Compte rendue
Projet de Simulation
Supermarché
2014-2015

Auteur: **Basri Mohamed**

Le Projet contient :

- 3 fichiers header.
- 4 fichiers « .c ».
- 1 fichier makefile.
- 1 fichier gplt.

A la compilation on obtient :

- 4 fichier « .o »
- 1 fichier exécutable

A l'exécution on obtient :

- 1 fichier « .data »
- 1 fichier PDF

Sommaire

I. Modélisation sur papier :

1. Evènement

Les Evènement utilisé sont les suivant : AC1, AC2, DBC, DS, FS.

AC1 (arrivé client dans une file de type blocs) : on considère l'arrivé d'un client comme étant le passage en caisse d'un client son arrivé dans une file d'attente de type blocs, on ne prend pas en compte le temps passé à faire les courses.

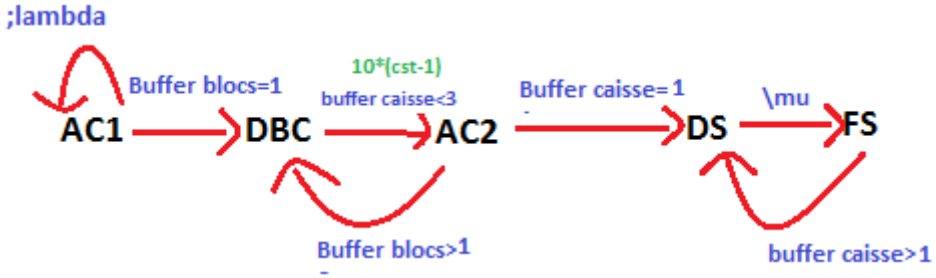
AC2 (arrivé client dans une file de type caisse) : le client choisi la caisse avec le moins de monde, la file d'attente de type blocs est bloqué si toutes les files d'attente de type caisse sont de tailles 3.

DBC (Déplacement d'un Blocs a une Caisse) : si la file d'attente de type blocs n'est pas bloqué, le service de cette file (qui n'en n'est pas vraiment un) consiste à faire le choix sur une caisse est à ce déplacé vers celle-ci.

DS (Debut de service- arrivé en caisse) : selon l'énoncé il nous faut en moyenne 100 unités de temps pour servir 1 client.

FS (Fin de Service) : la sortie du client dans notre système de modélisation.

Nos évènements suivent le schéma suivant :

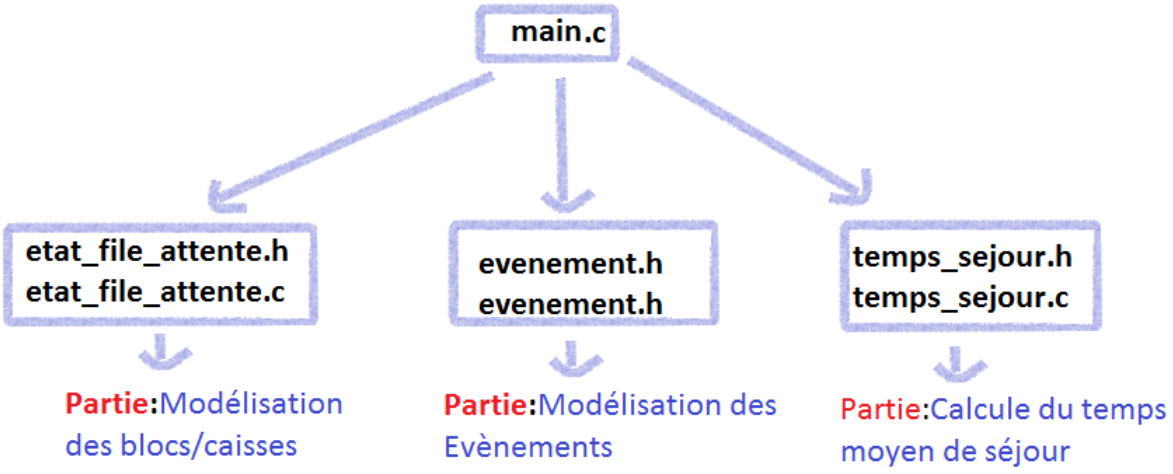


2. Les variables utilisés :

Il nous ait demandé de mesuré le temps de séjour d'un client dans le système, on se doit de sauvegarder les temps d'arrivé de chaque client, il nous faut aussi sauvegarder le nombre de client servi pour le calcul de cette moyenne.

II. L'implémentation

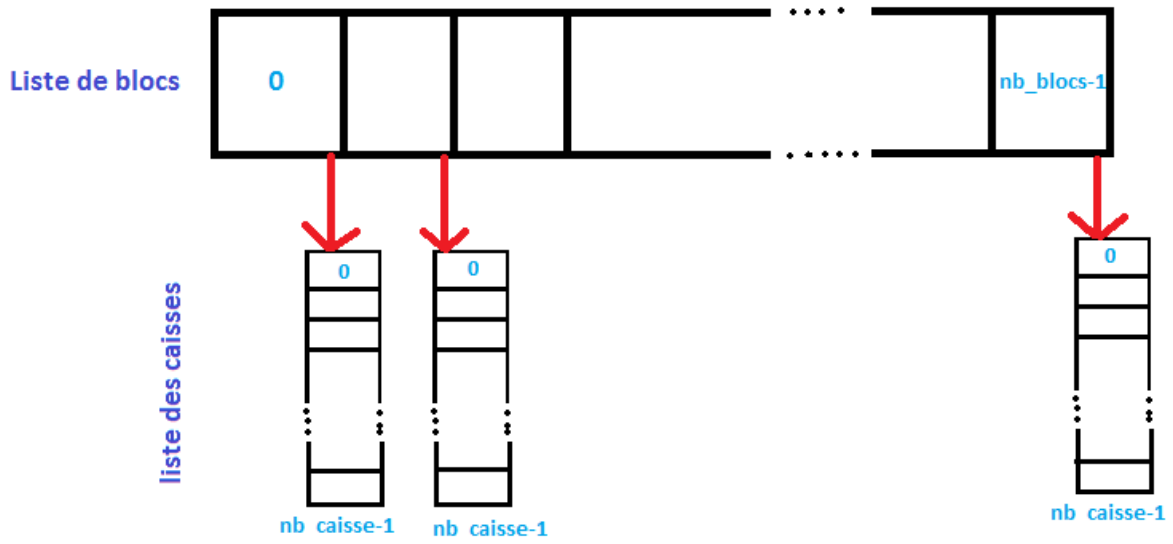
1. Structure du Programme



Nous allons détailler chaque partie.

2. Modélisation des blocs/caisses

On les a modélisé via une structure de donnée, visuellement parlant ça nous donne ceci :



On peut accéder directement a une entité liste_de_bloc[i].liste_de_caisse[j] (pas besoin d'une liste chainée) cela est dû au faite que les blocs/caisses sont indexé de 0 à N, N étant une valeur entière définit en début du programme.

Sur chaque entité on stocke le nombre actuel de client, le nombre de client servi, temps moyen de séjour, temps d'arrivé des clients dans la file (une liste chainé sur une autre structure)

Nombre actuel de client : va principalement nous servir pour connaitre si une file de type caisse est pleine

Nombre de client servi : on est obligé de stoker cette information pour le calcul du temps moyen de séjour d'un client

Temps moyen de séjour : pour chaque file d'attente on calcule sont temps moyen de séjour=>on déduit le temps moyen de séjour de tout le système

Temps d'arrivé des clients dans la file : on stocke ces informations dans une liste chainée, l'arrivé d'un client consiste en l'ajout vers la fin de la liste, la sortie d'un client consiste en la lecture du début de la liste afin de calculé le temps de séjour de ce client dans cette file d'attente, puis on avance d'un pas dans la liste chainée (en libérant la mémoire), avec cette méthode on utilise que la mémoire dans on a besoin, ***nous détaillerons ceci plus en bas (II-4-Calcule du temps moyen de séjour)***

3. Modélisation des Evènements

Dans chaque évènement on stocke le type de cet évènement, le numéro de bloc/caisse sur lequel on ait

Type d'évènement : cela nous permet de déterminer l'action à faire

Numéro de bloc/caisse : avec ces informations on accède directement à l'entité blocs/caisse sur lequel se trouve le client, exemple d'utilisation :

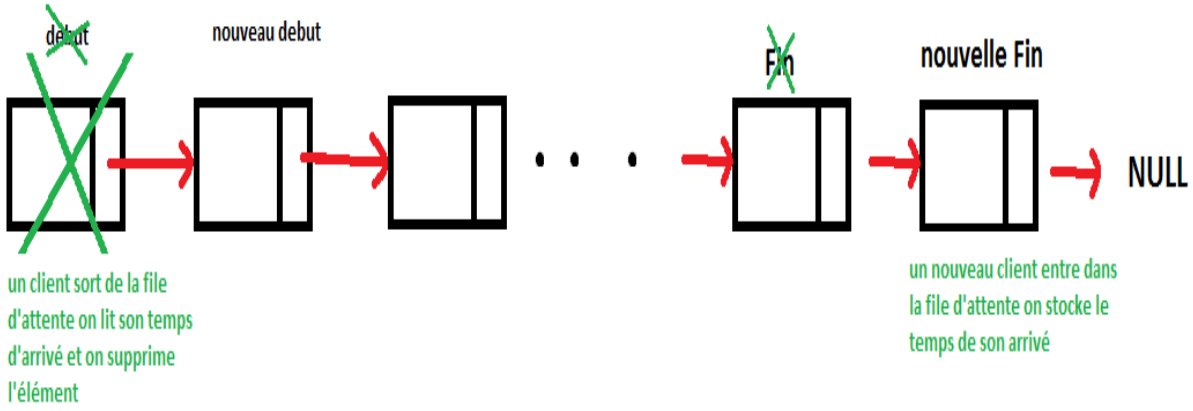
-déplacement d'un bloc vers une caisse, on doit connaitre le bloc ou se trouve le client pour connaitre si il y'a une place de libre dans une des caisses.
- vu que pour chaque file on calcule le temps moyen de séjour, avec ses information on connait quel est l'entité ou il faudra mettre à jour son temps moyen de séjour

Dans certain cas ses informations nous sont futile, tels que pour ajouter un évènement de type AC1 (arrivé de client dans un bloc) on ne connait pas encore le numéro de bloc/caisse, dans ce cas-là on ajoute l'évènement et on initialise les numéros de bloc/caisse a -1.

4. Calcule du temps moyen de séjour

Afin de calculé le temps de séjour dans le système, on calcul se dernier dans chaque file d'attente, pour cela on stocke le temps d'arrivé de chaque client qui entre dans une file d'attente, à sa sortie de la file on lit son temps d'arrivé et on calcul son temps de séjour, connaissant le temps moyen de séjour de chaque file on déduit le temps moyen de séjour dans toute le système sans oublié le durée du déplacement entre un bloc est une caisse.

Pour stocké les temps d'arrivé on utilise une liste chainé pour chaque file d'attente (meilleur solution que si on les avait stocké dans un tableau vu que dans un bloc on peut accueillir une infinité de client), visuellement ça donne ceci :



III. Travail à faire

Expliquer pourquoi il est suffisant de ne simuler qu'un seul bloc.

Le choix d'un bloc se fait suivant une loi uniforme ce qui signifie que sur chaque blocs il y'aura un nombre égal de client qui sont rentrés et vu que tous les blocs on le même nombre de caisse et le débit de service de ses caisse sont tous les mêmes alors on obtiendra le même résultat sur chaque blocs, ça revient au même que de faire une simulation sur un seul bloc.

Donner une approximation pour chaque valeur de B de la valeur de lambda limite pour la stabilité.

On part du principe que d'un même bloc plusieurs déplacements (d'un bloc vers une caisse) peuvent se faire en parallèles (dépend du nombre de place disponible)

Pour que le système soit stable il faut que le débit d'arrivé soit inférieur au débit de sortie

-**B=1** (1 bloc de 60 caisses) :

$$\frac{\lambda}{1} < 60 * 0.01$$
$$\lambda < 0.6$$

sur notre simulation on constate que notre système devient instable a partir d''un debit > 0.59

-**B=5** (5 blocs de 14 caisses) :

$$\frac{\lambda}{5} < 12 * 0.01$$
$$\lambda < 0.6$$

sur notre simulation on constate que notre système devient instable a partir d''un debit > 0.57

-**B=15** (15 blocs de 4 caisses) :

$$\frac{\lambda}{15} < 4 * 0.01$$
$$\lambda < 0.6$$

sur notre simulation on constate que notre système devient instable a partir d''un debit > 0.56

-**B=30** (30 blocs de 2 caisses) :

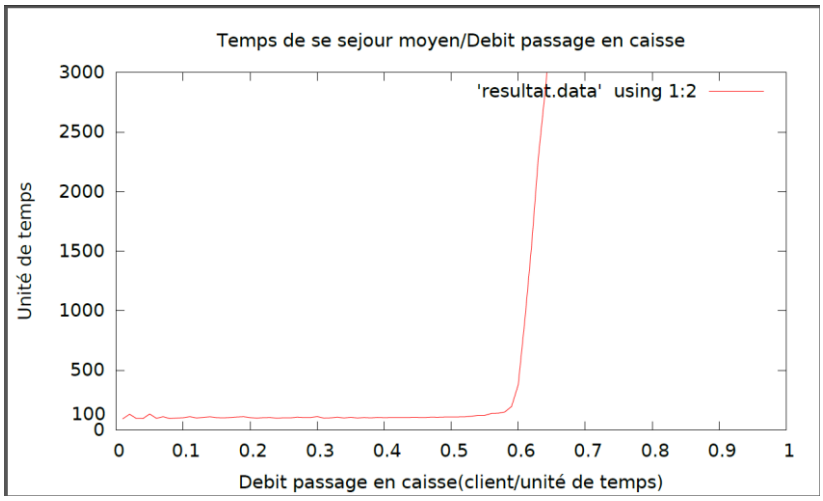
$\lambda < 0.6$ sur notre simulation on constate que notre système devient instable a partir d''un debit > 0.59

-**B=60** (60 blocs de 1 caisse) :

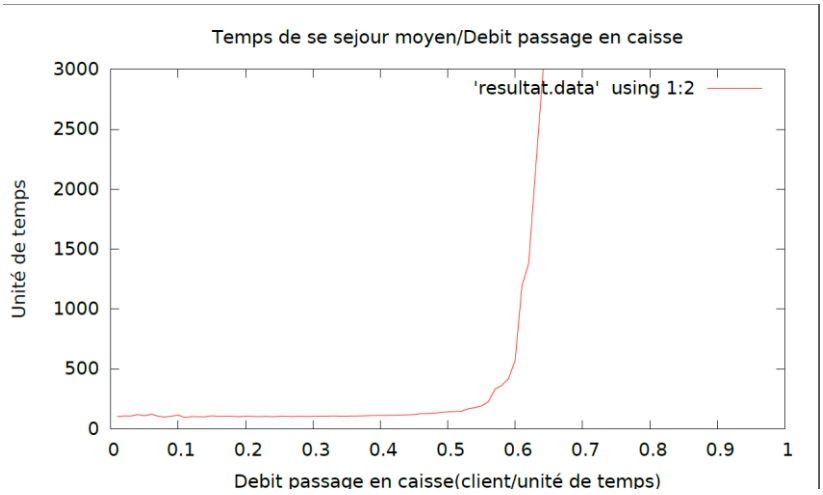
$\lambda < 0.6$ sur notre simulation on constate que notre système devient instable a partir d''un debit > 0.54

On obtient tout le temps le même résultat est c'est normal vu que quel que soit le nombre de bloc le nombre de caisse au total reste constant et pour que le système soit stable le débit d'entré ne doit pas dépassé le débit de sortie de toutes les caisses réunies

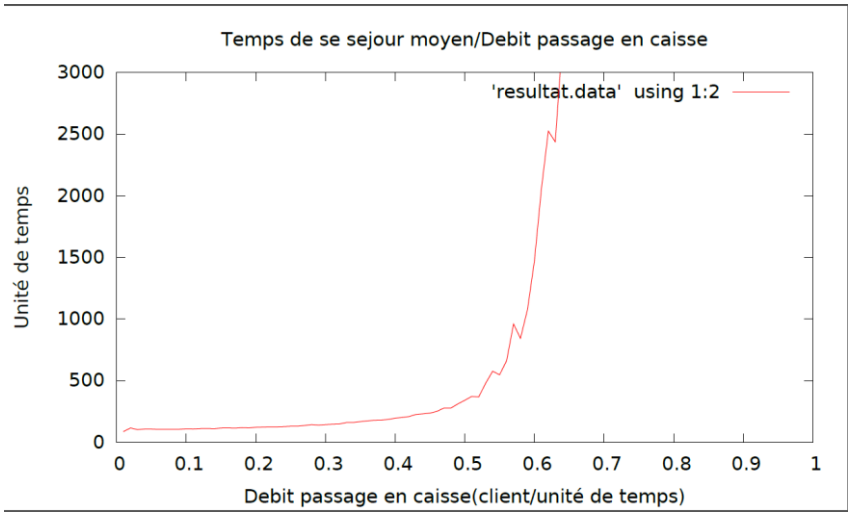
Pour chaque valeur de B, donner une courbe avec lambda en abscisse et R en ordonnée.



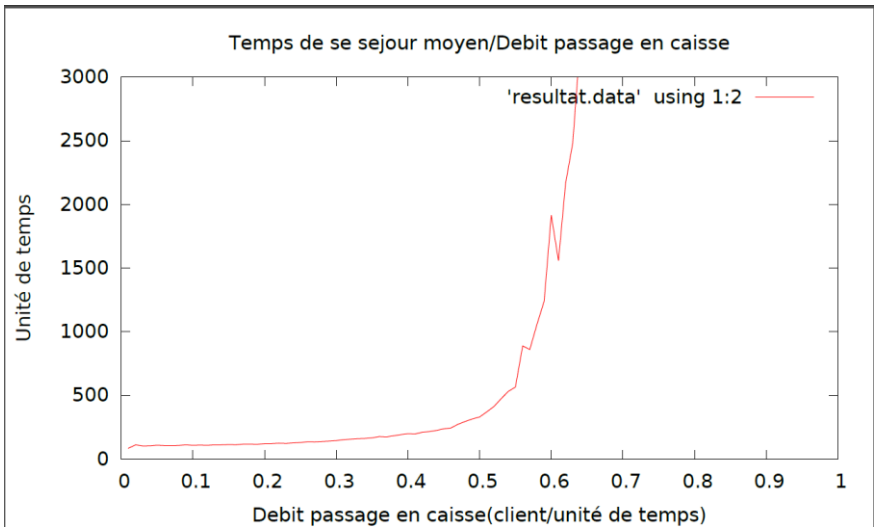
1 Blocs de 60 caisses



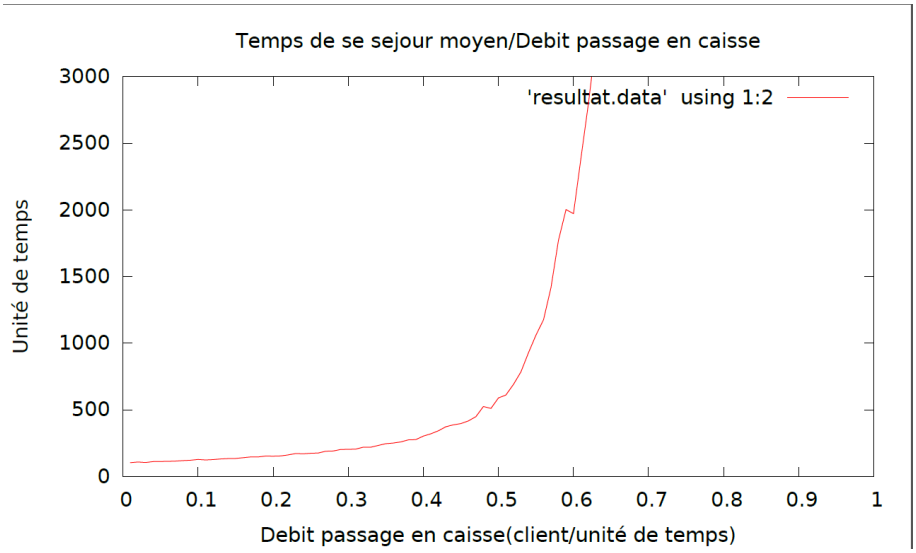
5 Blocs de 12 caisses



15 Blocs de 4 caisses

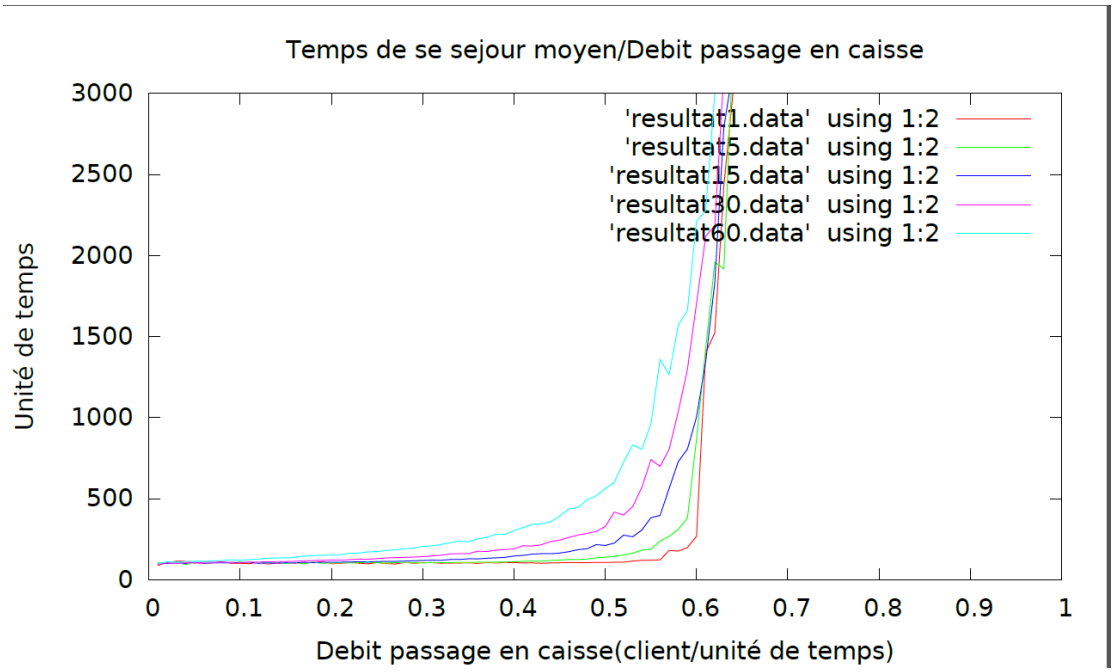


30 Blocs de 2 caisses



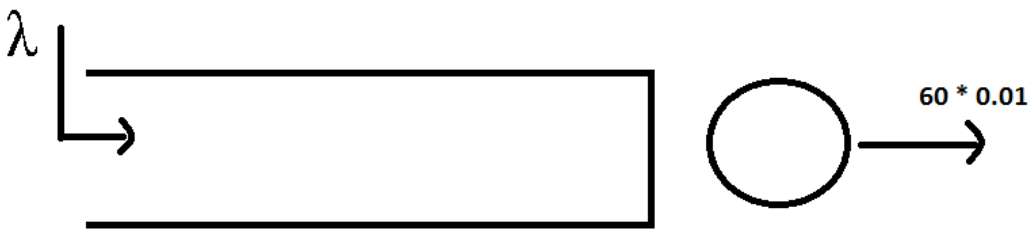
60 Blocs de 1 caisse

Superposer les cinq courbes des différentes valeurs de B. Les courbes se croisent-elles ? Expliquez pourquoi.



Superposition des 5 courbes présentées en haut

On remarque bien que les 5 courbe se croisent et commence à divergé au même moment, est c'est normal vu que quel que soit le nombre de blocs le nombre de caisse au total reste constant => Débit de sortie constant, notre système se résume au schéma suivant :



Comparer les résultats de simulation et les files M/M/m.

On peut dire que notre système ressemble a une file de type M/M/m surtout si les conditions suivantes sont respectés :

Nombre de blocs =1

Nombre de caisse = 60

Nombre maximum de client dans une file d'attente de type caisse = 1

Sous c'est contrainte on obtient une file M/M/m.

VI. Remarque et compréhension

On ***suppose*** que notre *unité de temps* est la *seconde*

➔ Une fois en caisse un client passe 100 secondes à se faire servir (ce qui est compréhensible pour une file d'attente d'un supermarché, ce qui explique mon choix de la seconde comme unité de temps)

1. Condition d'arrêt de notre simulation

On est censé lancé notre simulation avec différant débit d'arrivé de client dans un bloc, on risque de rencontré souvent des cas ou notre temps moyen de séjour diverge, on ne peut pas mettre comme condition d'arrêt la stabilité du temps moyen de séjour.

Condition d'arrêt choisie : on simule l'équivalant de 24 heures (86 400 secondes (=unité de temps))

2. Borne inferieur du temps moyen de séjour

Vu qu'un client passe en moyenne 100 unité de temps à se faire servir, en risque pas de croisé un temps moyen de séjour inferieur a 100, sauf si notre variable aléatoire (suivant une loi exponentiel), n'est pas fiable