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Funcionalidades</b>	<b>3</b>
2.1	Transformar . . . . .	3
2.2	Estado de funcionamento do servidor . . . . .	4
2.3	Ajuda . . . . .	4
2.4	Execução do servidor . . . . .	5
<b>3</b>	<b>Tópicos adicionais</b>	<b>6</b>
3.1	Cliente-servidor . . . . .	6
3.2	Encerramento do servidor . . . . .	6
<b>4</b>	<b>Conclusão</b>	<b>7</b>

# 1 Introdução

Este projeto, proposto na unidade curricular de Sistemas Operativos, consiste na elaboração de um serviço capaz de aplicar um ou vários filtros em ficheiros de áudio. Para além de transformar ficheiros de áudio, o servidor também disponibiliza outras opções como obter o estado de funcionamento do servidor e a informação de utilização (ajuda).

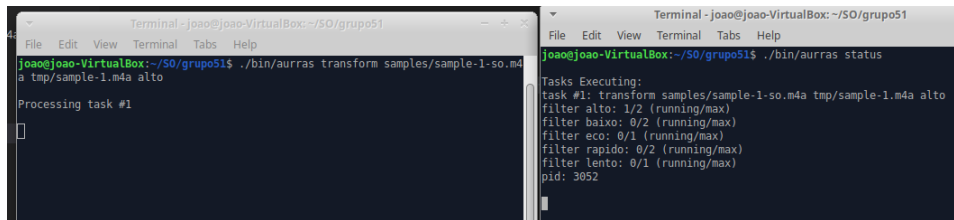
Para a comunicação cliente-servidor utilizámos dois pipes com nome, um que envia os comandos pedidos pelo cliente para o servidor e também o resultado do pedido para o cliente, e o outro que envia para o servidor informações úteis para conseguir ser feita a ligação cliente-servidor.

Inicialmente, não sentimos qualquer dificuldade em elaborar as partes simples do projeto. Mais tarde, tivemos algumas dificuldades em conseguir a execução de vários filtros ao mesmo tempo e também encerrar o cliente certo quando estão vários ligados ao servidor.

## 2 Funcionalidades

### 2.1 Transformar

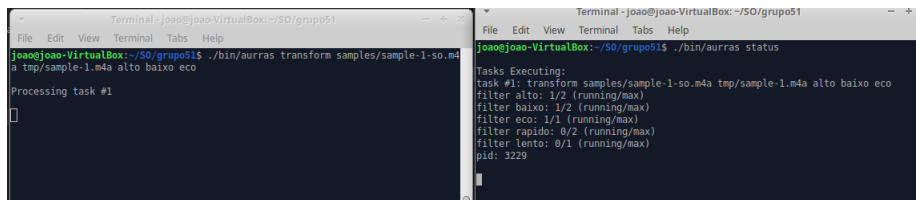
Comando principal do servidor, permite a transformação de um ficheiro de áudio por aplicação de um ou de vários filtros. Começamos por fazer *parsing* do input, separando todos os filtros dados. Verificamos se os filtros pedidos estão disponíveis e se sim, incrementamos e executámos cada um deles individualmente, de salientar que estas funções comunicam entre si através de pipes anónimos. Definimos um máximo de 32 pipes anónimos para cada tarefa, pois achamos que este número é mais que suficiente para a execução de uma tarefa.



```
Terminal - joao@joao-VirtualBox: ~/SO/grupo51
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurras transform samples/sample-1-so.m4a tmp/sample-1.m4a alto
Processing task #1

Terminal - joao@joao-VirtualBox: ~/SO/grupo51
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurras status
Tasks Executing:
task #1: transform samples/sample-1-so.m4a tmp/sample-1.m4a alto
filter alto: 1/2 (running/max)
filter baixo: 0/2 (running/max)
filter eco: 0/1 (running/max)
filter rapido: 0/2 (running/max)
filter lento: 0/1 (running/max)
pid: 3852
```

Figura 1: Transformar com 1 filtro



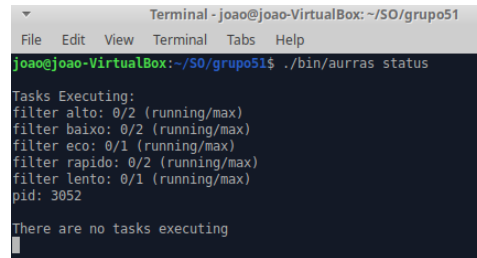
```
Terminal - joao@joao-VirtualBox: ~/SO/grupo51
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurras transform samples/sample-1-so.m4a tmp/sample-1.m4a alto baixo eco
Processing task #1

Terminal - joao@joao-VirtualBox: ~/SO/grupo51
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurras status
Tasks Executing:
task #1: transform samples/sample-1-so.m4a tmp/sample-1.m4a alto baixo eco
filter alto: 1/2 (running/max)
filter baixo: 1/2 (running/max)
filter eco: 1/1 (running/max)
filter rapido: 0/2 (running/max)
filter lento: 0/1 (running/max)
pid: 3229
```

Figura 2: Transformar com 3 filtros

## 2.2 Estado de funcionamento do servidor

Comando que mostra ao cliente o estado de funcionamento do servidor, isto é, mostra se existe algum processo em execução, os filtros disponíveis e o número máximo e de utilização de cada um.



```
Terminal - joao@joao-VirtualBox: ~/SO/grupo51
File Edit View Terminal Tabs Help
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurras status
Tasks Executing:
filter alto: 0/2 (running/max)
filter baixo: 0/2 (running/max)
filter eco: 0/1 (running/max)
filter rapido: 0/2 (running/max)
filter lento: 0/1 (running/max)
pid: 3052
There are no tasks executing
```

Figura 3: Estado sem qualquer pedido de transformação a executar

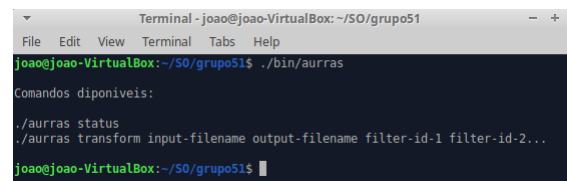


```
Terminal - joao@joao-VirtualBox: ~/SO/grupo51
File Edit View Terminal Tabs Help
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurras status
Tasks Executing:
task #1: transform samples/sample-1-so.m4a tmp/sample-1.m4a alto baixo eco
filter alto: 1/2 (running/max)
filter baixo: 1/2 (running/max)
filter eco: 1/1 (running/max)
filter rapido: 0/2 (running/max)
filter lento: 0/1 (running/max)
pid: 3229
```

Figura 4: Estado com um pedido de transformação a executar

## 2.3 Ajuda

Comando que envia para o cliente uma lista de todos os comandos existentes e como os deve utilizar.



```
Terminal - joao@joao-VirtualBox: ~/SO/grupo51
File Edit View Terminal Tabs Help
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurras
Comandos diponiveis:
./aurras status
./aurras transform input-filename output-filename filter-id-1 filter-id-2...
joao@joao-VirtualBox:~/SO/grupo51$
```

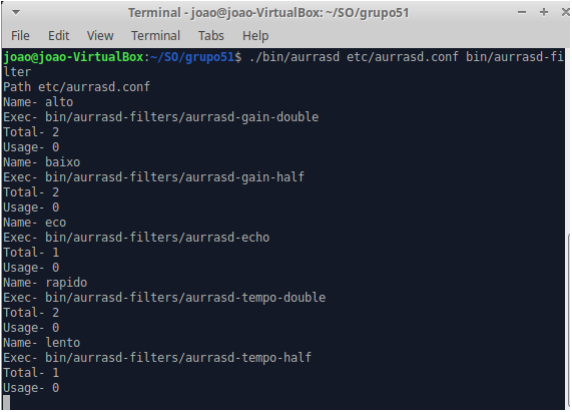
Figura 5: Comando que informa ao cliente os comandos disponíveis

## 2.4 Execução do servidor

Comando para a inicialização do servidor. Recebe o caminho para o ficheiro da configuração dos filtros e o caminho para a pasta que contém os executáveis dos filtros. Para facilitar a execução de um filtro, criámos uma estrutura:

```
struct filter {  
    char name[50];  
    char exec_path[100];  
    char exec[100];  
    int total;  
    int usage;  
};
```

Criámos também um array que vai conter elementos com esta estrutura, ou seja, vai conter todos os filtros que estão no ficheiro da configuração. Este passo vai-nos facilitar imenso todas as verificações necessárias antes da transformação de um ficheiro de áudio.



```
Terminal - joao@joao-VirtualBox: ~/SO/grupo51
joao@joao-VirtualBox:~/SO/grupo51$ ./bin/aurrasd etc/aurrasd.conf bin/aurrasd-filters
Path etc/aurrasd.conf
Name- alto
Exec- bin/aurrasd-filters/aurrasd-gain-double
Total- 2
Usage- 0
Name- baixo
Exec- bin/aurrasd-filters/aurrasd-gain-half
Total- 2
Usage- 0
Name- eco
Exec- bin/aurrasd-filters/aurrasd-echo
Total- 1
Usage- 0
Name- rapido
Exec- bin/aurrasd-filters/aurrasd-tempo-double
Total- 2
Usage- 0
Name- lento
Exec- bin/aurrasd-filters/aurrasd-tempo-half
Total- 1
Usage- 0
```

Figura 6: Inicialização do servidor

## 3 Tópicos adicionais

### 3.1 Cliente-servidor

Para evitar colisões e para uma melhor gestão dos vários clientes que se podem ligar ao servidor ao mesmo tempo, decidimos criar um pipe com nome para cada cliente. Para diferenciarmos cada um, optámos por utilizar o *pid* do cliente para definirmos o nome do pipe.

O problema desta implementação é que o servidor não vai saber qual é o *pid* do cliente para poder estabelecer uma ligação. A solução que encontramos foi criar um pipe com nome que apenas vai tratar das ligações, ou seja, o servidor vai esperar que um cliente envie para esse pipe o caminho para o seu próprio pipe. De salientar que quando o servidor termina o pedido de um cliente, envia para o pipe do cliente o *pid* do mesmo. O cliente vai esperar até receber o seu *pid* para terminar.

### 3.2 Encerramento do servidor

Se o servidor receber um *SIGTERM*, o sinal é tratado de modo a que o servidor rejeite quaisquer novos pedidos, permitindo que os pedidos em processamento acabem.

## 4 Conclusão

Em suma, e tendo em conta os objetivos e desafios que este projeto continha, achamos que o mesmo se encontra num nível satisfatório. Possui todas as funcionalidades pedidas operacionais e uma boa gestão dos filtros, permitindo ainda a comunicação concorrente entre vários clientes e o servidor. Deste modo, podemos concluir que construímos um serviço capaz de transformar ficheiros de áudio por aplicação de uma sequência de filtros perfeitamente funcional.