Sistemas Operativos

Trabalho Prático

Aurras: Processamento de Ficheiros de Audio

Grupo de Sistemas Distribuídos
Universidade do Minho

27 de Abril de 2021

Informações gerais

- Cada grupo deve ser constituído por até 3 elementos;
- O trabalho deve ser entregue até às 23:59 de 17 de Junho;
- Deve ser entregue o código fonte e um relatório de até 10 páginas (A4, 11pt) no formato PDF (excluindo capas e anexos), justificando a solução, nomeadamente no que diz respeito à arquitectura de processos e da escolha e uso concreto dos mecanismos de comunicação.
- 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 em data a anunciar, previsivelmente entre os dias 18 e 23 de Junho;
- O trabalho tem carácter obrigatório e representa 50% 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 disponíveis e em uso.

O serviço a implementar deverá ser constituído por um servidor e por um cliente que deverão comunicar por pipes com nome. Tanto o cliente como o servidor devem receber os parâmetros de funcionamento como argumento da linha de comando. O *standard output* deverá ser usado pelo cliente para apresentar o estado do serviço ou o estado de processamento do pedido ("pending", "processing"), e pelo servidor para apresentar informação de debug que julgue necessária.

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

- Existem diferentes tipos de filtros (transformações) que podem ser aplicadas;
- Os tipos de filtros, respectivos identificadores, comandos externos correspondentes e número máximo de instâncias que podem estar a correr concorrentemente são definidos em ficheiro de configuração (ver exemplo abaixo);

- Cada cliente solicita o processamento de um ficheiro áudio, através da aplicação no servidor de uma sequência de filtros. Para tal, o cliente passa como argumentos de linha de comando os nomes do ficheiro original e do ficheiro processado, assim como uma sequência de identificadores de filtros;
- O cliente é informado caso o pedido tenha ficado pendente, e quando entra em processamento. O comando termina quando o resultado (ficheiro processado) está disponível.
- O servidor deve suportar o processamento concorrente de pedidos, mas o processamento de cada pedido não poderá ser iniciado enquanto, para algum dos filtros necessários para a sua execução, o número de instâncias a correr esteja no limite máximo.
- 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), e que convertem qualquer formato de áudio para MP3, fazendo uso do programa `ffmpeg`. Assegure-se que este programa está instalado e é executável sem necessidade de especificação do seu caminho (*path*) no sistema de ficheiros. Para efeito do trabalho considere os filtros disponibilizados na pasta `aurrasd-filters`, não sendo necessário desenvolver novos filtros.

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

- O servidor deve evitar a criação de ficheiros temporários;
- Se receber o sinal SIGTERM, o servidor deve terminar de forma graciosa, deixando primeiro terminar os pedidos em processamento ou pendentes, mas rejeitando a submissão de novos pedidos.
- Deve utilizar a estrutura de código e makefile disponibilizada como ponto de partida do projecto.

Interface

Este serviço deverá poder ser usado do seguinte modo:

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

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

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

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

- obter informação de utilização:

```
$ ./aurras
./aurras status
./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
```

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()`).

Exemplo de um ficheiro de configuração do servidor

O ficheiro de configuração do servidor é composto por uma sequência de linhas de texto, uma por tipo de filtro, contendo: identificador do filtro; ficheiro executável que o implementa (sem argumentos); número máximo de instâncias concorrentes. 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
```

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