Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Programação de Sistemas Computacionais
Verão 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. Pretende-se criar uma biblioteca que permita aos seus utilizadores manipular informação áudio, codificada em ficheiros WAVE[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 WAVE;

   `Wave *wave_create()` - Criar um objeto **`Wave`** vazio;

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

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

`void wave_set_bits_per_sample(Wave *wave, int bits_per_sample)` -

Definir o número de bits por amostra;

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

`void wave_set_number_of_channels(Wave *wave, int number_of_channels)` -

Definir o número de canais;

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

`void wave_set_sample_rate(Wave *wave, int sample_rate)` - Definir o ritmo de amostragem;

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

`size_t wave_append_samples(Wave *wave, uint8_t *buffer, size_t frame_count)` – Acrescentar no final da informação áudio existente no objeto **`wave`**, copiando de **`buffer`**, a sequência de *frames* com a dimensão **`frame_count`**. Uma *frame* é um conjunto de amostras, uma por cada canal. A função devolve o número de frames efetivamente copiadas.

`int wave_store(Wave *wave, char *filename)` - Criar um ficheiro em formato WAVE com o nome **`filename`** e com o conteúdo indicado por **`wave`**.

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 - **`libwave.a`** - e na versão de ligação dinâmica - **`libwave.so`**.

c) Teste o funcionamento de ambas as versões da biblioteca criando um programa executável que a utilize. Utilize um *makefile* para gerar os executáveis.

   Em anexo, o programa **`wave_capture.c`**, exemplifica a utilização da biblioteca ALSA e pode também ser usado para testar a biblioteca **`libwave`**.

   Para gerar o executável, depois de ter a biblioteca construída, pode usar o seguinte comando:

   ```
   gcc wave_capture.o -lwave -L<libwave_dir> -lasound -o wave_capture
   ```

   Pode verificar a correção do ficheiro WAVE produzido por inspeção do seu conteúdo, com o utilitário o `hexdump`:

   ```
   $ hexdump -C sample.wav | more
   ```

   Também poderá ouvir o som gravado com o utilitário `aplay`:

   ```
   $ aplay sample.wav
   ```

2. Construa um programa de captura de informação áudio para ficheiros em formato WAVE. O programa executa um ciclo permanente de aceitação e execução de comandos. Os comandos a implementar são: **start**, **stop**, **list**, **save** e **exit**. O comando **start** desencadeia uma captura de áudio; o comando **stop** termina a captura que está a decorrer; o comando **list** mostra as capturas realizadas; o comando **save** guarda em ficheiro WAVE a captura indicada em argumento; o comando **exit** termina o programa.

Data limite de entrega: 18 de junho de 2022

ISEL, 13 de maio de 2022