Fiche annexe sur l'organisation des répertoires

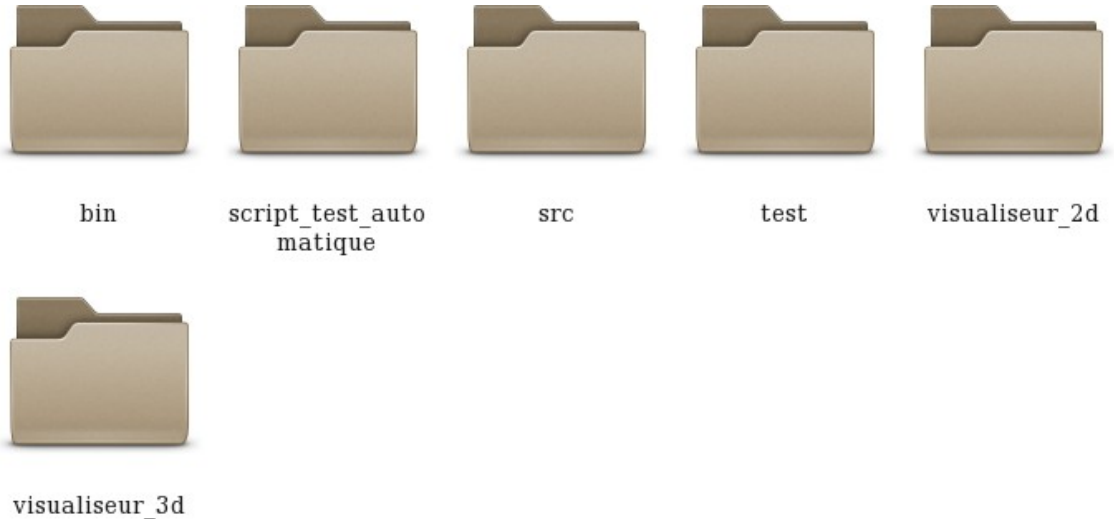
Votre répertoire devra toujours être organisé de la manière suivante:

Premièrement, l'ensemble de votre projet sera contenu dans une archive unique. Cette archive tar.gz devra être nommée sous la forme nom1_nom2_projet_echec.tar.gz conformément aux consignes de rendues.

C'est sous cette forme que vous rendrez votre projet.

Une fois décompressée à l'aide de la commande
`tar xvfz nom1_nom2_projet_echec.tar.gz`
cette archive devra donner un répertoire du même nom.

Ce répertoire contiendra votre environnement de travail qui suivra cette hiérarchie:



bin/ est le répertoire contenant votre exécutable. Il contiendra également temporairement les fichiers de sorties de l'application. Les différents visualiseurs et scripts viendront supposer par défaut que votre exécutable se trouve dans *bin/jeu_echec*.
Au moment de rendre votre travail, ce répertoire ne devra contenir qu'uniquement votre fichier exécutable, et rien d'autre.

script_test_automatique/ est un répertoire contenant un script de lancement automatique de tests de votre application. Il sera utilisé en séance ultérieur.
Au moment de rendre votre travail, ce répertoire ne doit contenir que ce script sauf si vous écrivez vous même d'autres scripts de lancement de tests automatique. Ce répertoire est optionnel au moment du rendu, il n'est la que pour vous aider. Si vous n'avez pas modifié le code du script, il est inutile de mettre ce répertoire avec votre archive.

src/ est un répertoire contenant l'ensemble de vos fichiers sources du projet. Il contiendra l'ensemble des fichiers C (.c et .h), ainsi que les scripts de compilations et le fichier Makefile.

Au moment de rendre votre travail, ce répertoire ne devra contenir que ces fichiers de codes et scripts de compilations. Il ne devra pas contenir de fichiers objets (.o) binaires, ni exécutables. Ce répertoire ne devra pas non plus contenir de fichiers temporaires (commençant par un ., ou terminant par un ~).

test/ est un répertoire contenant vos fichiers de tests d'application qui seront appelés par le scripts contenues dans script_test_automatique/. Ce répertoire sera formé (d'au moins) trois sous répertoires: *test_unitaire_ok*, *test_unitaire_ko* et *test_integration*.
Au moment de rendre votre travail, ce répertoire ne devra contenir que des fichiers de tests textuels. Il ne devra contenir aucun fichier binaire ni fichier temporaire (commençant par un . ou terminant par un ~).

visualiseur_2d/ et
visualiseur_3d/ sont des répertoires contenant un code permettant de visualiser l'état de l'échiquier.
Au moment de rendre votre travail, ces répertoires ne devraient contenir que les fichiers sources du code. Ils ne devraient contenir ni fichiers binaires (ex. fichiers objets .o) ni exécutables. Ces répertoires sont optionnels au moment du rendu, ils ne sont la que pour vous aider. Si vous n'avez pas modifié le code des visualiseurs, il est inutile de mettre ces répertoires avec votre archive.

Note: Les répertoires indispensables au moment du rendu final sont: **src/**, **bin/**, et **test/**

Note: Vous devez demander explicitement au système à visualiser les fichiers cachés (fichiers terminant par ~, ou commençant par un .) pour savoir si ils sont présent ou non. Demandez à un enseignant si vous n'arrivez pas à visualiser ces fichiers.

Organisation de votre espace de travail.

Le projet place le fichier exécutable dans un répertoire différent des sources. Pour éviter de devoir constamment passer d'un répertoire à l'autre à partir du même terminal, il est conseiller d'ouvrir plusieurs terminaux. Chacun positionné dans le répertoire approprié.
Un premier terminal servira ainsi principalement à lancer la compilation à partir du répertoire **src/**, alors qu'un second terminal pourra servir à lancer l'exécutable à partir du répertoire **bin/**.

Vous avez le droit de lancer autant de terminaux que vous souhaitez.

Un exemple d'organisation vous est proposé dans la figure suivante alliant 2 terminaux, un éditeur de code, et un visualiseur d'échiquier. Libre à vous cependant de trouver l'organisation que vous préférez (multi-bureau, raccourci etc).
L'important étant d'être capable de réaliser la chaîne d'édition suivante de manière efficace (c.a.d rapidement, avec peu de commandes à taper et peu de mouvements ou clic de souris):

- modification du code
- compilation
- lancement de l'executable
- interaction avec l'échiquier
- visualisation du résultat

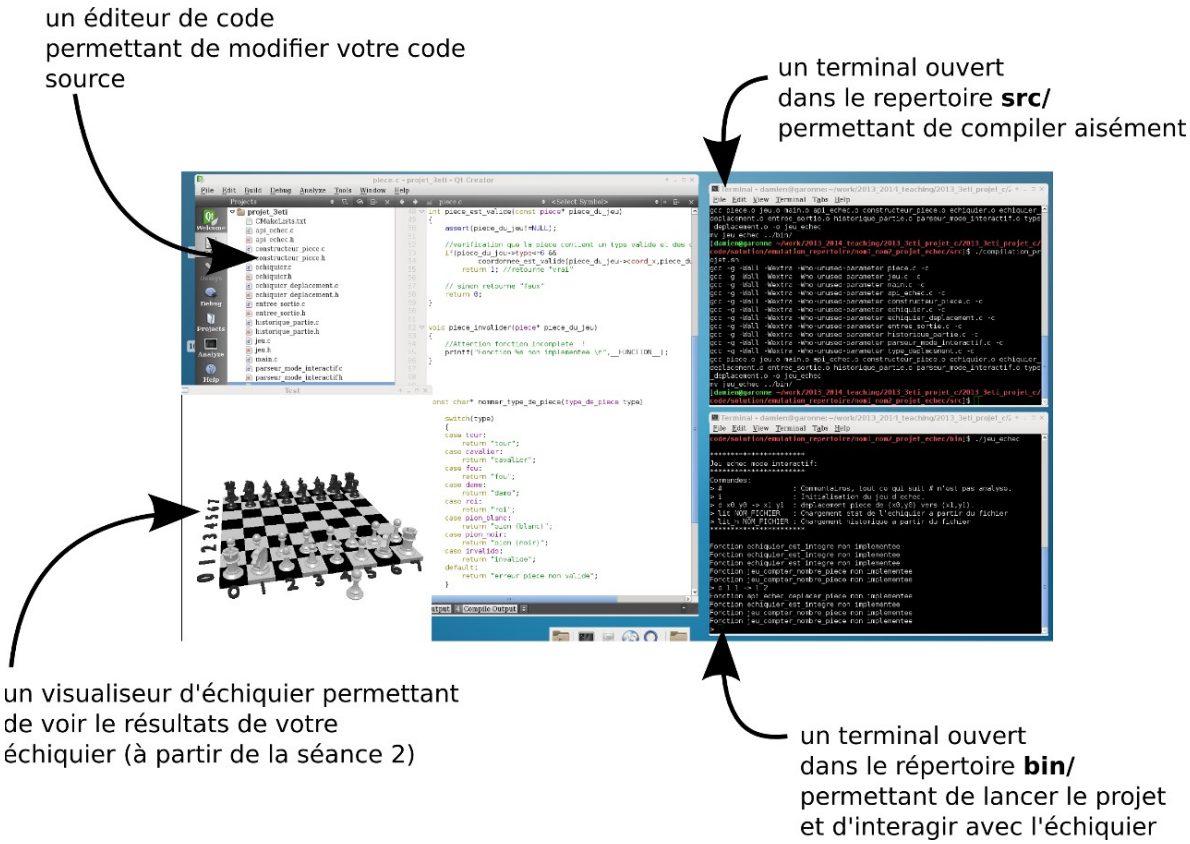


Illustration d'une organisation de l'espace de travail possible afin de passer rapidement de l'édition du code à la visualisation du résultat.

Répertoire de lancement d'exécutables.

Les différents exécutable et scripts interagissent entre eux en supposant que ceux-ci respecte la hiérarchie décrite précédemment et qu'ils sont lancés depuis leurs répertoires locales respectifs.

Voici les différents chemins à respecter en fonction de l'action à réaliser.

Pour compiler: On lancera la compilation depuis un terminal ouvert dans le répertoire **src/**. L'exécutable devant être automatiquement copié ou déplacé dans le répertoire **bin/**.

Pour lancer l'executable: L'executable du nom de *jeu_echec* sera lancé depuis un terminal ouvert dans le répertoire **bin/**. La commande associée sera alors la suivante:

```
$ ./jeu_echec
```

***Note 1**:* Les fichiers d'état d'échiquier et d'historiques écrit par le projet seront créés dans le répertoire courant du terminal. Dans le cas présent, ils seront dans le répertoire bin/.

***Note 2**:* Il serait possible de lancer directement l'executable depuis un autre répertoire en indiquant le chemin relatif ou absolue vers l'executable *jeu_echec*. Par exemple, depuis la racine du projet

avec la commande
```
$ bin/jeu_echec
```
Ou bien encore depuis le répertoire src/ avec la commande
```
$ ../bin/jeu_echec
```
Dans ces deux cas, les fichiers d'états d'échiquier seraient respectivement créés dans la racine du projet et dans le repertoire src/.

Note 3: Si vous transférez par mails/clé un fichier d'un ordinateur à l'autre, il est possible que celui-ci perde les informations indiquant que le fichier est executable. Il peut alors être nécessaire d'invoquer la commande suivante (en supposant le terminal ouvert dans le répertoire bin/) redonnant les droits d'executions sur le fichier:
```
$ chmod +x jeu_echec
```

Notons que dans tout les cas, il est préférable de recréer le fichier binaire sur chaque PC. Un binaire compilé sur un système 64 bits ne pourra pas être exécuté sur un système 32 bits.

Pour lancer le visualiseur: Le visualiseur (2D ou 3D) devra être lancé depuis un terminal ouvert dans son répertoire locale. Un Makefile est fourni permettant de compiler le code du visualiseur.
En lancant la commande
```
$ ./visualiseur
```
Le fichier d'état d'échiquier visualisé est cherché au chemin locale *../bin/echec_etat.txt.*
Si vous souhaitez visualiser un autre fichier d'état à un autre chemin appelé [PATH], vous pouvez lancer le visualiseur avec le chemin spécifique en tant qu'argument:
```
$ ./visualiseur [PATH]
```
Il est également possible de modifier le code du visualiseur afin de changer le fichier cherché par défaut.

Note: Le visualiseur 3D nécessite l'installation des librairies de visualisation 3D sur l'ordinateur. Pour ceux souhaitant utiliser le visualiseur 3D depuis leur ordinateurs personnels, il est nécessaire d'installer les librairies liées à OpenGL et GLUT. Ce sont des packages standards des distributions Linux. L'installation sous Mac peut nécessiter un travail manuel. La mise en place sous Windows est plus complexe.
Dans le cas contraire, le visualiseur 2D devrait pouvoir compiler sur toutes les plateforme.

Pour lancer un test d'application: Supposons que vous souhaitiez lancer le test d'application situé dans le fichier *test/test_unitaire_ok/test_unitaire_ok_01.txt*.
Il faut se placer dans le répertoire **bin/** de l'executable, et lancer la commande
```
$ ./jeu_echec < test/test_unitaire_ok/test_unitaire_ok_01.txt
```

Pour écrire un test d'application: On suppose que les fichiers de tests seront toujours lues depuis un executable situé dans le répertoire **bin/** (ou depuis le repertoire de script de lancement automatique). Dans le cas où votre test viens lire un fichier, il devra donc indiquer le chemin relatif depuis le répertoire **bin/** jusqu'au fichier à lire.
Pour cette raison, la lecture du *test_unitaire_ok_01.txt* contient la commande suivante indiquant le chemin pour lire le fichier depuis **bin/**:
```
lit ../test/test_unitaire_ok/test_unitaire_ok_01_entree.txt
```

Pour lancer l'ensemble des tests automatisés: On utilisera un terminal ouvert dans le répertoire script_test_automatique/ et on lancera la commande
```
$ python run_test.py
```