Projet n°2

Le but du projet est de manipuler les outils System V. Vous devez rendre le projet (le code avec le readme.txt) avant le vendredi 1er mars, délai de rigueur).

I. Jeu du labyrinthe invisible infernal

Le but du jeu du labyrinthe invisible infernal est de déplacer un valeureux guerrier dans un labyrinthe invisible poursuivi par un ou plusieurs méchants Minotaures. Le valeureux guerrier commence à une entrée et doit atteindre la sortie en évitant de se cogner aux murs invisibles (certains murs sont visibles au départ) et de se faire toucher par un méchant Minotaure. Dès qu'un mur est percuté par le valeureux guerrier, il s'affiche (mais il ne perd pas de vie). Dès qu'il est touché par un méchant Minotaure (s'il se trouve sur une case adjacente), il perd une vie et retourne au départ. S'il n'a plus de vie, il meurt et des funérailles nationales sont organisées (vous n'en tiendrez pas compte dans votre projet).

Nous reprenons l'éditeur du projet précédent pour la création des cartes. Le format des fichiers est identique.

Il est dorénavant possible de placer intégralement la carte en mémoire dans l'éditeur. Si votre éditeur fonctionnait, inutile de le modifier. Nous ne prenons plus en compte les sauvegardes.

Le jeu est constitué de 3 programmes différents (en plus de l'éditeur) : le **contrôleur**, le **minotaure** et le **joueur**.

Le **contrôleur** prend en argument un nom de fichier contenant la carte. Il crée ensuite trois outils IPC : une file de messages (FdM), un segment de mémoire partagée (SMP) et un tableau de sémaphores (TdS). Le SMP contient la carte (pour rappel, les cases valent 0 pour vide, 1 pour un mur invisible, 2 pour un mur visible, 10 pour le valeureux guerrier et 11 et plus pour les méchants Minotaures). La FdM permet au programme **minotaure** et au programme **joueur** de communiquer avec le **contrôleur**. Ce dernier reçoit des requêtes pour la connexion ou l'arrêt d'un programme **minotaure** ou **joueur**. Lorsque le contrôleur reçoit un SIGINT, il envoie d'abord un SIGINT à tous les programmes (**minotaure** et **joueur**), attend leur fin, puis supprime les outils IPC et s'arrête.

Le programme **minotaure** peut être exécuté jusqu'à 5 fois maximum. Une fois démarré, il envoie une requête dans la FdM pour récupérer les clés du SMP et du TS. Il place ensuite un Minotaure sur une case aléatoire de la carte. Périodiquement, il le déplace aléatoirement (haut, bas, droite et gauche). Dès qu'il reçoit un SIGINT, le programme **minotaure** prévient le programme principal via la file de messages puis s'arrête. Nous utiliserons un sémaphore pour s'assurer que le programme **minotaure** ne s'exécute pas plus de 5 fois.

Le programme **joueur** est identique au programme **minotaure**, sauf qu'il permet de déplacer le valeureux guerrier jusqu'à ce qu'il gagne (quand il atteint la sortie) ou qu'il perde (il se fait toucher par un méchant Minotaure). Contrairement au programme **minotaure**, il n'est pas possible d'exécuter plusieurs fois le programme **joueur**. La vérification se fera également à l'aide d'un sémaphore.

II. Recommandations

Voici quelques recommandations pour arriver à bout de ce projet :

- Commencer par tester les outils IPC indépendamment, ce qui vous permettra de valider chaque partie de votre projet ; les énoncés de TP sont là pour ça !
- Pour la FdM, commencer par identifier les différents messages et créez les structures nécessaires.
- Pour le TdS, identifiez d'abord quels sont les sémaphores nécessaires, autrement dit quels sont les cas d'exclusion mutuelle.

Le programme **joueur** doit à la fois permettre à l'utilisateur de déplacer le valeureux guerrier, mais également mettre à jour l'affichage avec le déplacement des méchants Minotaures. La solution est de rendre le `getch` non bloquant (en utilisant `timeout`, par exemple, ou `nodelay`, etc.).