UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
Info0601

Projet n°2 - IPC

Les voitures

Le but de ce projet est d'utiliser les outils IPC et les signaux.

1 Présentation du projet

Nous souhaitons réaliser une simulation autoroutière. Une carte contient des routes sur lesquelles des voitures se déplacent. Chaque voiture est simulée par un programme différent, nommé *voiture*, et la simulation est contrôlée et affichée par un programme nommé *contrôleur*.

Une file de messages permet de communiquer entre les voitures et le contrôleur. Nous distinguons les messages suivants :

— Récupération de la configuration ; données : PID du programme demandeur

— Envoi de la configuration ; données : clé du segment de mémoire partagée et du tableau de sémaphores

— Avertissement de modification de la carte par une voiture ; données : un message, un numéro de voiture (le contrôleur peut mettre à jour l'affichage)

La carte est mise en mémoire dans un segment de mémoire partagée par le contrôleur. Elle correspond à un tableau de 15 cases sur 30 (des `unsigned char`) contenant soit des 0 pour du vide, soit des 1 pour des routes, soit des valeurs supérieures à 1 pour les voitures (2 pour la voiture n°1, 3 pour la voiture n°2, *etc.*). Elle est suivi par les positions de toutes les voitures (deux `int` pour chacune). Le segment est accessible par tous les programmes : le contrôleur pour afficher les modifications de la carte à l'écran et les voitures pour récupérer les routes ou les obstacles (les cases vides ou les autres voitures), ainsi que les positions des voitures.

Les sémaphores sont utilisés pour gérer les problèmes de synchronisation sur le segment de mémoire partagée et le nombre de voitures connectées. Pour gérer les modifications de la carte, vous avez le choix entre deux solutions : un sémaphore pour toute la carte ou un sémaphore par chaque zone de 5 cases par 5. Évidemment, le deuxième choix sera mieux noté. Quel que soit votre choix, la gestion du nombre maximum de voitures doit être géré par un/des sémaphores.

1.1 Le contrôleur

Le contrôleur a besoin des arguments suivants :

— Le nom du fichier contenant la carte ; le format du fichier est identique au projet n°1 (titre de la simulation suivi par les cases, les obstacles correspondant aux routes)

— Le nombre de voitures maximum

— Les trois clés : de la file de messages, du segment de mémoire partagée et du tableau de sémaphores

Lors du démarrage, il crée les outils IPC en fonction de ses arguments puis les initialise (lecture de la carte, etc.). Il se met ensuite en attente de messages sur la file. Il peut également recevoir un signal d'arrêt (un `SIGINT`). Dans ce cas, il doit envoyer lui-aussi un signal à toutes les voitures présentes puis supprimer les outils IPC.

1.2 La voiture

La voiture a besoin des arguments suivants :

— La clé de la file de messages

— Sa rapidité (la valeur passée à la fonction `ncurses timeout`)

Lors du démarrage d'une voiture, celle-ci récupère les informations à l'aide de la file de messages. Puis elle vérifie qu'elle peut se connecter (via les sémaphores). À partir de ce moment, la voiture peut choisir une place libre pour se placer. Elle met à jour la carte et sa position. Régulièrement, la voiture se déplace : elle choisit une case route adjacente (droite, gauche, haut ou bas).

Le programme s'arrête lorsqu'il reçoit un signal du contrôleur ou de l'utilisateur. Si c'est de l'utilisateur, elle doit s'arrêter en libérant sa place (segment et tableau de sémaphores à mettre à jour).

— Pour obtenir des déplacements "intelligents", il est conseillé de faire choisir une direction par les voitures et de ne la changer qu'une fois un obstacle rencontré.

— Il peut être également intéressant de proposer à la voiture un changement de direction dès que deux possibilités sont possibles.

— L'algorithmique mise en œuvre dans les déplacements n'est pas un point important dans ce projet de programmation système.

2 Recommandations

Aucun rapport n'est demandé pour ce projet. Il est attendu que les étudiants fassent un effort particulier sur la conception de l'application, aussi bien en termes de programmation système que de programmation C. Par exemple, les structures doivent être déclarées dans des fichiers différents (un fichier *header* pour la définition d'une structure et la déclaration des fonctions, un fichier C pour la définition des fonctions), la fonction principale ne doit pas contenir tout le code, les appels systèmes doivent être vérifiés, *etc.* Un `makefile` sera obligatoirement fourni permettant de compiler les programmes avec les options attendues et présentées en cours, ainsi qu'un fichier `_lisezMoi.txt` indiquant comment utiliser vos applications.