Este é o CS50x

OpenCourseWare

Doar (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://orcid.org/0000-0001-5338-2522)
(https://www.quora.com/profile/David-J-Malan)
(https://www.reddit.com/user/davidjmalan) (https://twitter.com/davidjmalan)

Laboratório 4: Volume

Você está convidado a colaborar com um ou dois colegas de classe neste laboratório, embora seja esperado que todos os alunos em qualquer um desses grupos contribuam igualmente para o laboratório.

O GitHub agora requer que você use SSH ou um token de acesso pessoal em vez de uma senha para fazer login, mas você ainda pode usar `check50` e `submit50`! Consulte **cs50.ly/github (https://cs50.ly/github)** para obter instruções, caso ainda não o tenha feito!

Escreva um programa para modificar o volume de um arquivo de áudio.

```
$ ./volume input.wav output.wav 2.0
```

Quando fazer

No sábado, 1º de janeiro de 2022, 1h59 GMT-3 (https://time.cs50.io/2021-12-31T23:59:00-05:00).

Arquivos WAV

Arquivos WAV

Os arquivos WAV (https://docs.fileformat.com/audio/wav/) são um formato de arquivo comum para representar áudio. Os arquivos WAV armazenam áudio como uma sequência de "amostras": números que representam o valor de algum sinal de áudio em um determinado momento. Os arquivos WAV começam com um "cabeçalho" de 44 bytes que contém informações sobre o próprio arquivo, incluindo o tamanho do arquivo, o número de amostras por segundo e o tamanho de cada amostra. Após o cabeçalho, o arquivo WAV contém uma sequência de amostras, cada uma com um único inteiro de 2 bytes (16 bits) representando o sinal de áudio em um determinado momento.

Escalar cada valor de amostra por um determinado fator tem o efeito de alterar o volume do áudio. Multiplicar cada valor de amostra por 2,0, por exemplo, terá o efeito de dobrar o volume do áudio de origem. Multiplicar cada amostra por 0,5, entretanto, terá o efeito de cortar o volume pela metade.

Tipos

Até agora, temos visto um número de diferentes tipos em C, incluindo `int`, `bool`, `char`, `double`, `float`, e `long`. Dentro de um arquivo de cabeçalho chamado `stdint.h` estão as declarações de vários outros tipos que nos permitem definir com muita precisão o tamanho (em bits) e o sinal (com ou sem sinal) de um inteiro. Dois tipos em particular serão úteis para nós neste laboratório.

- `uint8_t` é um tipo que armazena um inteiro sem sinal de 8 bits (ou seja, não negativo). Podemos tratar cada byte do cabeçalho de um arquivo WAV como um `uint8_t` valor.
- `int16_t` é um tipo que armazena um inteiro com sinal de 16 bits (ou seja, positivo ou negativo). Podemos tratar cada amostra de áudio em um arquivo WAV como um `int16_t` valor.

Começando

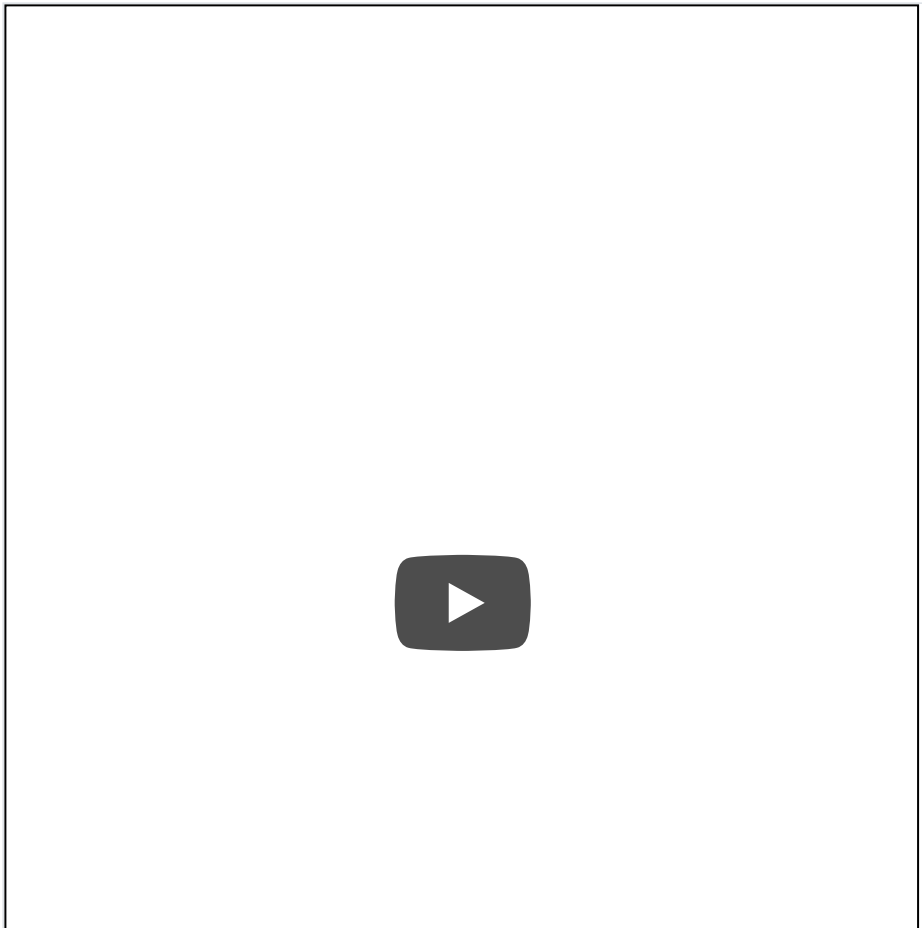
1. Faça login em ide.cs50.io (https://ide.cs50.io/) usando sua conta GitHub.
2. Na janela do seu terminal, execute `wget https://cdn.cs50.net/2020/fall/labs/4/lab4.zip` para baixar um arquivo Zip do código de distribuição do laboratório.
3. Na janela do seu terminal, execute `unzip lab4.zip` para descompactar (ou seja, descompactar) esse arquivo Zip.
4. Na janela do seu terminal, execute `cd lab4` para alterar os diretórios em seu `lab4` diretório.

Detalhes de Implementação

Conclua a implementação de `volume.c`, de modo que mude o volume de um arquivo de som por um determinado fator.

- O programa aceita três argumentos de linha de comando: `input` representa o nome do arquivo de áudio original, `output` representa o nome do novo arquivo de áudio que deve ser gerado e `factor` é a quantidade pela qual o volume do arquivo de áudio original deve ser escalado.
  - Por exemplo, se `factor` for `2.0`, então seu programa deve dobrar o volume do arquivo de áudio em `input` e salvar o arquivo de áudio recém-gerado em `output`.
- Seu programa deve primeiro ler o cabeçalho do arquivo de entrada e gravar o cabeçalho no arquivo de saída. Lembre-se de que esse cabeçalho tem sempre exatamente 44 bytes.
  - Observe que `volume.c` já define uma variável para você chamada `HEADER_SIZE`, igual ao número de bytes no cabeçalho.
- Seu programa deve então ler o restante dos dados do arquivo WAV, uma amostra de 16 bits (2 bytes) por vez. Seu programa deve multiplicar cada amostra por `factor` e gravar a nova amostra no arquivo de saída.
  - Você pode presumir que o arquivo WAV usará valores com sinal de 16 bits como amostras. Na prática, os arquivos WAV podem ter vários números de bits por amostra, mas assumiremos amostras de 16 bits para este laboratório.
- Seu programa, se for usado `malloc`, não deve vazar memória.

Passo a passo



Dicas

- Você provavelmente desejará criar uma matriz de bytes para armazenar os dados do cabeçalho do arquivo WAV que lerá do arquivo de entrada. Usando o `uint8_t` tipo para representar um byte, você pode criar uma matriz de `n` bytes para o seu cabeçalho com sintaxe como

```
uint8_t header[n];
```

substituindo `n` pelo número de bytes. Então você pode usar `header` como um argumento para `fread` ou `fwrite` ler dentro ou gravação do cabeçalho.

- Provavelmente, você desejará criar um "buffer" para armazenar amostras de áudio lidas do arquivo WAV. Usando o `int16_t` tipo para armazenar uma amostra de áudio, você pode criar uma variável de buffer com sintaxe como

```
int16_t buffer;
```

Então você pode usar `&buffer` como um argumento para `fread` ou `fwrite` ler dentro ou gravação do buffer. (Lembre-se de que o `&` operador é usado para obter o endereço da variável.)

- Você pode encontrar a documentação `fread` (https://man.cs50.io/3/fread)e `fwrite` (https://man.cs50.io/3/fwrite)útil aqui.
  - Em particular, observe que ambas as funções aceitam os seguintes argumentos:
    - `ptr`: um ponteiro para o local na memória para armazenar dados (ao ler de um arquivo) ou de onde gravar dados (ao gravar dados em um arquivo)
    - `size`: o número de bytes em um item de dados
    - `nmemb`: o número de itens de dados (cada um dos `size` bytes) para ler ou escrever
    - `stream`: o ponteiro do arquivo a ser lido ou escrito
  - De acordo com sua documentação, `fread` retornará o número de itens de dados lidos com sucesso. Pode ser útil verificar quando chegar ao final do arquivo!

Como testar seu código

Como testar seu código

Seu programa deve se comportar de acordo com os exemplos abaixo.

```
$ ./volume input.wav output.wav 2.0
```

Quando você ouve `output.wav` (clicando duas vezes em `output.wav` no navegador de arquivos ou executando a `open output.wav` partir do terminal), deve ser duas vezes mais alto do que `input.wav`!

```
$ ./volume input.wav output.wav 0.5
```

Quando você ouve `output.wav`, deve ser metade tão alto quanto `input.wav`!

▶ **Não sabe como resolver?**

Execute o seguinte para avaliar a exatidão do seu código usando `check50`. Mas certifique-se de compilar e testar você mesmo!

```
check50 cs50/labs/2021/x/volume
```

Execute o seguinte para avaliar o estilo do seu código usando `style50`.

```
style50 volume.c
```

Como enviar

Execute o seguinte para enviar seu trabalho.

```
submit50 cs50/labs/2021/x/volume
```