Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Inverno de 2021/2022

Série de Exercícios 3

Construa os programas e a biblioteca indicados na Parte II, usando a linguagem C, tendo o cuidado de eliminar repetições no código fonte e de isolar funcionalidades distintas em diferentes ficheiros fonte. Entregue o código desenvolvido, devidamente indentado e comentado, bem como os *makefiles* para gerar os executáveis e as bibliotecas a partir do código fonte. Assegure-se de que o compilador não emite qualquer aviso sobre o seu código com a opção **`-Wall`** activa, e de que no final da execução do programa não existem recursos por libertar (memória alocada dinamicamente e ficheiros abertos). Deve verificar essa situação com o utilitário Valgrind. Em caso de erro irrecuperável, o programa não deve terminar descontroladamente, deve, no mínimo, emitir uma mensagem informativa e em seguida terminar. O código desenvolvido será valorizado segundo os seguintes critérios, em importância decrescente: correção, eficiência, clareza.

Encoraja-se a discussão dos problemas e das respectivas soluções com outros colegas (tenha em consideração que a partilha directa de soluções implica, no mínimo, a anulação das entregas dos alunos envolvidos).

Parte I - Preparação do ambiente de desenvolvimento

Valgrind

Para verificar se um programa liberta toda a memória alocada dinamicamente deverá utilizar a ferramenta **`valgrind`**.

Instalação: **`$ sudo apt-get install valgrind`**

Documentação: **`$ man valgrind`**

ALSA

Advanced Linux Sound Architecture é um projeto *open source* vocacionado para a criação de funcionalidades áudio para o sistema operativo Linux (https://www.alsa-project.org/wiki/Main_Page).
Um dos produtos deste projeto é uma biblioteca que permite aos utilizadores incorporarem nos seus programas o acesso a dispositivos de som (https://www.alsa-project.org/wiki/ALSA_Library_API).

Instalação: **`$ sudo apt-get install libasound2-dev`**

Parte II - Realização

1. a) Programe a função **`file_tree_foreach`** que percorre a árvore de diretórios abaixo de **`dirpath`** e, para todos os ficheiros encontrados, invoca a função apontada por **`doit`** passando como argumentos o nome do ficheiro atual e o parâmetro **`context`**.

   ```
   int file_tree_foreach(const char *dirpath, void (*doit)(const char *, void *),
                         void *context);
   ```

   b) Realize um programa de teste que apresente na consola, por ordem alfabética, todos os ficheiros que se encontram sob um dado diretório e que obedeçam a determinado requisito. Por exemplo: **`$ prog_test /home/aluno "*.c"`** lista os ficheiros com extensão **`c`**, que se encontram sob o diretório **`/home/aluno`**. (é preciso colocar *.c entre aspas para que o interpretador de comando processe * como um caractere normal)

Pode utilizar o programa **`file_search.c`**, fornecido em anexo, como ponto de partida para este exercício.

2. Pretende-se criar uma biblioteca que permita aos seus utilizadores manipular informação áudio, codificada em ficheiros WAV[1], sem terem que conhecer os detalhes da sua formatação.

   A biblioteca deve apresentar a seguinte interface de utilização:

   `Wave` - tipo opaco que representa um conteúdo áudio em formato WAV;

   `Wave *wave_load(const char *filename)` - Criar um objeto `Wave` referente ao conteúdo do ficheiro indicado;

   `void wave_destroy(Wave *wave)` - Eliminar um objeto `Wave`;

   `int wave_get_bits_per_sample(Wave *wave)` - Obter o número de bits por amostra;

   `int wave_get_number_of_channels(Wave *wave)` - Obter o número de canais;

   `int wave_get_sample_rate(Wave *wave)` - Obter o ritmo de amostragem;

   `size_t wave_get_samples(Wave *wave, size_t frame_index,`
   `char *buffer, size_t frame_count);` - Obter um conjunto de amostras.
   (Uma *frame* é uma sequência de amostras, uma por cada canal.)

   a) Defina o tipo de dados `Wave` e programe as funções indicadas acima de acordo com as definições dadas. Baseie-se no formato canónico da norma WAVE definido neste documento: WAVE PCM soundfile format.

   b) Construa a biblioteca com as funções programadas na alínea anterior. Crie um *makefile* para gerar a biblioteca na versão de ligação estática e na versão de ligação dinâmica.

   c) Teste o funcionamento de ambas as versões da biblioteca criando um programa executável que a utilize. Pode utilizar o programa **`wave_dump.c`**, fornecido em anexo, que mostra dados do ficheiro WAVE indicado na linha de comando. Crie um *makefile* para gerar os executáveis.

3. Construa um programa de reprodução áudio a partir de ficheiros em formato WAVE. O programa começa por procurar no sistema de ficheiros todos os ficheiros com extensão **`wav`** e anotar os respectivos nomes e localizações. Posteriormente o programa entra numa fase de aceitação de comandos. Os comandos são: inserir ficheiro na lista de reprodução; vazar a lista de reprodução; reproduzir o áudio dos ficheiros da lista de reprodução.

   Em anexo, o programa **`wave_play.c`**, exemplifica a utilização da biblioteca ALSA e também da biblioteca construída no exercício 2.

Data limite de entrega: 22 de Janeiro de 2022

ISEL, 13 de Dezembro de 2021

[1] https://pt.wilipedia.org/wiki/WAV