```c
/*  Simulação de um Sistema de Call Center
    (para disciplina de Ferramentas de Análise de Desempenho de Sistemas,
    Mestrado em Redes de Computadores, Unifacs 2004)

    2004 by Marcos Portnoi
*/

#include <stdio.h>
#include "smpl.h"
#include <stdlib.h>

#define N_sources 3 //número de fontes no modelo
#define N_facilities 5 //número de facilities no modelo
#define Name_facilities {"Voz1", "Voz2", "Voz3", "Espera", "Humano"}
#define Tau {40, 40, 40} //tempos de interchegada
#define S {15, 15, 15, 15, 10} //tempos de serviço atendimento eletrônico de voz, espera forçada e atendente humano
#define Max_time 2600000 //tempo máximo de simulação
#define Max_jobs 1750000 //número máximo de jobs ou clientes a gerar
#define SEED 2 //especifica o stream do gerador de números aleatórios do SMPL a usar (1 a 15)

void main()
{
    /*job é o número do token ou cliente; não há diferenciação entre eles
      num_cheg_fila[] é um contador para o número de chegadas por fila/facility
      tam_max_fila[] é um acumulador para armazenar o tamanho máximo medido por fila/facility
      num_job_serv[] é um contador para o número de jobs servidos por facility
      num_jobs_gerados[] é um contador de jobs gerados por fonte de geração
      name_facility contém o nome de cada facility
      tot_jobs_gerados é o acumulador global de jobs gerados por todas as fontes
      tx_cheg_fila[] representa o cálculo da taxa de chegada por fila
    */
    int job=1, event, server[N_facilities];
    int i, num_cheg_fila[N_facilities], tam_max_fila[N_facilities], num_job_serv[N_facilities];
    int num_jobs_gerados[N_sources], tot_jobs_gerados=0;
    char name_facility[][10]=Name_facilities, file_out[35]; //file_out -> nome de saída do arquivo de resultados
    float tx_cheg_fila[N_facilities], tau_source[]=Tau, s_server[]=S;
    float t_espera_linha1, t_espera_linha2, t_espera_linha3;
    FILE *fp;

    //inicialização de todos os contadores/acumuladores
    for (i=0;i<N_facilities;i++) {
      num_cheg_fila[i]=tam_max_fila[i]=num_job_serv[i]=tx_cheg_fila[i]=0;
    }
    for (i=0;i<N_sources;i++) {
      num_jobs_gerados[i]=0;
    }

    //colocar aqui o nome do arquivo de saida de resultados.  o valor do SEED será acrescentado
    sprintf(file_out, "callcenter.out.seed%d.txt", SEED);
    fp = fopen(file_out,"w");

    smpl(0,"Simulacao de um Call Center");
    //inicialização das facilities
    for (i=0; i<N_facilities; i++) {
      server[i]=facility(name_facility[i],1); //Voz1=0; Voz2=1; Voz3=2; Espera=3; Humano=4
    }
    //geração dos eventos iniciais para todas as fontes
    for (i=0; i<N_sources; i++) {
      schedule(i+1,expntl(tau_source[i]),job);
    }
    stream(SEED); //alimenta (seed) o gerador de números aleatórios do SMPL

    while (stime()<Max_time && tot_jobs_gerados<Max_jobs) //pára a simulação no primeiro estouro de tempo ou geração
      {
        cause(&event,&job);
        switch(event)
          {
            case 1:  //chegada no Voz1 (vindo da Linha1)
```

```
          schedule(4,0.0,job); //escalona atendimento no Voz1
          num_cheg_fila[0]++; //incrementa acumulador de jobs que chegaram para a fila ↙
Voz1
          num_jobs_gerados[0]++; //incrementa número de jobs gerados pela fonte
          tot_jobs_gerados++;  //incrementa acumulador de jobs gerados total
          schedule(1,expntl(tau_source[0]),job); //escalona nova chegada na Linha1
          break;
        case 2:  //chegada no Voz2 (vindo da Linha2)
          schedule(5,0.0,job); //escalona atendimento no Voz2
          num_cheg_fila[1]++; //incrementa acumulador de jobs que chegaram para a fila ↙
Voz2
          num_jobs_gerados[1]++; //incrementa número de jobs gerados pela fonte
          tot_jobs_gerados++;  //incrementa acumulador de jobs gerados total
          schedule(2,expntl(tau_source[1]),job); //escalona nova chegada na Linha2
          break;
        case 3:  //chegada no Voz3 (vindo da Linha3)
          schedule(6,0.0,job); //escalona atendimento no Voz3
          num_cheg_fila[2]++; //incrementa acumulador de jobs que chegaram para a fila ↙
Voz3
          num_jobs_gerados[2]++; //incrementa número de jobs gerados pela fonte
          tot_jobs_gerados++;  //incrementa acumulador de jobs gerados total
          schedule(3,expntl(tau_source[2]),job); //escalona nova chegada na Linha3
          break;
        case 4:  //reserva servidor Voz1
          if (request(server[0],job,0)==0)
             schedule(7,expntl(s_server[0]),job); //se estiver livre, escalona final de↙
 serviço
          else
          {
            if (inq(server[0])>tam_max_fila[0]) tam_max_fila[0] = inq(server[0]); //   ↙
atualiza tamanho máximo de fila
          }
          break;
        case 5:  //reserva servidor Voz2
          if (request(server[1],job,0)==0)
             schedule(8,expntl(s_server[1]),job); //se estiver livre, escalona final de↙
 serviço
          else
          {
            if (inq(server[1])>tam_max_fila[1]) tam_max_fila[1] = inq(server[1]); //   ↙
atualiza tamanho máximo de fila
          }
          break;
        case 6:  //reserva servidor Voz3
          if (request(server[2],job,0)==0)
             schedule(9,expntl(s_server[2]),job); //se estiver livre, escalona final de↙
 serviço
          else
          {
            if (inq(server[2])>tam_max_fila[2]) tam_max_fila[2] = inq(server[2]); //   ↙
atualiza tamanho máximo de fila
          }
          break;
        case 7:  //chegada no Espera Forçada (vindo do Voz1)
          num_job_serv[0]++; //incrementa acumulador de jobs servidos
          release(server[0],job);
          schedule(10,0.0,job); //escalona atendimento no Espera
          num_cheg_fila[3]++; //incrementa acumulador de jobs que chegaram para a fila ↙
Espera
          break;
        case 8:  //chegada no Espera Forçada (vindo do Voz2)
          num_job_serv[1]++; //incrementa acumulador de jobs servidos
          release(server[1],job);
          schedule(10,0.0,job); //escalona atendimento no Espera
          num_cheg_fila[3]++; //incrementa acumulador de jobs que chegaram para a fila ↙
Espera
          break;
        case 9:  //chegada no Atendimento Humano (vindo do Voz3)
          num_job_serv[2]++; //incrementa acumulador de jobs servidos
          release(server[2],job);
          schedule(12,0.0,job); //escalona atendimento no Humano
          num_cheg_fila[4]++; //incrementa acumulador de jobs que chegaram para a fila ↙
Humano
          break;
        case 10:  //reserva servidor Espera
```

```
          if (request(server[3],job,0)==0)
             schedule(11,expntl(s_server[3]),job); //se estiver livre, escalona final de serviço
          else
          {
            if (inq(server[3])>tam_max_fila[3]) tam_max_fila[3] = inq(server[3]); //atualiza tamanho máximo de fila
          }
          break;
        case 11:  //chegada no Atendimento Humano (vindo do Espera)
          num_job_serv[3]++; //incrementa acumulador de jobs servidos
          release(server[3],job);
          schedule(12,0.0,job); //escalona atendimento no Humano
          num_cheg_fila[4]++; //incrementa acumulador de jobs que chegaram para a fila Humano
          break;
        case 12:  //reserva servidor Humano
          if (request(server[4],job,0)==0)
             schedule(13,expntl(s_server[4]),job); //se estiver livre, escalona final de serviço
          else
          {
            if (inq(server[4])>tam_max_fila[4]) tam_max_fila[4] = inq(server[4]); //atualiza tamanho máximo de fila
          }
          break;
        case 13:  //saída servidor Humano
          num_job_serv[4]++; //incrementa acumulador de jobs servidos
          release(server[4],job);
          break;

      }
  }
//report();

fprintf(fp,"Simulacao de um Call Center com SPML\n2004 por Marcos Portnoi");
fprintf(fp,"\n\n\nTempo Simulado        :  %.2f\nNumero de Jobs Gerados:  %d", stime(), tot_jobs_gerados);
fprintf(fp, "\nSeed Utilizado        :  %d", stream(0));
fprintf(fp,"\n\n *** FONTES ***\n");
fprintf(fp,"\nFONTE - NUM.JOBS.GER. - TAXA DE GERACAO");
fprintf(fp,"\n          [jobs]        [uts/jobs]");
fprintf(fp,"\n-------------------------------------------------------------");
for(i=0;i<N_sources;i++) {
  fprintf(fp,"\n    %d %13d      %8.4f", i, num_jobs_gerados[i], tau_source[i]);
}

fprintf(fp,"\n\n\n *** SERVIDORES ***\n");
fprintf(fp,"\nSERV. -   NOME  - UTILIZ. - JOB.SERV. - TX SERVICO - TEMPO SERVICO - OCUP.FINAL");
fprintf(fp,"\n                                [jobs]     [jobs/uts]    [uts/jobs]  [0-livre;1-ocup.]");
fprintf(fp,"\n-------------------------------------------------------------------------------");
for(i=0;i<N_facilities;i++) {
  fprintf(fp,"\n  %d    %-8.8s %10.4f %10d %12.4f     %8.4f          %d",
    i, fname(server[i]), U(server[i]), num_job_serv[i], num_job_serv[i]/(stime()*U(server[i])), s_server[i], status(server[i]));
}

fprintf(fp,"\n\n\n *** FILAS ***\n");
fprintf(fp,"\nFILA -   NAME   - TX.CHG. - TAM.MED.FIL - TEM.MED.FIL - TAM.MAX - CLI. FINAL");
fprintf(fp,"\n               [jobs/uts]    [jobs]        [uts]       [jobs]     [jobs]");
fprintf(fp,"\n-------------------------------------------------------------------------------");
for(i=0;i<N_facilities;i++) {
  tx_cheg_fila[i]=num_cheg_fila[i]/stime();
  fprintf(fp,"\n  %d    %-8.8s %10.4f %10.4f   %10.4f %10d   %6d",
    i, fname(server[i]), tx_cheg_fila[i], Lq(server[i]), Lq(server[i])/tx_cheg_fila[i], tam_max_fila[i], inq(server[i]));
}

fprintf(fp, "\n\n\nOUTRAS ESTATISTICAS:");
```

```
    fprintf(fp, "\n\nLinha1:");
    t_espera_linha1=Lq(server[0])/tx_cheg_fila[0]+Lq(server[3])/tx_cheg_fila[3]+Lq(server
    [4])/tx_cheg_fila[4]+
      s_server[0]+s_server[3];
    fprintf(fp, "\n\nTempo de Espera ate ser atendido pelo Atendente Humano [uts]:  %.4f",
    t_espera_linha1);
    fprintf(fp, "\nTempo de Resposta Total ate final atendimento [uts]         :  %.4f",
    t_espera_linha1+s_server[4]);
    fprintf(fp, "\n\nLinha2:");
    t_espera_linha2=Lq(server[1])/tx_cheg_fila[1]+Lq(server[3])/tx_cheg_fila[3]+Lq(server
    [4])/tx_cheg_fila[4]+
      s_server[1]+s_server[3];
    fprintf(fp, "\n\nTempo de Espera ate ser atendido pelo Atendente Humano [uts]:  %.4f",
    t_espera_linha2);
    fprintf(fp, "\nTempo de Resposta Total ate final atendimento [uts]         :  %.4f",
    t_espera_linha2+s_server[4]);
    fprintf(fp, "\n\nLinha3:");
    t_espera_linha3=Lq(server[2])/tx_cheg_fila[2]+Lq(server[4])/tx_cheg_fila[4]+s_server[2]
    ;
    fprintf(fp, "\n\nTempo de Espera ate ser atendido pelo Atendente Humano [uts]:  %.4f",
    t_espera_linha3);
    fprintf(fp, "\nTempo de Resposta Total ate final atendimento [uts]         :  %.4f",
    t_espera_linha3+s_server[4]);

    fprintf(fp, "\n\n\nLEGENDA:");
    fprintf(fp, "\n\nNUM.JOBS.GER.   = Numero de Jobs Gerados por Fonte");
    fprintf(fp, "\nTAXA DE GERACAO = Taxa de Chegada de Jobs por Fonte");
    fprintf(fp, "\nUTILIZACAO      = Utilizacao por Servidor");
    fprintf(fp, "\nJOB.SERV        = Numero de Jobs Servidos por Servidor");
    fprintf(fp, "\nTX SERVICO      = Taxa de Servico por Servidor");
    fprintf(fp, "\nTEMPO SERVICO   = Tempo Medio de Servico por Servidor");
    fprintf(fp, "\nOCUP.FINAL      = Estado do Servidor ao Final da Simulacao (ocupado ou
    livre)");
    fprintf(fp, "\nTX.CHG.         = Taxa de Chegada de Jobs por Fila");
    fprintf(fp, "\nTAM.MED.FIL     = Tamanho Medio da Fila");
    fprintf(fp, "\nTEM.MED.FIL     = Tempo Medio de Espera em Fila");
    fprintf(fp, "\nTAM.MAX         = Tamanho Maximo Atingido pela Fila durante Simulacao");
    fprintf(fp, "\nCLI.FINAL       = Numero de Jobs em Fila ao termino da Simulacao");
    fclose(fp);
}
```