Tutorial de Instalação da SDL no Linux

Introdução

Este tutorial é para a instalação da SDL 2 em um ambiente Linux. É necessário usar uma versão nova da sua distro, de preferência, a mais recente. Isso evita que você seja forçado a compilar a SDL2 você mesmo, tenha problemas ao tentar fazê-lo, ou que tenha que usar uma versão obsoleta do g++.

Linux Debian-based (Ubuntu, Mint)

1. g++

Use `sudo apt-get install build-essential`. Terminada a instalação, digite `g++ --version`. O ideal é estar na versão 4.7 ou superior, para podermos usar as features do C++11. Quanto mais baixa a versão, menos features ela deve suportar.

Se, por algum motivo, você estiver impossibilitado de atualizar o g++ para 4.7, mas quiser usar os recursos disponíveis do do C++11, lembre-se de usar a flag `-std=c++0x` ao invés de `-std=c++11` durante a compilação.

2. SDL

Se você estiver no Ubuntu 14.04 ou superior, ou numa distro tão recente quanto, todas as bibliotecas estão presentes no repositório da distro. Faça:

```
sudo apt-get install libsdl2-dev libsdl2-image-dev
libsdl2-mixer-dev libsdl
```

E só. Aproveite o tempo do download pra fazer um lanche, esticar as pernas…

Caso as bibliotecas não estejam disponíveis, precisaremos compilá-las por conta própria. Instale o utilitário checkinstall. Ele faz o mesmo que o make install, mas cria uma package .deb para podermos desinstalar/reinstalar a biblioteca mais tarde.

```
sudo apt-get install checkinstall
```

Nas páginas da(s) biblioteca(s) faltante(s), baixe o arquivo tar.gz com o source code.

https://www.libsdl.org/download-2.0.php
https://www.libsdl.org/projects/SDL_image/
https://www.libsdl.org/projects/SDL_ttf/
https://www.libsdl.org/projects/SDL_mixer/

Extraia a pasta contida no tar para uma pasta à sua escolha. No terminal, faça:

```
cd <pasta extraída>
./configure --prefix=/usr
```

Se houver algum erro relativo a uma dependência faltante, instale-a e tente de novo. Por exemplo, o meu falhou pedindo a biblioteca freetype, que eu resolvi com instalando `libfreetype6-dev`. Resolvidos estes problemas de dependências, tente o configure novamente.

```
make
sudo checkinstall
```

O checkinstall te fará de três a cinco perguntas. As respostas recomendadas estão em negrito.

```
... set of package docs?  [y]: n
... description for the package >> Aperte Enter
... or press ENTER to continue: Aperte Enter
Do you want me to list them?  [n]: n
... (Saying yes is a good idea)  [n]: y
```

Pronto. Agora a biblioteca está instalada. O checkinstall gera um arquivo .deb na pasta de onde a instalação ocorreu. Guarde esse arquivo.

Obs: Esse guia se baseou (de fato, é quase uma tradução) no tutorial disponível no link a seguir. Agradecimentos ao autor original.

http://nothingtocode.blogspot.com.br/2013/07/setting-up-sdl2-in-ubuntu-or-linux-mint.html

Outra obs: Especialmente se for compilar a SDL2 fique atento a testes que falham na etapa do configure, eles podem ser responsáveis por problemas de compilação silenciosos que só se manifestam em runtime.

3. IDE

Você pode usar a IDE que quiser, de fato, é recomendado usar uma que você tenha experiência em configurar. Se você não souber fazê-lo, seguem algumas recomendações de IDEs em que conseguimos configurar o build com SDL2.

Para testar o seu projeto, cole o seguinte código na main. Se o projeto for configurado corretamente, o programa deve compilar sem erros.

http://pastebin.com/Jgk50FB7

- Qt Creator

Você pode baixar o Qt Creator do Software Center (ou similar), ou pelo apt-get (package `qtcreator`). Ele normalmente é usado com o Qt SDK (bibliotecas para fazer GUI), mas suporta projetos sem Qt e é uma boa IDE para C++. Também tem a vantagem de ser um pouco mais leve que IDEs populares, por ser escrita em C++.

Crie um projeto novo do tipo Non-Qt. Há duas opções de projeto C++, uma com o qmake, e uma com o cmake. O qmake é um gerenciador de builds próprio do Qt Creator, é mais fácil de usar com a IDE mas é menos flexível. O cmake (pode precisar da package `cmake`), é um gerenciador de builds muito mais poderoso e útil fora do Qt Creator, mas que também pode complicar sua vida.

Criado o projeto, copie o código de teste para o arquivo `main.cpp`.

Agora vá no arquivo do projeto, que vai depender de qual gerenciador de builds você escolheu.

Para o qmake (arquivo .pro):

- ➢ Adicione a linha `CONFIG += c++11` no início do arquivo
- ➢ Adicione a linha `LIBS += -lSDL2 -lSDL2_image -lSDL2_ttf -lSDL2_mixer` no final do arquivo
- ➢ Adicione a linha `INCLUDEPATH += /usr/include/SDL2` no final do arquivo

Para o cmake (cmakelists.txt):

- ➢ Adicione as linhas abaixo antes de `add_executable`:
  ```
  list(APPEND CMAKE_CXX_FLAGS "-std=c++11 ${CMAKE_CXX_FLAGS}")
  link_libraries(SDL2 SDL2_image SDL2_ttf SDL2_mixer)
  include_directories(/usr/include/SDL2)
  ```
- ➢ Ao executar o CMake (na criação do projeto ou via Build > Run CMake), você pode especificar build debug ou release passando como argumento `-DCMAKE_BUILD_TYPE=debug` ou `-DCMAKE_BUILD_TYPE=release`

Se tudo estiver certo, ao dar build no projeto agora, ele deve compilar sem problemas.

Mais uma coisa: Crie o hábito de setar o Working Directory para a pasta do projeto nos seus projetos da disciplina. Será útil para não ter que copiar os recursos do jogo para a pasta do executável sempre. Na aba lateral, clique em Projects, depois em Run > Working Directory.

- Eclipse

O Eclipse é uma IDE extremamente popular entre desenvolvedores Java, que tem, já há alguns anos, suporte para desenvolvimento em C e C++. Não baixe pelo site (bugs) ou pelo Software Center (vem sem suporte ao desenvolvimento em C/C++). Baixe a package `eclipse-cdt` pelo terminal.

Crie um projeto novo de C++, usando o Linux GCC. Crie um arquivo Main.cpp e coloque o código de teste lá.

Agora, vá ao menu Project > Properties > C/C++ Build > Settings.

- ➢ Em GCC C++ Compiler > Miscellaneous, acrescente `-std=c++11` às Other Flags.
- ➢ Em GCC C++ Compiler > Includes, acrescente /usr/include/SDL2 aos Include Paths.
- ➢ Em GCC C++ Linker > Libraries, adicione quatro entradas à lista Libraries (-l): SDL2, SDL2_image, SDL2_mixer, SDL2_ttf.

Só isso deve ser suficiente para compilar seu projeto. Acontece que o indexador do Eclipse tem uma peculiaridade que reconhece algumas construções de C++ (como o std::unordered_map) como erro. Para resolver isso: Project > Properties > C/C++ General > Paths and Symbols > Symbols > Add.

Crie um símbolo com nome `__GXX_EXPERIMENTAL_CXX0X__` e sem valor. Ao

confirmar, o Eclipse vai pedir pra reconstruir o índice, aceite. Dê Clean e em seguida Build no projeto. O editor deve parar de acusar erros.

Por último, procure a configuração de Working Directory do projeto e altere para uma pasta que achar adequada. Essa pasta será onde você colocará as pastas de recursos (imagens, mapas…) que forneceremos, e é importante que você saiba exatamente qual o Eclipse vai usar.