Boîte aux lettres

Il arrive souvent que des threads (les *émetteurs*) aient à communiquer, via un canal limité, des informations de manière fiable à d'autres threads (les *récepteurs*). La fiabilité signifie ici qu'un émetteur qui place une information dans le canal est certain qu'elle sera prise en compte par un récepteur et qu'elle ne sera pas ignorée ou prise en compte plusieurs fois.

Afin de comprendre la synchronisation nécessaire, avec les conditions POSIX, pour ce motif appelé la « boîte aux lettres », on demande de rédiger un programme avec la syntaxe suivante :

`mbox` t p n_e n_r

où t (en millisecondes) est la borne supérieure des temps aléatoires, p est le nombre de messages envoyés par chacun des n_e threads émetteurs et n_r est le nombre de threads récepteurs.

Chaque thread émetteur doit effectuer p fois la boucle suivante : attendre un délai aléatoire borné par t, puis attendre que la boîte aux lettres soit libre, y déposer un message (que l'on matérialisera par l'affectation de 1 à une variable entière partagée) et recommencer.

Chaque thread récepteur doit, également dans une boucle, attendre que la boîte aux lettres contienne un message puis lire et traiter ce message. Le traitement se limitera ici à incrémenter un compteur de messages reçus par ce thread et attendre un délai aléatoire borné par t.

Pour que ce système fonctionne, il faut qu'il sache s'arrêter : les threads récepteurs doivent détecter que tous les threads émetteurs ont fini d'émettre. Pour ce faire, on propose que la boîte aux lettres conserve le nombre d'émetteurs actifs : initialisé à n_e par le thread principal, ce compteur est décrémenté à chaque fois qu'un émetteur se termine. Lorsque ce compteur est à 0 et qu'il n'y plus de message en attente de traitement dans la boîte aux lettres, chaque récepteur peut alors afficher le nombre de messages reçus et se terminer.

Rédigez le programme avec la syntaxe ci-dessus avec les conditions POSIX. Utilisez `rand_r` pour générer les temps aléatoires et `usleep` pour réaliser les attentes. Bien sûr, votre programme ne doit pas comporter de variable globale.

Par exemple :

```
> ./mbox 1 20 3 2                  # attente de 1 ms max, 5 thread générés
T3 : nb recus = 34                 # le premier récepteur a reçu plus de messages
T4 : nb recus = 26                 # ... que le deuxième
total = 60, attendu = 60           # tout va bien : on en a reçu autant qu'attendu
```