Durée 4h

# Tri à bulle parallèle

## 1 Présentation

Le travail demandé consiste à développer en langage C une version parallèle du tri à bulle, ainsi que quelques utilitaires en C et/ou en langage shell `bash` destinés à en évaluer les performances. De brèves questions accompagnent ces développements.

Le répertoire `/home/mauran/Public-retrait` contient

- une archive `f.tar` regroupant des fichiers qui peuvent être utiles aux différents développements à réaliser.
- un répertoire `Minisite`, qui rassemble l'essentiel des ressources et documents fournis pour l'enseignement de systèmes d'exploitation centralisés (voir le fichier `Lisez_moi.html` contenu dans ce répertoire).

Les réponses aux questions sont à rendre avec le code source des programmes C (voir la dernière section pour plus de précisions sur le rendu).

## 2 Le tri à bulle

Le tri à bulles est un algorithme de tri des éléments d'un tableau d'entiers. Son principe consiste à itérer un parcours du tableau en échangeant les éléments adjacents que l'on trouve mal rangés lors du parcours. L'algorithme sous forme séquentielle itérative peut s'écrire ainsi :

**Algorithm 1** Version itérative du tri à bulle

```
1: T[N] : tableau de N entiers à trier
2: for  i := N-1 to 1 do
3:    for  j := 0 to i-1 do
4:       if T[j+1] < T[j] then
5:          échanger T[j+1] et T[j]
6:       end if
7:    end for
8: end for
```

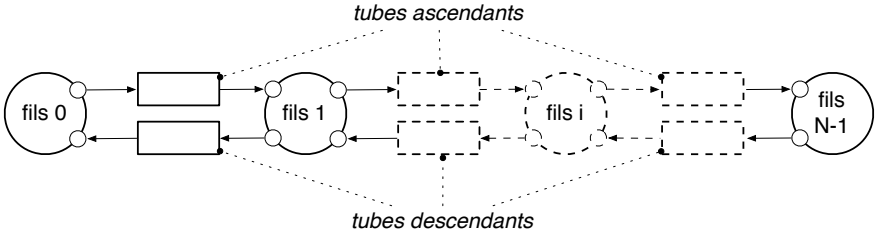
La boucle interne correspond à un parcours complet du tableau. À l'issue de ce parcours, le plus grand élément restant à trier se trouve nécessairement placé en fin de l'ensemble des éléments qui restaient à trier. En effet, partant du début du tableau, il sera nécessairement trouvé, puis sera échangé successivement avec tous les autres éléments restant à trier, jusqu'à atteindre la fin des éléments non encore triés. Ainsi à la fin de la première boucle, le plus grand élément se trouve placé en fin de tableau ; à l'issue de la seconde boucle, le deuxième plus grand élément se trouve placé en avant-dernière position, etc... Les boucles internes successives peuvent donc se terminer une position avant à chaque fois, d'où la progression décroissante de $i$, et la terminaison en $i - 1$ de la boucle interne.

# 3 Programmation système sous UNIX

On se propose de réaliser une version parallèle du tri à bulles, en s'appuyant sur les possibilités offertes par la bibliothèque d'appels système UNIX en matière de création de processus parallèles et de communication entre processus.

L'idée de base est de réaliser **en parallèle (en pipeline)** les itérations de la boucle interne. Pour cela, une chaîne de processus est construite à partir du processus père (principal), chaque élément de la chaîne étant en charge de gérer un élément du tableau. Les processus du pipeline utilisent des tubes (pipes) pour échanger les valeurs traitées. Plus précisément, chaque élément de la chaîne sera relié à son successeur par deux tubes : un tube ascendant pour transmettre les valeurs vers le successeur, et un tube descendant pour recevoir les valeurs du successeur. Ces tubes sont utilisés pour réaliser l'échange de valeurs entre éléments successifs du tableau.

L'architecture globale de l'application est ainsi la suivante :




Le processus principal (père) crée les différents processus et tubes de la chaîne. Il affecte une valeur entière à chacun des éléments, au fur et à mesure de la création. La valeur à affecter est lue dans un fichier dont le nom est fourni en paramètre lors du lancement de l'application. Ce fichier contient (uniquement) la suite des entiers à trier.

Chaque élément de la chaîne dispose
- de sa valeur
- et d'une copie de celle de son successeur immédiat.

Le comportement d'un élément $E$ de la chaîne (à l'exception du premier et du dernier) consiste à

1. **attendre** en sortie de son tube ascendant précédent sa nouvelle valeur, fournie par **son prédécesseur** $pred(E)$. Le fait de disposer d'une valeur sur ce tube signifie que $pred(E)$ vient de tester et de déterminer si $E$ et $pred(E)$ devaient échanger leurs valeurs. La valeur obtenue sera soit la valeur courante de l'élément $E$ (si les valeurs étaient bien rangées), soit l'ancienne valeur de $pred(E)$ (dans le cas contraire).
2. procéder à son tour à l'échange de valeurs avec son successeur. Pour cela,
   (a) l'élément $E$ compare (et échange au besoin) sa (nouvelle) valeur avec celle de sa copie de son successeur $succ(E)$.
   (b) puis il transmet à son successeur (via le tube ascendant qui le relie à $succ(E)$) la nouvelle valeur que ce successeur doit prendre, suite à l'échange (éventuel) de valeurs. La valeur transmise sera soit la valeur courante de $succ(E)$ (si les valeurs étaient bien rangées), soit l'ancienne valeur de $E$ (dans le cas contraire).
   (c) enfin, il acquitte l'échange initié par $pred(E)$ en transmettant sa valeur courante sur le tube descendant qui le relie à $pred(E)$.
3. puis, attendre et lire sur le tube descendant qui le relie à $succ(E)$ la valeur suivante prise par $succ(E)$. Le fait de disposer de cette valeur signifie que $succ(E)$ a procédé à son tour à un échange, et par conséquent (indirectement) qu'il a bien pris en compte la valeur qui lui avait été transmise dans l'étape précédente.

Ce comportement doit être répété jusqu'à ce que l'élément se trouve en fin de l'ensemble des éléments triés. Autrement dit, il doit être répété $k$ fois pour l'élément d'indice $N-1-k$. Il sera donc judicieux d'intégrer ce comportement dans une boucle contrôlant le nombre d'itérations en fonction de l'indice.

Le comportement du premier élément de la chaîne est similaire, sauf qu'il n'exécute pas l'étape (1) ci-dessus, n'ayant pas de prédécesseur. Il lance donc un nouvel échange dès que le second élément a acquitté l'échange précédent. Les parcours démarrent donc successivement à partir du premier élément, et s'exécutent en parallèle, en se suivant en pipeline.

De manière similaire, le dernier élément de la chaîne n'exécute pas les étapes (2.$b$) et (3), n'ayant pas de successeur.

**Question 1 (1 pt) :** Que se passerait-il si les étapes (3) et (2.$c$) étaient inversées ?

**Travail demandé (10 pts)**

Réaliser cette application en C, en utilisant les appels système UNIX de création et d'interaction entre processus. Le lancement de l'application en fournissant un paramètre correspondant au chemin d'accès au fichier contenant l'ensemble des entiers à trier.

Ce programme sera écrit dans un fichier `tri_bulle_par.c`.

**Conseils et précisions :**

- Contrôler l'affichage de sorte que l'invite du shell ne s'affiche qu'une fois l'application terminée.
- Prendre en compte les échecs lors des appels système (en particulier pour les appels relatifs à la création de processus et de tubes), et les traiter de manière appropriée.
- Ne laisser ouverts que les descripteurs (entrée/sorties de tubes) effectivement utilisés.
- Il sera sans doute judicieux de consulter le manuel en ligne de la commande `killall` **avant** de commencer le test de votre programme.
- Une archive `f.tar` est fournie. Cette archive contient les binaires
  - `Garnir`, qui permet de générer un fichier d'entiers à trier. Cet exécutable doit être lancé avec 2 paramètres : le nombre d'entiers du fichier, et le chemin d'accès au fichier généré.
  - `Voir`, qui permet de visualiser le contenu d'un fichier d'entiers. Cet exécutable doit être lancé avec 1 paramètre : le chemin d'accès au fichier
  - `taille.o`, qui implémente une fonction fournissant le nombre d'entiers d'un fichier d'entiers. Le fichier d'en-tête correspondant (`taille.h`) se trouve aussi dans l'archive.
  - `TBPL` qui est une implémentation parallèle du tri à bulle, fournie à des fins éventuelles d'évaluation et/ou de mise au point de votre code. Cet exécutable doit être lancé avec 1 paramètre : le chemin d'accès au fichier.
  - `TBS` qui est une implémentation séquentielle du tri à bulle, fournie à des fins éventuelles d'évaluation et/ou de mise au point de votre code. Cet exécutable doit être lancé avec 1 paramètre : le chemin d'accès au fichier.

**Question 2 (2 pts)** Quel est l'effet de l'envoi d'un signal `SIGKILL` à l'un des processus de la chaîne ? En particulier

a) Est-ce que cela influence le résultat final ? Si oui, en quoi ? Si non, pourquoi ?
b) Y aura-t-il des messages d'erreur ? Si oui, lesquels, et pourquoi ? Si non, pourquoi ?
c) Tous les processus se termineront-ils ? Si oui, pourquoi ? Si non, précisez lesquels se termineront.

# 4 Evaluation

On se propose maintenant d'évaluer les temps d'exécution des différents binaires réalisés ou utilisés, pour différentes tailles du tableau.

L'affichage/le document attendu est le suivant : pour chaque exécutable, son nom, suivi d' une série de lignes consécutives, chaque ligne indiquant : la taille du tableau, le temps d'exécution, le pourcentage processeur utilisé.

Cette évaluation pourra être réalisée par un script de test en langage shell `bash` (**bonus : 2 points**) ou par une série de commandes saisies à la console (dans ce cas, donner les commandes avec leur résultat)

**Conseils et précisions :**

- la commande `time cde` fournit les mesures nécessaires, pour une commande `cde` passée en paramètre. Le manuel en ligne donne les détails permettant d'obtenir les informations voulues, au format voulu.
  *Attention* : la commande documentée dans le manuel en ligne est la commande `/usr/bin/time`, qui est celle qui doit être utilisée, mais qui est différente de la commande `time` interne du `bash`.
- Il vous appartient de fixer les tailles des fichiers de test pour que les résultats soient lisibles, significatifs et produits avant la fin de l'épreuve.

**Travail demandé**

**Tâche 1 (5 pts)** Développer en C une version séquentielle du tri à bulle, afin de procéder à une évaluation comparative avec la version parallèle. Afin que la comparaison soit significative, il s'agira de dériver directement la version séquentielle à partir de la version parallèle, c'est-à-dire que l'architecture de la version parallèle devra être conservée, mais à chaque instant, au plus un processus pourra progresser.

Pour cela, le plus simple sera que le premier élément lance un parcours, puis attende la fin de ce parcours avant de lancer le parcours suivant. La fin du parcours courant pourra être signalée au premier élément par le dernier élément restant à traiter dans le parcours. Ce programme sera écrit dans un fichier `tri_bulle_seq.c`. Dans le cas où cette version ne serait pas réalisée, l'exécutable `TBS` pourra être utilisé pour l'évaluation

**Tâche 2 (4 pts)** Développer en C une version des outils `Garnir` et `Voir` Ces programmes seront écrits dans les fichiers `Garnir.c` et `Voir.c`.

*Remarque* : vous pouvez utiliser `sscanf` (`man sscanf`) pour ranger un nombre entier stocké dans une chaîne (par exemple un des arguments du `main`) dans une variable de type `int`.

**Tâche 3 (1 pt)** Développer en C une version de la fonction `taille(..)`, **en vous appuyant sur la primitive** `fstat` (`man fstat`). Ce programme sera écrit dans un fichier `taille.c`.

**Tâche 4 (2 pts)** Réaliser l'évaluation proprement dite des différentes versions, pour différentes tailles de tableau (fichiers de test).

**Question 3 : commentaire des résultats (2 pts)**

1. existe-t-il des différences significatives de performances entre la version séquentielle et les versions parallèles ? Pourquoi ?
2. Commentez le taux d'utilisation du processeur, ainsi que son évolution en fonction de la taille du tableau pour la version séquentielle et pour la version parallèle.

**Question 4 : limites matérielles des différentes versions (1 pt)** Il existe des limites liées à l'environnement d'exécution (architecture matérielle, structures de données système...) qui bornent de fait le nombre d'éléments triés. Indiquez la nature des limites qui pourront être rencontrées en premier

a dans le cas de la version séquentielle `TBS` ;

b dans le cas de la version parallèle .

# 5 Rendu du BE

Vous constituerez un fichier d'archive tar comportant les documents suivants

- les fichiers sources C correspondant aux différentes tâches ;
- un fichier texte, contenant les sorties du script d'évaluation, ou le résultat des différentes commandes lancées, avec le texte du script ou des commandes (Tâche 4) ;
- un fichier texte, comportant les réponses aux questions 1, 2, 3, 4.

**Le nom de l'archive devra comporter votre nom.**

A l'issue de la séance, vous déposerez cette archive dans le répertoire `/home/mauran/Public-depot`

Vous pourrez aussi de plus envoyer cette archive par mail à l'adresse `mauran@enseeiht.fr`. Avant de partir, assurez-vous de la bonne réception de votre envoi auprès de son destinataire.