# Concevoir un moteur d’échec simple :

Pour intégrer un moteur d’échec dans une interface graphique telle que Arena, il faut qu’ils communiquent entre eux.

Nous avons le choix entre deux protocoles de communication, soit on utilise **UCI** ([Universal Chess Interface](https://fr.wikipedia.org/wiki/Universal_Chess_Interface)), ou bien on utilise le protocole **Xboard.**

Il faut savoir que la communication avec xboard se fait à l’aide des instructions **printf** du langage C.

Dans ce qui suit on verra le minimum de commandes nécessaires pour que notre moteur puisse interagir avec Arena.

Voici les commandes que notre moteur d’échec doit comprendre quand il les reçoit dans son entrée standard.

**new**

**force**

**go**

**<MOVE>**

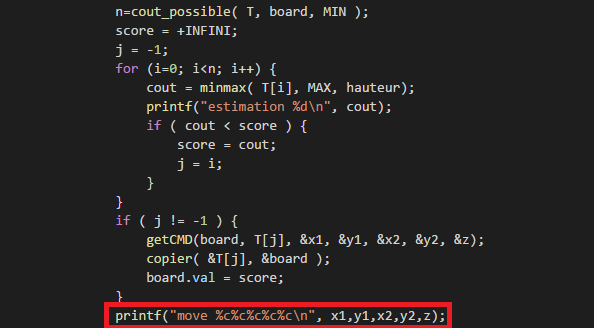
* **<MOVE>** : veut dire déplacement (4 caractères ou 5 caractères lorsqu’une promotion est impliquée). Le moteur d’échec envoie à Arena les mouvements sous la forme :

**printf("move d2d4");**

La machine quant à elle, elle envoie au moteur l’ors d’un mouvement une chaine de caractères de la forme :

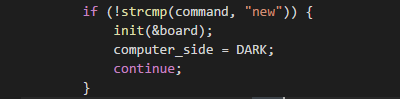
"d2d4" (sans le mot « move »).

Dans la portion du code suivante, on génère les configurations possibles, puis on estime le cout de chacune d’elle avec l’algorithme Min-Max et après avoir trouvé la bonne configuration on génère le déplacement, on met à jour la configuration de l’échiquier et en fin on envoie la commande de déplacement à Arena via l’instruction **printf**.



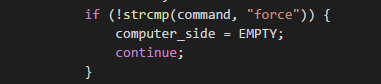
* **new** : indique à notre moteur de configurer l’échiquier pour un nouveau jeu, et de se considérer le coté qui ne va pas jouer en premier, tout simplement attendant les événements. Cela signifie qu'il va commencer à chercher et à faire un mouvement après avoir reçu un mouvement d'entrée.

Ce bout de code vérifie la commande envoyée depuis Arena, s’il s’agit de la commande « new » on initialise l’échiquier et on choisit le camp de la machine (les noirs dans ce cas).



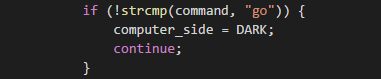
* **force** : Après cette commande le moteur ne fera plus aucun mouvement lui-même, il va juste accepter des mouvements pour le côté qui a le droit de le faire (en gardant une trace de qui a la main de faire un mouvement).

Là on met juste la variable qui décide le camp de la machine à une valeur neutre.



* **go** : indique au moteur de commencer à jouer pour le côté qui a maintenant le droit de faire un mouvement (indépendamment de ce qu'il faisait avant), et de continuer à générer des mouvements pour ce côté chaque fois que ce côté à la main.

À l’inverse de ce qu’on a fait dans la commande précédente (fore), ici on choisit la partie pour laquelle va jouer la machine (les noirs dans ce cas).



* **quit :** Cette commande fait arrêter le jeux. Donc on renvoie un simple return pour que le programme s’arrête.



Remarque : Il existe bien d’autres commandes un peu avancées à utiliser dans la communication en utilisant le protocole Xboard telles que les commandes « undo » et « redo » pour revenir en arrière, aussi les commandes de temps qui permet d’avoir le temps (pour avoir le temps configuré pour une partie) et la commande « hint » pour donner un indice au joueur.

# Récapitulatif :

* On envoie nos commandes à Arena à l’aide des commandes **printf**.
* Arena nous envoie ses commandes sur l’entrée standard (**stdin**), qu’on va recevoir dans une variable de type chaîne de caractères en utilisant l’instruction :

**fgets(line, 256, stdin);**

line : c’est la variable qui va contenir la chaîne lue.

256 : c’est le nombre ma de caractères.

stdin : l’entrée standard.

* Il faut créer une fonction (par exemple **void xboard()**) qui va faire ce traitement et puis faire un appel à cette fonction dans le main. Sinon, mettre le tout dans le main.

# Intégration dans Arena :

1. Générer l’exécutable de notre moteur d’échec.
2. Ouvrir Arena et cliquer sur **Engines**.
3. Cliquer sur **Intall** New Engine.
4. Sélectionner l’exécutable de notre moteur dans la fenêtre qui s’ouvre.
5. Choisir le protocole xboard quand il sera demandé de choisir entre UCI et xboard.