Chapitre 1

Installation et configuration de SDL2

1.1 Generalité bibliothéque

schema$_{c}ompil.png$
$Les fichierssource(les fichiers.c/.cpp/\dots)sontd'abordlusparleprprocesseurquines'occuperaquedetr$

$Ensuite, lecompilateurtransformelesourceenun fichierintermdiaire(.o/.obj), compil, maisquin'est$

1.2 Librairie Statique/Dynamique

Bibliothèque statique :

Sera compilée avec le programme et directement intégrée dans l'exécutable final. Il n'y aura donc pas besoin de .so/.dll lors de l'exécution mais des .a et .h. L'avantage de la compilation statique est de rendre l'exécutable indépendant (il peut fonctionner sur n'importe quelle plateforme compatible, sans pour autant nécessiter l'installation de la bibliothèque), toutefois, il est, de par ce fait, plus lourd et vous ne pourrez pas mettre à jour les bibliothèques sur lesquelles il repose, sans le recompiler.

Bibliothèque dynamique :

Fait que l'exécutable ira chercher les fonctions à exécuter dans des fichiers séparés (.so/.dll). L'exécutable sera plus léger et les bibliothèques pourront

être mises à jour sans recompiler l'exécutable. De plus, une bibliothèque peut être chargée à la volée (durant l'exécution d'un programme) et ainsi fournir des fonctionnalités en plus, si elle est présente : les systèmes de modules ou ajouts sont souvent implémentés sous la forme de bibliothèques chargées lorsque nécessaires.

Soit la librairie libXXX.a (ou libXXX.so) se trouvant dans un répertoire dont le chemin absolu estchemin. Pour compilerun fichier sourceprog.cfaisant appel à des fonctions de cette librairie, il faut taper la ligne de commande suivante :
gcc prog.c -Lchemin -lXXX -o prog

dans le cas des librairies dynamiques,si le programme allait toujours chercher les librairies au même emplacement, il suffirait de changer cet emplacement pour quele programme devienne inutilisable1, ou qu'il faille le recompiler. C'est pourquoi pour chercher l'emplacement des librairiesdynamiques, on s'aide d'une variable d'environnement appeléeLD$_LIBRARY_PATHCettevariableindiqueaupr$

$exportLD_LIBRARY_PATH = /usr/local/lib$

$Sil'onveutquelesprogrammescherchentdans/usr/local/lib, dans/usr/X11R6/libetdanslerp$

$exportLD_LIBRARY_PATH = . : /usr/X11R6/lib : /usr/local/lib$

$Enpratique, vousnedfinirezjamaiscettevariablemaisvousajouterezdesfichierssadfinition$

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/users/profs/habibi/libs
```

1.3 Installation SDL2

Pour l'installation de la SDL2 , il faut allez sur le site de la SDL "https ://www.libsdl.org" , une fois l'archive telecharger il faut la dezipper .

1.4 compilation

On compile un projet sdl de la facons suivante :

gcc -Wall -Wextra src/main.c -o main -L./lib -I./include -lSDL2-2.0

Option :
-L : permet de spécifier où trouver la bibliothèque libSDL2.so
-l : permet de dire qu'il faut utiliser la bibliothèque SDL2
-I : permet de spécificer où sont les fichiers .h

OU alors avec :

gcc -o executable fichier1.c fichier2.c fichier3.c ... ‘sdl2-config –cflags –libs‘

Les options respectives à ajouter à la compilation avec GCC (après -lSDLmain -lSDL) sont :

-lSDL$_image$: $pourSDL_image - lSDL_ttf$: $pourSDL_ttf - lSDL_mixer$: $pourSDL_mixer$
$(AvecCMake)$:

$PourlaSDL, ilestncessairedecrerunnouveaudossier(parexemple, build)ctdeceluiextrait.Ensuite, ils$

$cmake - G´UnixMakefiles˘../dossier\_des\_sources\_de\_la\_SDL$
$make$

$CetexemplecreraunMakefileetestdoncparfaitpourLinux.L'lmentquevouspouvezpasserl'option-Gdterminelatypedefichierdevanttregnr.$

Chapitre 2

Arborescence du projet

Cette organisation permet d'éviter à ce que vous ayez à installer (ou faire installer) la bibliothèque sur chacun des postes que vous utiliserez : elle est intégrée dans le projet.

Ce qui nous donne l'arborescence suivante :

```
/
include
- SDL2
- - ...
- - tous les fichiers .h de la SDL
- - ...
lib
- libSDL2-2.0.so (dynamique) ou libSDL2.a (statique) (pour Linux)
- libSDL2.lib (pour Windows) ou libSDL2*.a (pour MinGW)
src
- main.c
SDL2.dll (pour Windows)
les fichiers de projets (.cbp (pour Code::Blocks), .sln (pour Visual Studio).
```

Chapitre 3

Configuration du projet

Une fois que tous les fichiers sont prêts, que le code source est prêt, il ne reste plus qu'à créer le projet de votre nouvelle application SDL 2 afin de la compiler.
Nous devons spécifier le répertoire contenant les fichiers d'entête (include), le répertoire des fichiers de la bibliothèque (lib) et les bibliothèques à intégrer. Ne pouvant pas décrire ces étapes pour tous les éditeurs existants, seuls les plus connus seront présentés.

GCC Linux

Dans votre ligne de compilation, il suffit de spécifier le répertoire où chercher la bibliothèque (-L./lib), le répertoire des entêtes (-I./include) et d'indiquer que vous voulez utiliser la SDL 2.0 (-lSDL2-2.0).

Ce qui donne :

gcc -Wall -Wextra -L./lib -I./include -lSDL2-2.0 src/main.c -o main

Lorsque la bibliothèque est installée à l'aide du gestionnaire de paquets, ou de la commande make install, il n'est pas nécessaire de spécifier les dossiers lib et include car les fichiers sont dans les dossiers que le compilateur scanne par défaut.

Chapitre 4

Code SDL

4.1 Initialisation de la SDL

```
if (SDL_Init(SDL_INIT_VIDEO) != 0 )
{
    fprintf(stdout,"Échec de l'initialisation de la SDL (%s)\n",SDL_GetError(
    return -1;
}
```

$\mathrm{SDL}_{I}nit()prendunseulparametre$:

$\mathrm{SDL}_{I}NIT_{T}IMER$	Initialise le sous-système des chronomètres
$\mathrm{SDL}_{I}NIT_{A}UDIO$	Initialise le sous-système pour l'audio
$\mathrm{SDL}_{I}NIT_{V}IDEO$	Initialise le sous-système pour le rendu
$\mathrm{SDL}_{I}NIT_{J}OYSTICK$	Initialise le sous-système pour les joysticks
$\mathrm{SDL}_{I}NIT_{H}APTIC$	Initialise le sous-système pour le retour de force
$\mathrm{SDL}_{I}NIT_{G}AMECONTROLLER$	Initialise le sous-système pour les contrôleurs de jeux
$\mathrm{SDL}_{I}NIT_{E}VENTS$	Initialise le sous-système pour les événements
$\mathrm{SDL}_{I}NIT_{E}VERYTHING$	Initialise tous les sous-systèmes nommés ci-dessus
$\mathrm{SDL}_{I}NIT_{N}OPARACHUTE$	Avec cette option, la SDL ne mettra pas en place de f

4.2 Ouvrire une Fenetre

```
SDL_Window* pWindow = NULL;
pWindow = SDL_CreateWindow("Ma première application SDL2",SDL_WINDOWPOS_UNDEF
                                                          SDL_WINDOWPOS_UNDEF
                                                          640,
```

```
                                                    480,
                                                    SDL_WINDOW_SHOWN);

if( pWindow )
{
    /* Suite du programme */

    SDL_DestroyWindow(pWindow); // Destruction de la fenêtre
}
else
{
        fprintf(stderr,"Erreur de création de la fenêtre: %s\n",SDL_GetError());
}
```

La fonction créant une fenêtre est : SDL$_{C}reateWindow().Celle-ciretourneunpointeursurunevariable$

$SDL_{C}reateWindow()attendplusieursparamtresafindecrerlafentre.Ceux-$
$cisont : SDL_{C}reateWindow(charnom_{f}enetre, charposX, charposY, inttaille_{l}argeur, inttaille_{h}auteu$

$Pourlaposition$:

$-SDL_{W}INDOWPOS_{U}NDEFINED-- > laisselelibrechamplabibliothquedeplacerlafentreoellel$

$-SDL_{W}INDOWPOS_{C}ENTERED-- > permetdetoujoursavoirunefentrecentresurl'cran.$

$Pourlesoption$:

SDL$_{W}INDOW_{F}ULLSCREEN$	Crée une fenêtre plein écran
SDL$_{W}INDOW_{F}ULLSCREEN_{D}ESKTOP$	Crée une fenêtre plein écran à la résolution
SDL$_{W}INDOW_{O}PENGL$	Crée une fenêtre pouvant être utilisée pour
SDL$_{W}INDOW_{S}HOWN$	Crée une fenêtre et l'afficher
SDL$_{W}INDOW_{H}IDDEN$	Crée une fenêtre cachée
SDL$_{W}INDOW_{B}ORDERLESS$	Crée une fenêtre sans bordure
SDL$_{W}INDOW_{R}ESIZABLE$	Crée une fenêtre redimensionnable
SDL$_{W}INDOW_{M}INIMIZED$	Crée une fenêtre minimisée
SDL$_{W}INDOW_{M}AXIMIZED$	Crée une fenêtre maximisée
SDL$_{W}INDOW_{I}NPUT_{G}RABBED$	Crée une fenêtre et récupère le focus d'entré
SDL$_{W}INDOW_{I}NPUT_{F}OCUS$	Crée une fenêtre et donne le focus d'entrée
SDL$_{W}INDOW_{M}OUSE_{F}OCUS$	Crée une fenêtre et donne le focus de la sou
SDL$_{W}INDOW_{F}OREIGN$	La fenêtre n'est pas créée par la SDL