待ち行列シミュレーション

この資料の到達目標

- 待ち行列の数理について理解する
  - システム内のジョブ総数
  - 待ち行列の長さ

  } 待ち行列で重要な量

- 待ち行列の定常状態の意味を理解する
  - 状態遷移
  - 定常確率

  } 「確率」を使って，待ち行列の振る舞いをとらえる

内容

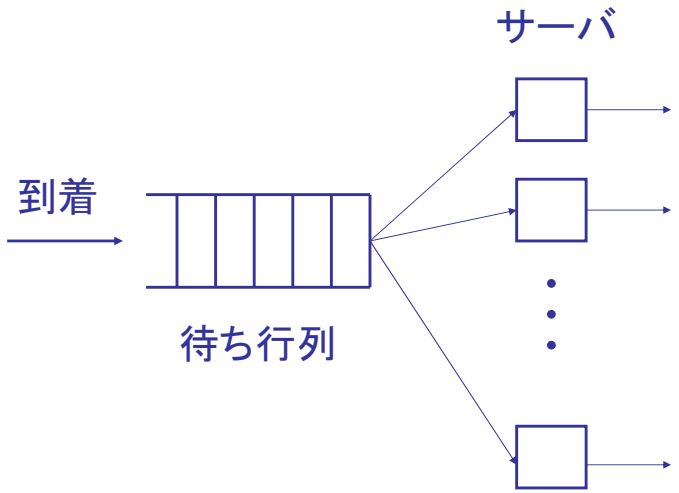
- 基本的な待ち行列 （M/M/1/1 待ち行列）による説明
  - 待ち行列の長さ
  - システム内のジョブ総数
  - 状態遷移
  - 定常状態，定常確率
- M/M/1 待ち行列
  - M/M/1/1 との違い
- M/M/S
  - 「M/M/S」の意味

待ち行列とは

スタックとキュー

- スタック
  - データの挿入と取り出しの両方を列の一方の端から行う
- キュー
  - 一方の端から挿入を，もう一方の端から取り出しを行う
  - 取り出されるのは最も古いデータ
  - 最初に入れたデータが最初に取り出される
  - **FIFO**(first-in-first-out，先入れ先出し)と呼ぶ

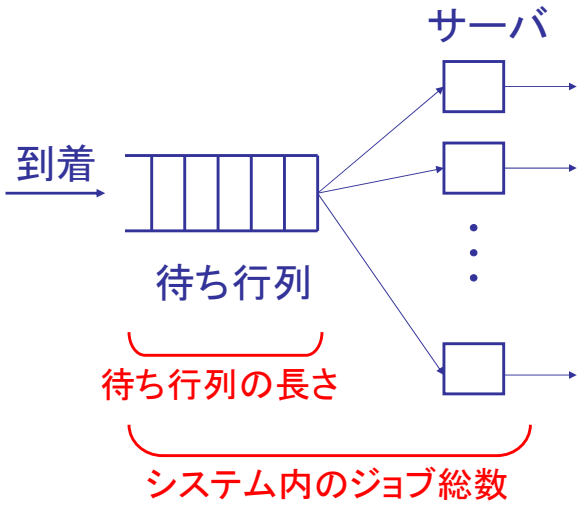
待ち行列



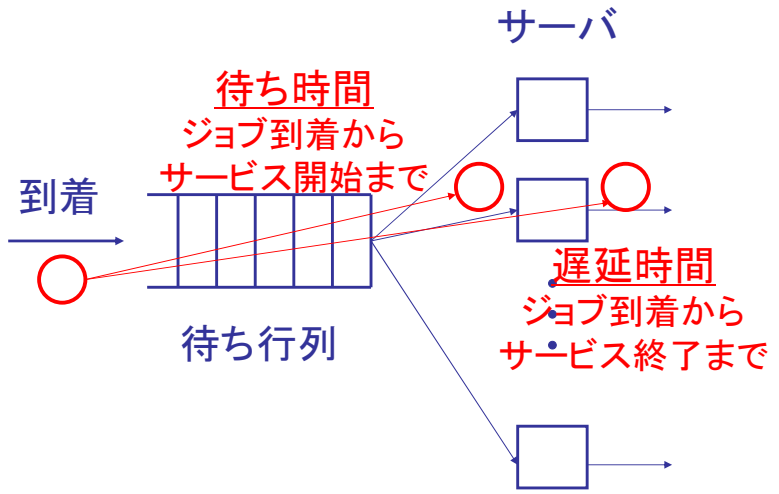
待ち行列

- 処理を受けるために順番待ちをする人がなす列
  - 銀行の窓口や入場券売り場など

待ち行列の長さ／システム内のジョブ総数



遅延時間／待ち時間



遅延時間とシステム内のジョブ総数の関係

$\lambda D = N$

D: 「遅延時間」の平均

N: 「システム内のジョブ総数」の平均

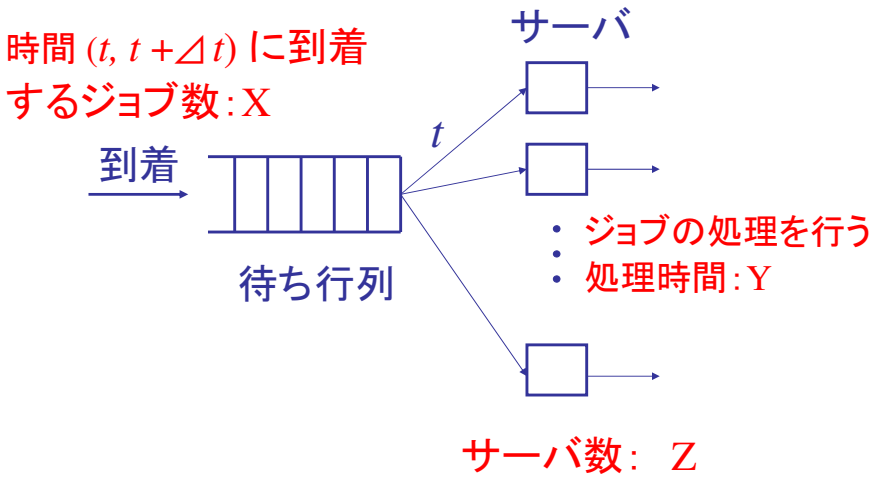
$\lambda Dw = Nw$

Dw: 「待ち時間」の平均

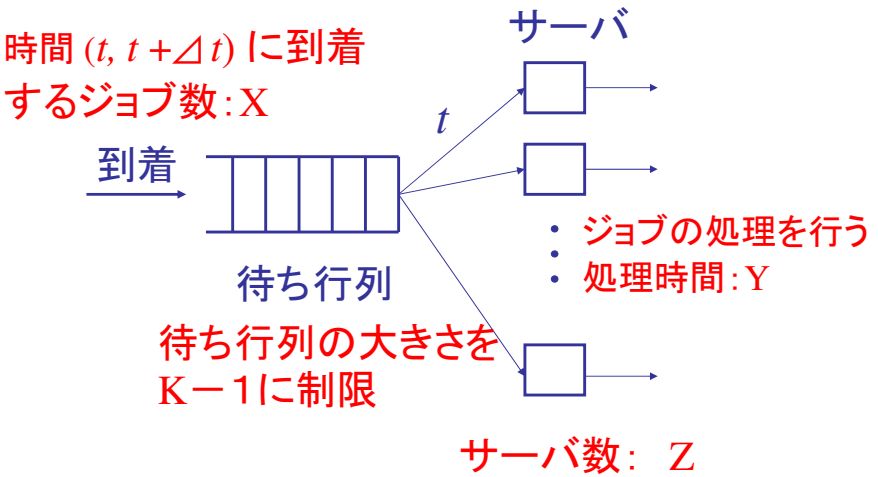
Nw: 「待ち行列の長さ」の平均

以下，システム内のジョブ総数，待ち行列の長さを考える

ケンドール記法 X/Y/Z



ケンドール記法 X/Y/Z/K



「待ち行列の大きさ制限」

- 待ち行列の大きさに限りがある
  - 待ち行列の最大長がＫ－１に制限されるとき，システム内のジョブ総数は　Ｋ　に制限される
- 特に　Ｋ＝０　ならば
  - すでにサーバが他のジョブを処理中
    - 到着したジョブは棄却される（待ち行列に入らない）
  - サーバがジョブを処理していない
    - 到着したジョブは直ちに処理される

ケンドール記法

X/Y/Z/K

X:　到着過程
Y:　処理時間分布
Z:　サーバ数
K:　待ち行列の大きさ制限
（待ち行列の最大長：Ｋ－１）

X/Y/Z

待ち行列の大きさに制限無し

待ち行列解析

待ち行列解析

- 待ち行列の長さはいくらか
- システム内のジョブ総数はいくらか
- ジョブの遅延時間はいくらか
- ジョブの待ち時間はいくらか

手順

- 待ち行列の長さ，システム内のジョブ総数の分布を求めたい
- 「システムの状態」と「状態遷移」を考えて，待ち行列の長さ，システム内のジョブ総数を算出する
  - システムの状態：$P_0, P_1, P_2, \cdots$
    （添字は，システム内のジョブ総数）
- 定常状態で考える
  - システム内のジョブ総数は，初期状態の影響を受ける
  - t→∞では，システムの状態は定常確率に漸近する
    （初期状態を無視できる）

方針

X: 到着過程
→ポアソン分布のみを考える
Y: 処理時間分布
→指数分布のみを考える
Z: サーバ数
→最初は1とする．あとで増やす
K: 待ち行列の大きさ制限
→最初は1とする．あとで増やす

平均到着率

- 単位時間に到着するジョブの平均値
- 待ち行列に加わろうとするジョブのやってくる頻度

到着率 λ のポアソン分布

- ジョブの到着がランダム
- 「時間 $(t, t+\Delta t)$ に到着するジョブ数」に注目
  - Δt に比例して増加
  - 平均値：$\lambda \Delta t$
- λ は単位時間あたりの平均ジョブ数

ポアソン分布

- 同じ幅をもった時間区間あたりの到着の仕方は，時刻に依存しない
- 共通部分のない時間区間たちのそれぞれの到着の仕方は独立である
- 同時刻に2人のジョブがやってくることはない
- ごく短い時間 Δtの間にジョブが1人来る確率は $\lambda \Delta t$

平均処理率

- 単位時間に処理を受けるジョブの平均値
- 処理がどの程度で行われているかの尺度

処理率 μ の指数分布

- ジョブの完了時刻がランダム
- 「あるジョブの処理の完了から次のジョブの完了までの時間」に着目
  - 平均値：$1 / \mu$
- μ は単位時間あたりの平均ジョブ完了数
  - サーバがジョブを処理中の間，Δt 内に完了する処理数：$\mu \Delta t$

指数分布

- 進行中の処理が終了する確率は，それまでに処理に要した時間に依存しない
- ある時刻に開始される処理は，それ以前に行われた処理や到着に依存しない
- ごく短い時間 Δtの間に処理が1つ終了する確率は $\mu \Delta t$

「分布」の種類

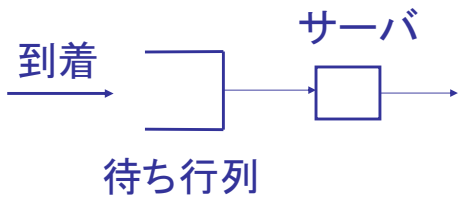
M:　ポアソン分布／指数分布
Ek:　k相のアーラン分布
D:　一定分布
G:　一般分布
GI:　独立性を有する一般分布

微小時間の意味

- 微小時間 ⊿tの間に到着するジョブ:
  - たかだか1人
- 時間 ⊿tの間に終了する処理:
  - たかだか1つ
- 時間 ⊿tの間に「ジョブの到着」「処理の終了」が同時になされることはない

M/M/1/1 待ち行列

M/M/1/1 待ち行列



- ジョブの到着: ランダム
- ジョブの完了時刻: ランダム
- サーバの個数: 1個
- 待ち行列の大きさ: 0個（K－1＝0なので）

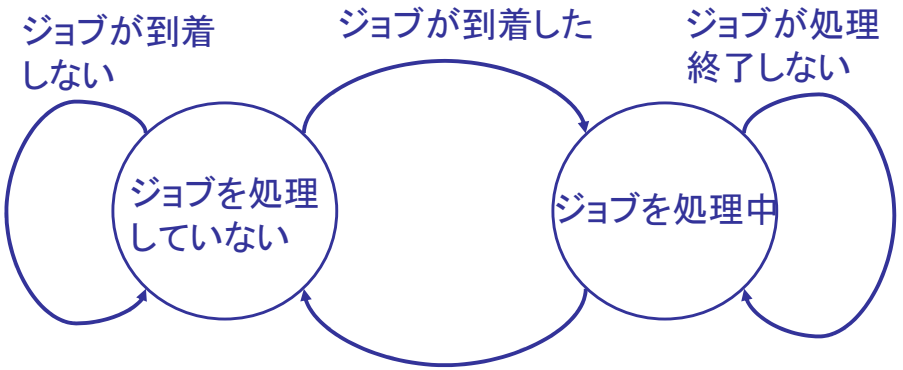
システム処理能力 ρ

$\rho = \lambda / \mu$

- $\lambda \Delta t$: 「時間 $(t, t+\Delta t)$ に到着するジョブ数」の平均
- $\mu \Delta t$: 「サーバがジョブを処理中の間，Δt 内に完了する処理数」の平均

待ち行列の大きさに限りがないとすると:
$\lambda < \mu$（つまり $\rho < 1$）である必要がある
（さもないと待ち行列があふれる）

サーバの状態遷移

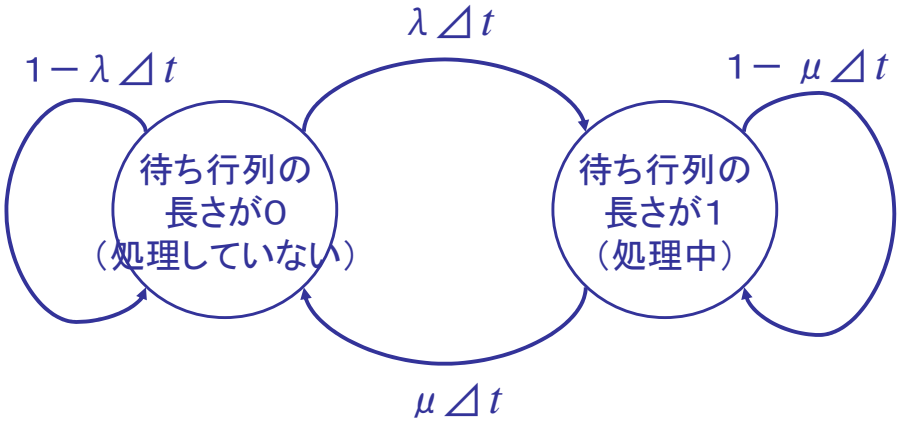


M/M/1/1 サーバの状態

- ジョブを処理中： P1
- ジョブを処理していない： P0

M/M/1/1待ち行列のサーバの状態遷移

- 制限：待ち行列の大きさは0か1



M/M/1/1待ち行列の サーバの状態遷移

$$P_0(t+\Delta t) = (1-\lambda\Delta t)P_0(t) + \mu\Delta t P_1(t)$$

$$P_1(t+\Delta t) = \lambda\Delta t P_0(t) + (1-\mu\Delta t)P_1(t)$$

定常確率

$\lim_{t\to\infty} P_0(t)$, $\lim_{t\to\infty} P_1(t)$ を求めよう

$t\to\infty$ のとき $P_0(t)\to P_0$, $P_1(t)\to P_1$（収束する）と仮定する

$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t)$ だが（理由は後述）

仮定より，$t\to\infty$ のとき $\frac{dP_0(t)}{dt} = 0$ なので

$-\lambda P_0 + \mu P_1 = 0$

これと $P_0 + P_1 = 1$ から，$P_0 = \frac{\mu}{\lambda+\mu}$, $P_1 = \frac{\lambda}{\lambda+\mu}$

定常確率

$$P_0(t+\Delta t) = (1-\lambda\Delta t)P_0(t) + \mu\Delta t P_1(t)$$

$$P_0(t+\Delta t) - P_0(t) = -\lambda\Delta t P_0(t) + \mu\Delta t P_1(t)$$

$$\frac{P_0(t+\Delta t) - P_0(t)}{\Delta t} = -\lambda P_0(t) + \mu P_1(t)$$

$$\frac{P_0(t+\Delta t) - P_0(t)}{\Delta t} = -(\lambda+\mu)P_0(t) + \mu$$

（$P_0(t) + P_1(t) = 1$から）

$P_0(t)$の方程式が求まった

定常確率

$$\lim_{\Delta t\to 0} \frac{P_0(t+\Delta t) - P_0(t)}{\Delta t} = -(\lambda+\mu)P_0(t) + \mu$$

$$\frac{dP_0(t)}{dt} = -(\lambda+\mu)P_0(t) + \mu$$

これは $P_0(t)$ の微分方程式

$$P_0(t) = \frac{\mu}{\lambda+\mu} + \left(P_0(0) - \frac{\mu}{\lambda+\mu}\right) e^{-(\lambda+\mu)t}$$

$$\lim_{\Delta t\to 0} P_0(t) = \frac{\mu}{\lambda+\mu}$$

定常状態における性質

$$-\lambda P_0 + \mu P_1 = 0$$

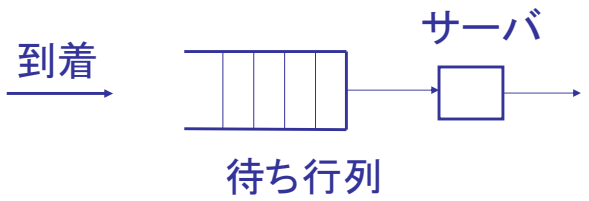
つまり $\underbrace{\lambda \Delta t \lim_{t\to\infty} P_0(t)}_{\text{新たなジョブが}\Delta t\text{以内に到着する確率}} = \underbrace{\mu \Delta t \lim_{t\to\infty} P_1(t)}_{\text{処理中のジョブが}\Delta t\text{以内に完了する確率}}$

M/M/1/1 まとめ

- 定常状態でのサーバの状態
  - $\lim_{t\to\infty} P_0(t) = \dfrac{1}{1+\rho}$
  - $\lim_{t\to\infty} P_1(t) = \dfrac{\rho}{1+\rho}$　（但し　$\rho = \lambda / \mu$ ）
- 定常状態でのサーバ内のジョブ総数
  - 0である確率: $\lim_{t\to\infty} P_0(t)$, 1である確率: $\lim_{t\to\infty} P_1(t)$

M/M/1 待ち行列

M/M/1 待ち行列



- ジョブの到着: ランダム
- ジョブの完了時刻: ランダム
- サーバの個数: 1個
- 待ち行列の大きさ: 制限なし

M/M/1 待ち行列

- 処理の窓口は1つ
- 処理を受けるための列は1つ
- いったん行列に加わったら，処理を受けるまで待ち続ける
- ジョブの到着の仕方はポアソン分布に従う
- 処理時間の分布は指数分布に従う

時刻t+⊿tにジョブが1個もない確率

- 時刻tにジョブが n 個ある確率： $P_n(t)$とおく $(n=0,1,2,\cdots)$
- 時刻 t+⊿t にジョブが1個もいないのは，次のいずれかの場合
  1. ジョブが1個もいなくて，⊿tに新たなジョブが来なかった
     $$P(A)=P_0(t)*(1-\lambda \Delta t)$$
  2. 1個のジョブが処理を受けていて，⊿tの間に処理が終了した
     $$P(B)=P_1(t)*\mu \Delta t$$
- これらは独立な事象なので，
  $$P_0(t+\Delta t) = P_0(t)*(1-\lambda \Delta t) + P_1(t)* \mu \Delta t$$

続き

- $P_n(t)$ は，時刻に依存しない（定常状態）と考えて
  $$P_n(t+\Delta t) = P_n(t)$$
- 前ページの式に代入すると
  $$P_0(t) \lambda \Delta t = P_1(t) \mu \Delta t$$
- つまり
  $$P_0(t) \lambda = P_1(t) \mu$$

$P_n(t)$

- 時刻が t から t + ⊿t になった時点で，ジョブの総数が n である場合は以下の3通り
  - 時刻tに n 個で，新たなジョブが到着せず，処理も終了しなかった
    $$P(A) = P_n(t)*(1-\lambda \Delta t)*(1-\mu \Delta t)$$
    $$= P_n(t)*(1-\lambda \Delta t - \mu \Delta t)$$
  - 時刻 t に n-1個で，新たなジョブが1つ到着した
    $$P(B) = P_{n-1}(t)* \lambda \Delta t$$
  - 時刻 t に n+1 個で，ジョブの処理が1つ終了した
    $$P(C) = P_{n+1}(t)* \mu \Delta t$$

続き

- $P_n(t+\Delta t)=P(A)+P(B)+P(C)$から

$$P_n(t)(\lambda + \mu) = P_{n-1}(t)\lambda + P_{n+1}(t)\mu$$

M/M/1 待ち行列

$$\lambda P_0 = \mu P_1$$

$$(\lambda+\mu)P_1 = \lambda P_0 + \mu P_2$$

$$\vdots$$

$$(\lambda+\mu)P_i = \lambda P_{i-1} + \mu P_{i+1}$$

M/M/1 待ち行列

$$P_1 = \rho P_0$$

$$P_2 = (1+\rho)P_1 - \rho P_0$$

$$= (1+\rho)\,\rho\, P_0 - \rho P_0$$

$$= \rho^2 P_0$$

$$P_i = \rho^i P_0$$

一方，$\Sigma P_i=1$ なので $P_i = (1-\rho)\rho^i$

続き

$$\Sigma P_i = 1$$

つまり，$P_0(t)*(1+\rho+\rho+\cdots) = 1$

- $\rho \geqq 1$
  - 処理できるジョブ数より，やってくる方が多い
  - $1+\rho+\rho+\cdots$は発散する
- $\rho < 1$
  - $1+\rho+\rho+\cdots=1/(1-\rho)$（発散しない）
  - $P_0(t)=1-\rho$

平均ジョブ数，平均待ちジョブ数

$$N = \Sigma nP_n$$

$$= \Sigma n(1-\rho)\rho^n$$

$$= \frac{\rho}{1-\rho}$$

$$Nw = \Sigma (n-1)P_n$$

$$= \Sigma nP_n - (1-P_0)$$

$$= N - (1-P_0)$$

$$= \frac{\rho^2}{1-\rho}$$

平均ジョブ数

- 平均ジョブ数をLとおくと

$$L = \sum_{n=0}^{\infty} nP_n$$

- Lを計算すると

$$L = \frac{\rho}{1-\rho}$$

- 処理を受けているジョブを含まない待ち行列内の平均ジョブ数: L_q

$$L_q = \sum_{n=1}^{\infty}(n-1)P_n$$

$$= \sum_{n=1}^{\infty}(n-1)(1-\rho)\rho^n$$

$$= \rho \sum_{n=1}^{\infty}(n-1)(1-\rho)\rho^{n-1}$$

$$= \rho \sum_{n=1}^{\infty} n(1-\rho)\rho^n$$

$$= \rho \sum_{n=1}^{\infty} nP_n$$

$$= \rho L$$

$$L_q = \frac{\rho^2}{1-\rho}$$

平均待ち時間

- ジョブが並びはじめて処理を受け始めるまでの時間の平均: W_q

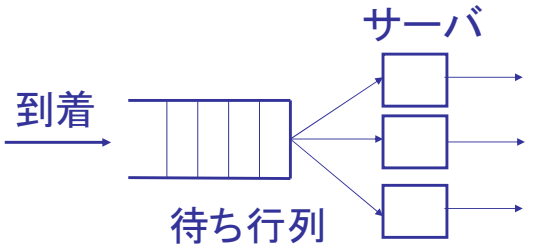
$$L_q = \lambda W_q$$

- このことから

$$W_q = \frac{L_q}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{\rho^2}{\mu(1-\rho)}$$

M/M/S 待ち行列

M/M/S 待ち行列



- ジョブの到着： ランダム
- ジョブの完了時刻： ランダム
- サーバの個数： S個
- 待ち行列の大きさ： 制限なし

M/M/S 待ち行列

$$N = \left(\sum_{n=1}^{S-1} \frac{(S\rho)^n}{(n-1)!} + \sum_{n=1}^{S-1} \frac{n(S\rho)^n}{S^{n-S}} \right) P_0$$

$$D = \frac{N}{\lambda}$$

課題

- 待ち行列プログラムを作成し，実行せよ
  - λ と μ をいろいろ変化させて実行せよ
  - また，$\lambda > \mu$ のときの振る舞いを確認せよ

サンプルプログラム

```
#include<stdio.h>
#include <stdlib.h>
#include <time.h>
main()
{
   FILE *outfile;
   int t,out[20], t_max;              /*⊿tをtとする*/
   long int n;
   float ramda,myu,n_sum,r;
   if ((outfile=fopen("output.dat", "w")) == NULL) {
      printf("can't open file %s¥n", out);
      exit(0);
   }
   printf("Input ( λ , μ ,T) :");
   scanf("%f,%f,%d", &ramda, &myu, &t_max);
   srand((unsigned int)time(NULL));       /* ランダム関数を初期化 */
   n=0;                  /*行列の初期ジョブ数は0*/
   n_sum=0;
   for (t=0;t<=t_max;t++)
   {
      /* tがt_maxになるまでシュミレーション*/
      r = (double)rand() / ((double)RAND_MAX );   /* 0≦r≦1のランダムな実数 */
      if ((n!=0) & (myu>=r)) {
         n=n-1; /* 処理を受ける*/
      }
      r = (double)rand() / ((double)RAND_MAX );   /* 0≦r≦1のランダムな実数 */
      if (ramda>=r) {
         n=n+1; /* ジョブが到着 */
      }
      fprintf(outfile,"⊿t = %d      ジョブ数 = %d¥n",t,n);    /*ファイルに出力*/
      n_sum=n_sum+n;
   }
   fprintf(outfile,"¥n¥n平均ジョブ数 = %f¥n",n_sum/t_max);
   fclose(outfile);
}
```