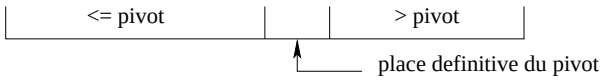
TP processus standards *vs* processus légers

Buts du TP :
- voir les façons de partager des données entre :
  - des processus *standards*
  - des processus légers (ou activités ou *threads*)
- manipuler les threads de la norme POSIX
- comparer les coûts d'exécution des processus standards et légers

Pour atteindre ces buts, nous nous plaçons dans le cadre de la programmation de l'algorithme du tri rapide.

1 Rappel sur le tri rapide

Ce tri fait partie des tris par dichotomie : on partage la liste à trier en deux sous-listes, telles que les éléments de la première soient inférieurs aux éléments de la seconde. On recommence jusqu'à avoir des sous-listes réduites à un seul élément.
Pour partager la liste en deux sous-listes, on chosit une des valeurs de la liste (ici celle du premier élément) que l'on utilise comme *pivot* : on construit une sous-liste gauche avec les éléments dont la valeur est inférieure à celle du pivot et une sous-liste droite avec les éléments supérieurs au pivot. Comme on fait cette construction directement sur le tableau qui représente la liste, la frontière entre les deux sous-listes donne la place définitive du pivot.



Vous disposez d'une fonction `tab_placer(tab, debut, fin, pivot)` qui répartit les éléments du tableau `tab` compris entre `debut` et `fin` en fonction du pivot `tab[debut]` et rend comme résultat `pivot`, l'indice où le pivot à été placé. La procédure de tri d'un tableau s'écrit alors :

```
tri_rapide( tab, debut, fin )
{
   int pivot ;
   if( debut < fin )
   {
       tab_placer( tab, debut , fin , &pivot) ;
       tri_rapide( tab, debut , pivot-1 );
       tri_rapide( tab , pivot+1 , fin );
   }
}
```

2 Téléchargements

Vous pouvez récupérer

- la définition et les fonctions de gestion du TDA tableau dans les fichiers sources `tab.{c,h}` : remplissages du tableau, affichage, placement du pivot...
- le code source d'un programme de tri rapide standard utilisant le TDA tableau
- un Makefile vous facilitant, en principe, la compilation de votre TP.

à partir

- de ma page *Enseignements* à
  `http ://www-ic2.univ-lemans.fr/ jacob/enseignement.html`
- du serveur de l'IUP à
  `/info/tmp/AnnexesTPM1_UE1_07/TP_Tri_Rapide/TP_tri_rapide.tar.gz`

3 Sujet du TP

Reprendre le programme `tri_standard.c` en parallélisant le tri de la partie gauche et le tri de la partie droite du tableau.

3.1 Réalisation avec des processus

Chaque appel de la fonction `tri_rapide` est réalisé par un processus créé par un `fork`. Le tableau est partagé par tous les processus par un segment de mémoire partagée. *Nom du programme* : `tri_fork.c`

3.2 Réalisation avec des *threads*

Chaque appel récursif de la fonction `tri_rapide` est réalisé par une activité (un thread). *Nom du programme* : `tri_pthread.c`

3.3 Contenu du rapport

Le rapport de ce TP devra comprendre

1. Les codes sources des programmes `tri_fork.c` et `tri_pthread.c`
2. Une comparaison et une analyse des résultats des 2 programmes pour un tri d'un tableau de 1000 éléments. Qu'est ce qui, d'après vous, explique les différences d'exécution des deux programmes ? Ces différences peuvent être, par exemple, dans les temps CPU ou simplement le fait que ça marche dans un cas et pas dans l'autre.