Projet d'algorithmique

Cours d'Informatique de Deuxième Année

—Deug Sciences M3-MI—

Gestion d'un aéroport

- Ce travail doit être rendu au plus tard le mardi 20 Mai au secrétariat du DEUG Sciences, la soutenance aura lieu lors de la dernière séance de Travaux Pratiques ;
- la clarté et le soin apportés à la rédaction de votre travail, ainsi que la lisibilité de votre *listing* seront pris en compte ;
- ce travail peut être réalisé en binôme, chaque personne devant être présente à la soutenance et pouvant être interrogée individuellement ;
- la partie programmation devra être réalisée en *C ANSI*, compilable sous *HP–UX* (l'*UNIX* de *HP*).

---

1 Présentation du sujet

Le projet de ce semestre consistera en la réalisation d'un logiciel de gestion d'un aéroport (avec une ou plusieurs pistes).

La grosse difficulté de ce projet consiste à donner un *modèle* pour simuler le trafic d'un aéroport type Roissy. Pour le moment, nous supposons gérer un aéroport disposant d'une seule piste. On considère celle-ci comme un système évoluant au cours du temps. Se donner un modèle, c'est se donner un ensemble de paramètres permettant de décrire l'évolution de notre système suivant les contraintes que nous allons lui imposer. Concrètement, on souhaite suivant le flux d'arrivée et le flux de départ (qui sera défini de manière aléatoire) gérer au mieux le trafic aérien.

1.1 Définition du modèle

Afin de simplifier le problème, nous proposons de définir un modèle *discret* de simulation. Cela signifie que nous supposons que tous les changements dans l'état du système (c-à-d. les variables de notre système) se déroulent à un instant donné du temps. Le problème est donc de définir correctement les évènements que nous allons considérer comme étant des évènements de base de notre modèle. Par exemple, nous considérons un aéroport avec une seule piste. Il est donc évident qu'à un instant `t`, on ne peut autoriser qu'un décollage ou un atterrissage.

Nous allons tout d'abord définir les objets que nous allons utiliser. On peut classer les avions en deux catégories: ceux qui décollent et ceux qui atterrissent. Les premiers sont caractérisés par leur heure de décollage, les seconds par leur capacité à rester en vol (c-à-d par la quantité de carburant qui leur reste).

Nous allons donc considérer qu'à chaque instant `t`, un évènement (et un seul) sera traité : un décollage, ou un atterrissage ou éventuellement aucune action.

Notre modèle de simulation obéira aux règles suivantes :

- nous souhaitons faire décoller tous les avions à l'heure. Par conséquent, si un avion est en instance de décollage, et qu'un autre souhaite atterrir, la priorité sera donnée à l'avion au décollage. Le second continuera à tourner jusqu'au moment où la tour de contrôle lui donnera l'autorisation d'atterrissage (sous réserve d'une capacité de carburant suffisante) ;
- un avion qui n'a plus de carburant est de fait prioritaire sur tous les autres avions (ceux qui souhaitent décoller ou atterrir).

En résumé, nous imposons les règles suivantes par ordre de priorité :

- atterrissage d'urgence (manque de carburant) ;
- respect des horaires de décollage ;
- atterrissage si aucun événement de priorité supérieure.

Dans certaines situations, il est possible d'arriver à une saturation du trafic (trop d'avions en attente d'atterrissage). Dans ce cas, nous pouvons envisager la possibilité de détourner une partie du trafic (vers un autre aéroport).

1.2 Un exemple...

Nous noterons les décollages et les atterrissages, respectivement par D_i et A_i où i est un entier....Il est à noter qu'à chaque avion au décollage est associé une heure de décollage. De même, à chaque avion en instance d'atterrissage, on associe une certaine quantité de carburant (on ne précise pas l'unité). On suppose que la consommation de carburant est de 5 unités par unité de temps.

Temps	Evénements	Commentaires
t=0	$(D_1, t_d = 1)$	un décollage prévu à t=1
t=1		(1)
t=2	$(A_1, c = 10), (A_2, c = 20)$	(2)
t=3	$(A_2, c = 15)$	
t=4		Aucun événement
t=5	$(A_3, c = 10), (A_4, c = 15), (D_2, t_d = 7), (D_3 = t_d = 8)$	
t=6	$(A_4, c = 10), (D_2, t_d = 7), (D_3 = t_d = 8)$	A_3 a atterrit
t=7	$(A_4, c = 5), (D_3 = t_d = 8)$	D_2 a décollé à l'heure
t=8	$(D_3, t_d = 8)$	A_4 était prioritaire
t=9		D_3 a décollé en retard

(1) : l'avion D_1 a décollé. Plus d'événement en attente.

(2) : deux avions souhaitent atterrir avec respectivement 10 et 20 unités de carburant.

1.3 Analyse de l'exemple

L'étude de l'exemple précédent va nous permettre de définir les structures de données nécessaires à la résolution de notre problème. Tout d'abord, il faut noter qu'à chaque instant `t`, une multitude d'événements peuvent survenir : demande d'atterrissage, de décollage.... Notre problème est donc de décider quel événement devra être traité en priorité. Nous proposons de gérer ces événements à l'aide de deux listes chaînées : une pour les avions au décollage, et une pour ceux à l'atterrissage. Les sections suivantes vous expliqueront plus en détail la gestion de ces deux listes.

1.3.1 Avion à l'atterrissage

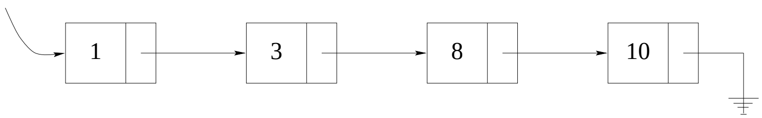
Dans notre hypothèse, nous avons décidé de donner la priorité aux avions au décollage. Par conséquent, si un avion est à l'atterrissage, et que la piste est occupée, il est nécessaire de le mettre "en attente". On peut donc envisager de définir un ensemble d'avions en attente. Mais d'une manière plus générale, cet ensemble va gérer tous les avions en instance d'atterrissage. Tout nouvel événement désignant un avion en instance d'atterrissage sera ajouté à cet ensemble.

Nous proposons de gérer cet ensemble d'événements à l'aide d'une liste chaînée triée. Le critère de tri étant la quantité de carburant disponible pour chaque avion. Ainsi, nous pouvons à tout moment connaître l'avion dont le carburant est au niveau le plus bas, et éventuellement procéder à un atterrissage d'urgence.

1.3.2 Avion au décollage

L'exemple précédant ne nous permet pas, au premier abord, de comprendre explicitement le processus de gestion des avions au décollage. Néanmoins, il apparait évident que ce processus s'apparente à la gestion d'une *file*. Nous rappelons qu'une file est un type particulier de liste, où les éléments sont insérés en *queue* et supprimés en *tête* (*i.e.* une file d'attente à un guichet, où "le premier arrivé est le premier servi"). Il nous parait utile de préciser un peu plus le processus de gestion des avions au décollage.

Par hypothèse, nous avions donné comme priorité de faire décoller les avions à l'heure. Nous allons supposer qu'il n'y a pas d'avions à l'atterrissage. Nous nous plaçons au temps `t=0` et nous supposons avoir la situation suivante :



La liste précédente représente des avions en instance décollage au temps `t=1, 3, 8, 10`. En fait, cette liste peut être parfaitement connue car nous pouvons imaginer que dans un système aussi complexe que celui d'un aéroport, il doit exister un processus de programmation des vols. Si tout se passe comme prévu, au temps `t=1`, le premier avion va décoller, au temps `t=3`, le second avion va décoller et ainsi de suite.

Mais supposons qu'au temps `t=4`, un nouvel avion arrive et désire décoller au temps `t=7`. Il doit donc se retrouver dans la liste des avions au décollage avant le vol décollant à `t=8`. Or il n'est pas possible de le faire "avancer" dans la queue ("premier arrivé, premier servi").

Par conséquent, il nous apparait nécessaire d'utiliser deux listes afin de gérer les avions au décollage :

- la première liste va gérer les avions à l'embarquement. Cela correspond aux avions dont l'heure de décollage `t` est encore "loin". Ces avions ne sont pas dans la seconde liste car leur insertion dans cette queue peut empêcher le décollage d'un autre avion dont l'heure de décollage serait plus proche ;
- la seconde liste, qui est une queue, gère les avions dont l'heure de décollage est "proche". En fait, on peut voir cette queue comme la queue des avions en bout de piste.

Au temps `t`, nous décidons qu'un avion ne pourra passer de la première à la seconde liste que si son heure de décollage est égal à `t+delai` (`delai` est un paramètre que nous pourrons modifier pour affiner notre modèle).

2 Description du travail à rendre

2.1 Langage de commande

Afin de tester le fonctionnement correct de votre logiciel, nous allons définir un "langage simple" pour exprimer les événements. Le programme que vous devez faire doit lire les événements dans un fichier écrit dans le format suivant :

- un vol sera codé par une chaine de 12 caractères. Le premier caractère indique s'il s'agit d'un atterrisage(`A`) ou d'un décollage (`D`).
  - Dans le cas d'un atterrissage, les cinq caractères suivants donnent des informations sur le carburant (trois caractères pour la quantité restante de carburant, et deux autres pour la consommation par unité de temps). On codera ensuite (sur deux caractères) l'heure prévue de l'atterrissage (la gestion de celle-ci est une option, si vous ne la traitez pas, on pourra mettre deux tirets (`--`)). Enfin, la désignation du vol remplit les quatres derniers caractères.
  - Dans le cas d'un décollage, l'information sur le carburant n'est pas indispensable, par conséquent, on mettra `-----` (cinq tirets), suivi de l'heure prévue du décollage (codée sur deux caractères), suivi de la désignation du vol (sur quatre caractères).
- une série d'événements relatifs à un instant t est représentée sur une même ligne.

Par exemple :

```
00 D-----01TWA8
02 A01005--AF01 A02005--AI02
05 A01005--LUF1 A01505--BA01 D-----06QU02 D-----07AA10
```

La première ligne représente l'événement :

> à `t=0`, vol `TWA8` décolle à `t=1`.

De même, la seconde ligne représente les événements :

> à `t=2`, vol `AF01` en arrivée avec `10` unités de carburant (coeff `5`), vol `AI02` en arrivée avec `20` unités de carburant (coeff `5`)

Le coefficient représente la quantité de carburant consommée par l'appareil entre chaque top d'horloge. Cette valeur est totalement arbitraire...Dans l'exemple précédent, les coefficients sont identiques, mais il est raisonnable de penser que cette valeur varie suivant les avions.
En plus de cette possibilité de lire des fichiers d'événements, le programme doit aussi laisser à l'utilisateur la possibilité de rentrer par la ligne de commande des vols à l'atterrissage ou au décollage.

2.1.1 Opérations de base

Le programme que vous devez faire doit être interactif, au sens où l'utilisateur doit pouvoir faire des requêtes et obtenir des réponses. On pourra donc prévoir un petit langage de commandes permettant à l'utilisateur de passer ses demandes au programme. Au lancement de `a.out` (par exemple), le programme affiche un *prompt*—nous avons choisi `SimAir:0>`—et se met en attente d'un ordre de la part de l'utilisateur. Nous affichons en plus du prompt la valeur de l'horloge du système (ici `0`).
La première commande utile sera la commande `h` fournissant de l'aide sur les commandes disponibles :

```
$a.out
SimAir:0> h
```

Commande	*Effet*
`h`	`Pour obtenir cette aide`
`open nom`	`Chargement du fichier`
`more dec`	`affichage de la liste des avions au decollage`
`more arr`	`affichage de la liste des avions a l arrivee`
`run`	`Saute de t à t+1.`
`q`	`Quitter`

Nous vous proposons également quelques *options* à implémenter :

```
app type  ajout d'un vol
          type :  0 - au decollage
          type :  1 - a l'arrivee
mode n    Selectionne un mode de fonctionnement
          - 0 :  mode continu
          - 1 :  mode pas a pas
run <n>   Saute de t à t+n.  Si n n'est pas preciser
          le saut est de 1 si on est en mode pas a pas
```

2.1.2 Exemple de session

Nous donnons un exemple de session à partir du fichier d'événements donné en exemple.

```
$a.out
SimAir:0> open exemple.airp
Fichier interprete :
t=0 : Vol TWA8 dec a t=1
t=2 : Vol AF01 arr carb 10 (coeff 5)
      Vol AI02 arr carb 20 (coeff 5)
t=5 : Vol LUF1 arr carb 10 (coeff 5)
      Vol BA01 arr carb 15 (coeff 5)
      Vol QU02 dec a t=6
      Vol AA10 dec a t=7
SimAir:0> app 0
       Heure : 3
       Designation : IB01
SimAir:0> mode 1
SimAir:0> run
          t=0 : No event
SimAir:1> run
          t=1 Vol TWA8 a decolle
SimAir:2> mode 0
SimAir:2> run
          t=2 Vol AF01 a l'arrivee
          t=3 Vol IB01 a decolle
          t=4 Vol AI02 a l'arrivee
          t=5 Vol LUF1 a l'arrivee
          t=6 Vol QU02 a decolle
          t=7 Vol BA01 a l'arrivee en urgence
          t=8 Vol AA10 a decolle avec retard
SimAir:9>
```

2.2 Structures de données

Nous proposons d'utiliser la structure de données suivante :

```
typedef struct evenement {
        information info ;
        struct evenement * suivant ;
        } StrEvenement ;

typedef StrEvenement * PtrEvenement ;
```

où le type `information` est un enregistrement qui contiendra des informations sur les vols (à vous de déterminer quelles sont les informations intéressantes...).

3 Améliorations et options

Les améliorations et les options ne sont pas obligatoires, mais constituent un *plus* pour l'évaluation finale de votre travail. Vous êtes invités à faire part de vos propositions aux enseignants pour savoir dans quelle mesure elles sont réellement pertinentes.

On peut envisager les options suivantes :

- la réalisation d'un affichage plus convivial ;
- les avions n'étant pas tous du même modèle, on peut considérer des consommations différentes en carburant. Par conséquent, la gestion de la liste des avions en instance d'atterrissage est complètement modifiée. On pourra envisager l'utilisation d'une liste circulaire ;
- l'hypothèse d'une seule piste est très restrictive. On peut envisager l'emploi de pistes supplémentaires. Il est donc important d'expliquer vos choix algorithmiques. Le but étant toujours de gérer au mieux le trafic (éviter les saturations) ;
- il serait algorithmiquement intéressant de gérer les horaires d'atterrisage ;
- il serait aussi intéressant d'établir des statistiques sur les différents vols gérés (retard, niveau de carburant restant dans les appareils...) ;
- il serait aussi utile d'écrire un module de génération aléatoire d'événements ;
- l'hypothèse de départ était d'essayer de faire décoller les avions à l'heure...Or il se peut que cette stratégie nous amène à des drames (avion en panne d'essence...). Peut-on alors élaborer un autre algorithme ?

Toutes ces options sont données à titre d'exemple. Nous vous laissons la plus grande liberté dans les options que vous voudrez ajouter. Cependant, Vous devrez justifier d'un point de vue algorithmique vos améliorations.