Haute Ecole de la Province de Liège

Informatique industrielle: Labo 2 Suite

Rahali Nassim & Schmidt Sébastien - M18

Liège, le 22 février 2015

Table des matières

1 Organisation du projet

Pour la réalisation de ce projet, nous avons essayé au maximum de scinder le programme en parties. Nous aurions pu créer un seul fichier de code source mais la compréhension du code en aurait pâti. C'est la raison pour laquelle une série de processus a été créés afin de rendre le code compréhensible.

De plus, les processus vont se créer automatiquement, il n'y aura pas besoin de lancer les processus manuellement. On peut considérer le processus Gestion comme le père des processus. Il aura pour tâche de lancer X processus Bateau et Y processus Port. Le port créera ensuite ses propres processus Quai et son processus GenVehicle. Le nombre de processus bateau et Port à lancer sera spécifier dans un fichier de configuration qui sera lu au démarrage de l'application.

2 Structures de données

Pour ce projet, nous avons défini deux structures de données : Une pour les bateaux et une pour les quais. Ces structures seront contenues dans des mémoire partagées accessibles par tous les processus.

2.1 Structure pour les bateaux

```
typedef enum {SEA, ENTERS_PORT, DOCK, LEAVES_PORT} boat_p;
typedef enum {UNDEFINED, DOVER, CALAIS, DUNKERQUE} boat_d;

typedef struct Boat_t
{
  pid_t        pid;
  int        index;
  boat_p       position;
  boat_d       direction;
  int        state_changed;
  MessageQueue  mq1;
  MessageQueue  mq2;
} Boat;
```

Dans un premier temps, on a défini deux énumération pour la direction ainsi que la position du bateau. Ces énumération seront utilisées dans la structure. Par rapport au point de vue théorique, nous avons rajouté deux données : l'index du bateau ainsi qu'un indicateur pour savoir si les données ont été modifiées ou pas. Cette données est uniquement utilisées pour l'affichage et n'a donc pas grand intérêt. La mémoire partagée aura une taille égale à 6 fois la taille de cette structure.

2.2 Structure pour les quais

```
typedef struct Dock_t
{
  int index;
  int boat_index;
} Dock;
```

Chaque port possède X quais, les informations essentielles de ces quais seront contenues dans une mémoire partagée par port. Dans cette structure, on spécifiera l'index du quai ainsi que le bateau qui y est actuellement accosté. Si aucun bateau n'est présent le boat_index vaudra -1 et sera libre.

3 Les différents processus

3.1 Processus Gestion

Comme spécifié précédemment, ce processus va créé des processus fils grâce à la fonction fork(). On va ensuite exécuter un fichier grâce a la fonction execl qui peut également passer des arguments à ce processus. Aux processus Bateau, on leur fournira leur numéro d'index : Comme il y a 6 bateaux, chaque processus aura un index compris entre 0 et 5. Ce numéro sera utilisé par le processus Quai et Port afin de savoir quel bateau est sur le point de rentrer dans le port ou d'accoster. Quant au processus Port, on lui fournir le nom de son port (Calais, Dunkerque ou Douvre) afin de notamment savoir le nombre de quais que le port possède mais également pour créer des sémaphores. En effet les ports ont besoin de sémaphores uniques : on va dans ce cas utiliser le nom du port concaténé à un nom de sémaphore commun pour avoir un nom de sémaphore unique et facilement retrouvable pour les autres processus qui en auront besoin (comme le bateau lors de son entrée).

```
1  for (i = 0; i < nb_boats; i++)
2  {
3    if ((child_pid = fork()) < 0)
4    {
5      perror("fork failure");
6      exit(1);
7    }
8
9    if (child_pid == 0)
10   {
11     char* p = malloc(sizeof(p));
12     sprintf(p, "%d", i);
13     execl("Boat", "BOAT", p, NULL);
14   }
15 }
16
17
18 // Création des ports
19 for (i = 0; i < nb_ports; i++)
20 {
21   if ((child_pid = fork()) < 0)
22   {
23     perror("fork failure");
24     exit(1);
25   }
26
27   if (child_pid == 0)
28   {
29     char* p = malloc(sizeof(p));
30     sprintf(p, "%d", (i == 0) ? 3 : 2);
31     execl("Port", "PORT", ports_name[i], p, NULL);
32   }
33 }
```