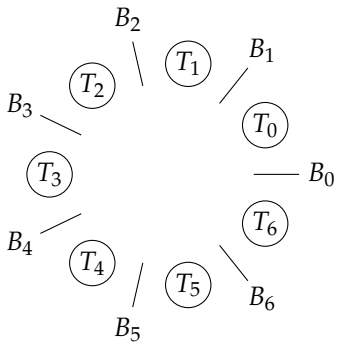
Le déjeuner des bavards et l'interblocage

On considère une tablée composée de n bavards $T_0 \dots T_{n-1}$ et de seulement n baguettes $B_0 \dots B_{n-1}$, parfaitement entrelacés. Il faut deux baguettes pour manger. Chaque bavard exécute l'algorithme suivant, qu'il répète p fois :



— parler pendant t_a secondes
— prendre la baguette à sa gauche
— parler pendant t_b secondes
— prendre la baguette à sa droite
— manger (et parler) pendant t_m secondes
— reposer les deux baguettes.

Ici, « prendre une baguette » signifie : attendre éventuellement qu'elle soit disponible, puisqu'à tout moment une baguette ne peut être utilisée que par un des deux bavards voisins.

On veut écrire un programme avec la syntaxe suivante :

```
bavards t n p
```

où t est le délai maximum d'attente en millisecondes (t_a, t_b et t_m ne sont pas fixes et doivent être des valeurs aléatoires entre 0 et t ms, vous utiliserez `rand_r` pour générer les temps aléatoires et `usleep` pour réaliser les attentes), n est le nombre de bavards et p le nombre de répétitions effectuées par chaque bavard. Votre programme doit utiliser un thread par bavard et garantir la bonne synchronisation de l'utilisation des baguettes. Le programme principal doit créer les threads, attendre qu'ils se terminent, puis afficher le message « fin du repas ».

1. Écrivez ce programme. Chaque thread doit afficher un message contenant son indice avant de « parler » ou « manger ».
2. Observez plusieurs exécutions de votre programme (en modifiant au besoin la valeur initiale du paramètre *seed* dans chaque thread). Votre programme se termine-t-il toujours en affichant le message final ? Si c'est le cas, fixez $t = 0$ Testez jusqu'à ce que votre programme se bloque (l'algorithme ci-dessus est incorrect, il est toujours possible de bloquer le programme).
3. Expliquez pourquoi le programme se bloque. Puis corrigez l'algorithme proposé plus haut pour que votre programme arrive toujours à se terminer (indication : il suffit de modifier le comportement d'un seul bavard, par exemple T_{n-1}).

   Pour réaliser cette version, utilisez la compilation conditionnelle avec l'option `-DOK`.