Sistemas Operativos

Trabalho Prático

Aurras: Processamento de Ficheiros de Audio

Grupo de Sistemas Distribuídos
Universidade do Minho

27 de Dezembro de 2020

Informações gerais

- Cada grupo deve ser constituído por até 3 elementos;
- O trabalho deve ser entregue até às 23:59 de 10 de Janeiro de 2020;
- Deve ser entregue o código fonte e um relatório de até 6 páginas (A4, 11pt) no formato PDF (excluindo capas e anexos);
- O trabalho deve ser realizado tendo por base o sistema operativo Linux como ambiente de desenvolvimento e de execução;
- O trabalho deve ser submetido num arquivo Zip com nome `grupo-xx.zip`, em que "xx" deve ser substituído pelo número do grupo de trabalho (p. ex: `grupo-01.zip`);
- O trabalho deve ser realizado com base na estrutura de pastas disponibilizada no Blackboard da unidade curricular;
- A apresentação do trabalho ocorrerá a 11 de Janeiro de 2020;
- O trabalho tem carácter obrigatório e representa 30% da classificação final.

Resumo

Implemente um serviço capaz de transformar ficheiros de áudio por aplicação de uma sequência de filtros. O serviço deverá permitir não só a submissão de pedidos de processamento de ficheiros de áudio como também a consulta das tarefas em execução, e número de filtros livres e em uso.

O serviço a implementar deverá ser constituído por um servidor e um por cliente que deverão comunicar por pipes com nome. Tanto o cliente como o servidor apenas disponibizarão um interface de linha de comando[1].

No desenho da sua solução tenha em conta os requisitos abaixo (obrigatórios):

- O servidor cria e mantém em execução um conjunto fixo de filtros de processamento de áudio;
- O conjunto de filtros é composto por múltiplas instâncias de vários tipos (diferentes transformações);
- Os tipos, respectivos identificadores, comandos externos e número de instâncias que devem ser criadas são definidos em ficheiro de configuração (ver exemplo abaixo);

[1] Os programas recebem os parâmetros de funcionamento como argumento da linha de comando e oferecem um interface textual ao utilizador. No caso concreto deste trabalho, o *standard input* não deverá ser usadonem pelo cliente nem pelo servidor. O *standard output* deverá ser usado pelo cliente para apresentar o resultado do estado do serviço, e pelo servidor para apresentar informação de debug que julgue necessária.

- Cada cliente solicita o processamento de um ficheiro áudio através da aplicação no servidor de uma sequência de filtros (sequência de identificadores);
- Os nomes do ficheiro original e do ficheiro processado são especificados como argumentos de linha de comando do cliente;
- No final da transformação o áudio resultante é transferido para o cliente que submeteu o pedido em causa (em formato MP3)[2];
- O processamento de um pedido de um cliente não poderá ser iniciado pelo servidor enquanto não existirem instâncias livres de todos os filtros necessários para a sua execução. As instâncias de filtro são consideradas ocupadas consoante estejam ou não a processar um ficheiro;
- O servidor inicia o tratamento dos pedidos dos clientes de forma sequencial;
- Considere que os programas executáveis que implementam os filtros já existem no sistema de ficheiros dentro de uma pasta definida no arranque do servidor (ver exemplo abaixo)[3]. Para efeito do trabalho considere os filtros disponibilizados na pasta aurrasd-filters++ (não é necessário desenvolver novos filtros).

Procure ter ainda em conta os seguintes aspectos, desejáveis mas não obrigatórios:

- O servidor deve suportar o processamento concorrente de pedidos;
- O servidor deve evitar a criação de ficheiros temporários;
- O servidor deve terminar de forma graciosa[4] a sua execução e a dos seus filtros sempre que receber o sinal SIGTERM;
- Deve utilizar a estrutura de código e makefile disponibilizada como ponto de partida do projecto.

Funcionalidade

Este serviço deverá suportar, pelo menos, as seguintes funcionalidades:

- submeter pedidos de processamento de ficheiro de áudio, conforme o exemplo abaixo;

```
$ ./aurras transform ./samples/sample-1.m4a output.m4a alto eco rapido
```

- obter o estado de funcionamento do servidor, conforme o exemplo abaixo;

```
$ ./aurras status
task #3: transform ./samples/sample-1.m4a output-1.m4a baixo rapido
task #5: transform ./samples/sample-2.m4a output-3.m4a rapido rapido baixo eco
filter alto: 0/3 (in use/total)
filter baixo: 2/4 (in use/total)
filter eco: 1/3 (in use/total)
filter lento: 0/2 (in use/total)
filter rapido: 3/3 (in use/total)
pid: 6354
```

- obter informação de utilização:

```
$ ./aurras
./aurras transform input-filename output-filename filter-id-1 filter-id-2 ...
```

- executar o servidor:

```
$ ./aurrasd config-filename filters-folder
```

  Por exemplo,

```
$ ./aurrasd etc/aurrasd.conf bin/aurras-filters
```

[2]Considere que os filtros utilizados pelo servidor convertem qualquer formato de áudio para MP3.

[3]Os filtros disponibilizados como exemplo fazem uso do programa `ffmpeg`. Assegure-se que esse programa está instalado e é executável sem necessidade de especificação do seu caminho (*path*) no sistema de ficheiros. Caso não esteja, deverá instalá-lo usando o gestor de pacotes recomendado pela sua distribuição de Linux.

[4]Por "graciosa" entende-se deixar terminar os pedidos em processamento e subsquentemente terminar todos os processos

Programas cliente e servidor

Deverá ser desenvolvido um cliente que ofereça uma interface com o utilizador via linha de comando que permita suportar a funcionalidade descrita e ilustrada acima. O utilizador poderá agir sobre o servidor através dos argumentos especificados na linha de comando do cliente. Deverá ser também ser desenvolvido um servidor, mantendo em memória a informação relevante para suportar a funcionalidade acima descrita.

Tanto o cliente como o servidor deverão ser escritos em C e comunicar via *pipes com nome*. Na realização deste projecto não deverão ser usadas funções da biblioteca de C para operações sobre ficheiros, salvo para impressão no *standard output*. Da mesma forma não se poderá recorrer à execução de comandos directa ou indirectamente através do interpretador de comandos (p. ex.: *bash* ou `system()`).

Note que, tal como no exemplos apresentados acima, poderão existir múltiplas tarefas em execução num dado momento.

Exemplo de um ficheiro de configuração do servidor

O ficheiro de configuração do servidor é composto por sequências de 3 linhas de texto: identificador do filtro; ficheiro executável que o implementa (sem argumentos); número de instâncias a criar.

Segue-se um exemplo de configuração:

```
alto
aurrasd-gain-double
2
baixo
aurrasd-gain-half
2
eco
aurrasd-echo
1
rapido
aurrasd-tempo-double
2
lento
aurrasd-tempo-half
1
```

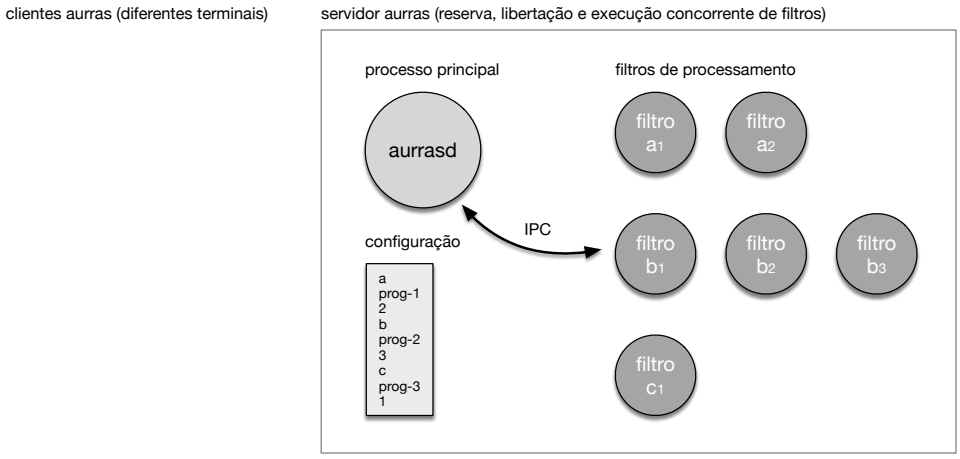
Ilustração do funcionamento do serviço

Os diagramas abaixo apresentados pretendem apenas exemplificar o processo de gestão do conjunto dos filtros criados por um servidor, em função do seu ficheiro de configuração e da sequência dos pedidos submetidos por três clientes. Neste exemplo, cada cliente inicia a sua execução de forma sequencial a partir de terminais independentes.

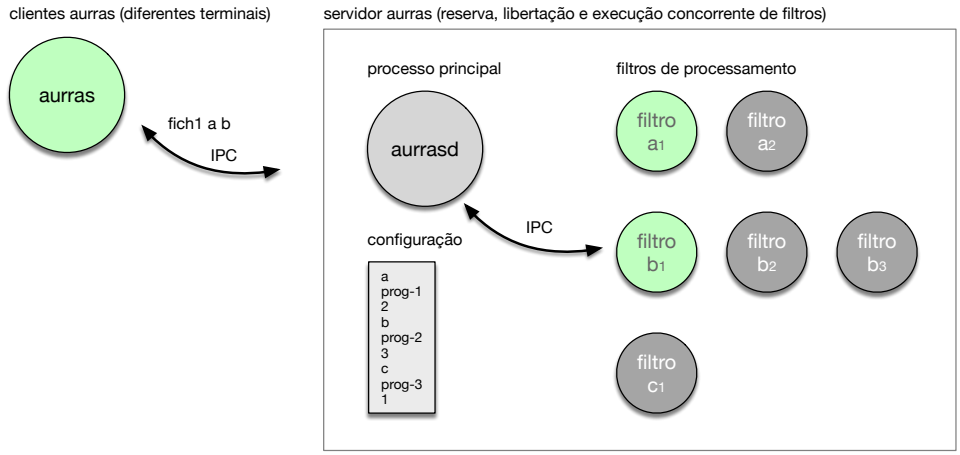
Tenha em conta que nos diagramas não estão detalhados os mecanismos concretos de comunicação entre processos (IPCs), nem em número nem em espécie. Conforme a solução proposta – e eventuais limitações inerentes – poderá ter necessidade de dois ou mais pipes com nome entre clientes e servidor. O mesmo se poderá dizer entre o servidor "principal" e os filtros por ele criados. Note ainda que cada filtro no diagrama pode, na prática, ter de ser implementado por um ou mais processos.

Em qualquer dos casos, no relatório que acompanhará o trabalho que irá desenvolver, terá de descrever e justificar a sua solução, desde logo no que diz respeito à arquitectura de processos como também no escolha e uso concreto dos mecanismos de comunicação.

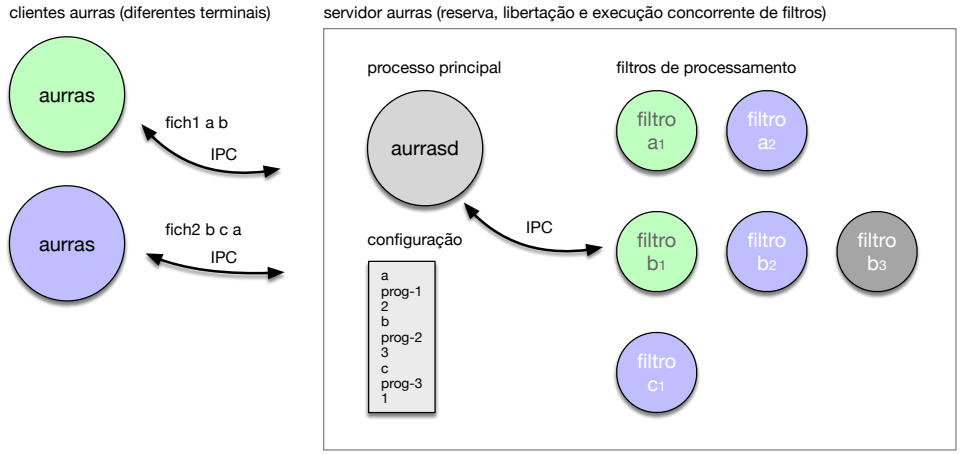
Estado inicial



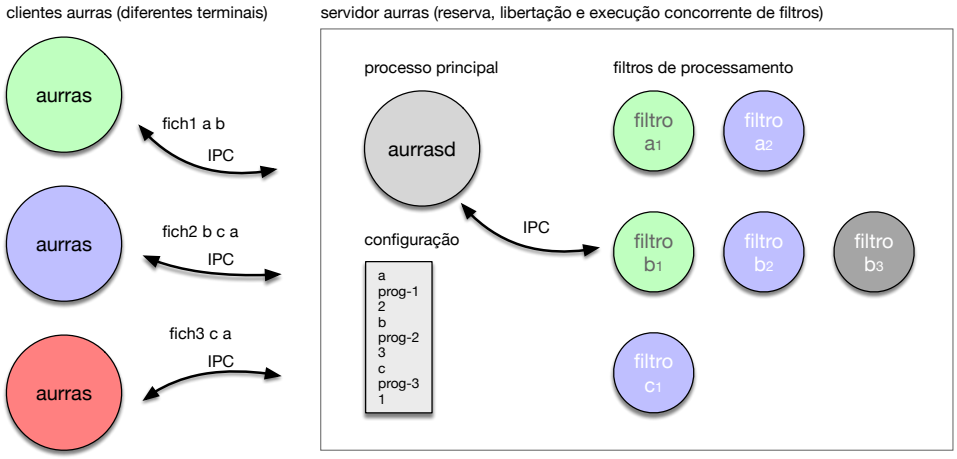
Pedido de um primeiro cliente



Pedido de um segundo cliente

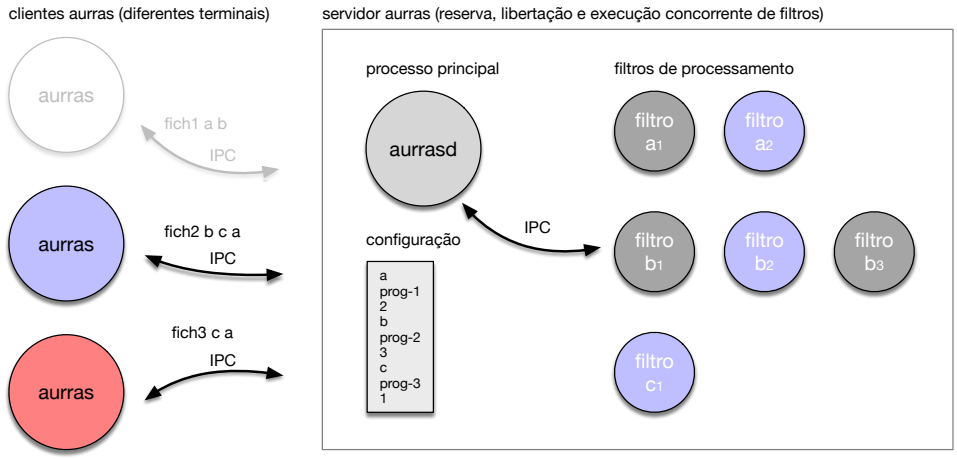


Pedido de um terceiro cliente (não pode ser processado)



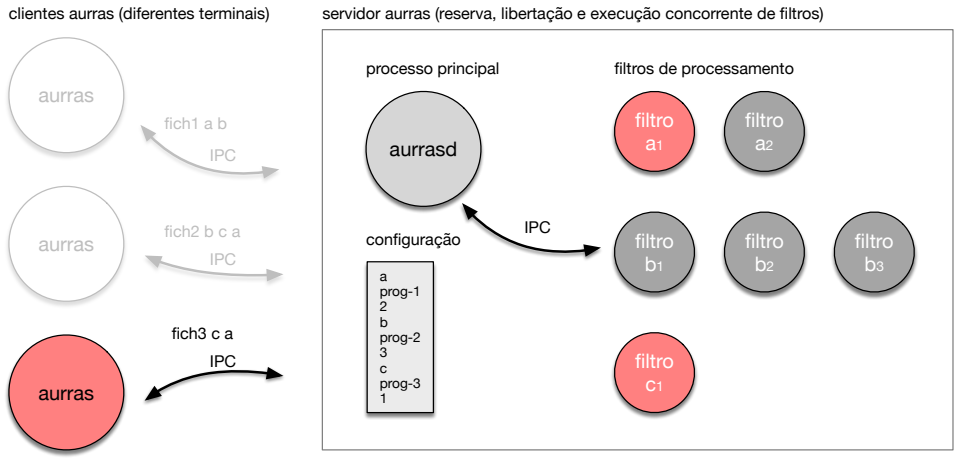
Note que o pedido não pode ser processado porque não há filtros disponíveis (livres) do tipo xxx++ e yyy++.

Conclusão do pedido do primeiro cliente



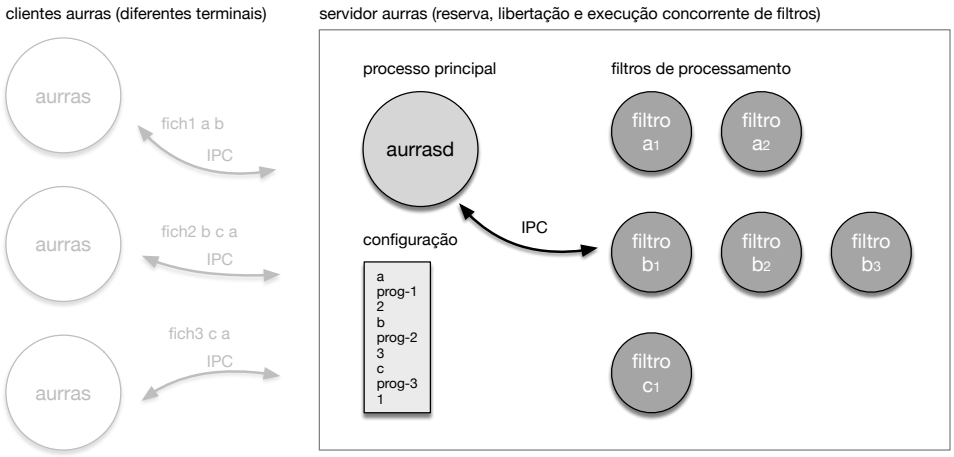
Note que o pedido do terceiro cliente ainda não pode ser satisfeito.

Conclusão do pedido do segundo cliente



Note que o pedido do terceiro cliente já pode ser processado.

Conclusão do pedido do terceiro cliente



Makefile

Tenha em conta que a Makefile que se apresenta deverá ser usada como ponto de partida mas poderá ter necessidade de a adaptar de modo a satisfazer, por exemplo, outras dependências do seu código-fonte. Em todo o caso, deverá manter sempre os objectivos (*targets*) especificados: all++, servidor++, cliente++, clean++, e test++. Não esqueça que por convenção a indentação de uma Makefile é especificada com uma tabulação (*tab*) no início da linha (nunca com espaços em branco).

```
all: server client

server: bin/aurrasd

client: bin/aurras

bin/aurrasd: obj/aurrasd.o
	gcc -g obj/aurrasd.o -o bin/aurrasd

obj/aurrasd.o: src/aurrasd.c
	gcc -Wall -g -c src/aurrasd.c obj/aurrasd.o

bin/aurras: obj/aurras.o
	gcc -g obj/aurras.o -o bin/aurras

obj/aurras.o: src/aurras.c
	gcc -Wall -g -c src/aurras.c obj/aurras.o

clean:
	rm obj/* tmp/* bin/{aurras,aurrasd}

test:
	bin/aurras samples/sample-1.mp3 tmp/sample-1.mp3
	bin/aurras samples/sample-2.mp3 tmp/sample-2.mp3
```