

```

/*****
/*
/* Modelisation en Promela (Spin) de la solution de la question 8
/* du sujet de Juin 2002;
/* Cette version est assez fidele du programme Ada correspondant
/*
*****/

#define false 0
#define true 1

#define N 3

/* ----- */
/* demande de client de rdv */

chan Demande_T1 = [N] of {byte, byte}; /* demande de client de rdv */
chan Demande_T4 = [N] of {byte}; /* demande de client pour servir */

chan Reponse_T1[N] = [1] of {bit}; /* reponse oui ou non */
chan Reponse_T4[N] = [1] of {bit, byte}; /* reponse oui ou non et Id client */

chan Requeue_T1 = [N] of {byte, byte};
chan Requeue_T4 = [N] of {byte};

/* ----- */
/* Le serveur
/* ----- */
proctype Serveur()
{
    bit Est_En_Attente_T1[N];
    bit Est_En_Attente_T4[N];

    byte Attent_QUI[N];
    byte Sert_QUI[N];

    bit Garde_Int_T1 = false;
    bit Garde_Int_T4 = false;

    byte X, Y, k;

    xr Demande_T1;
    xr Demande_T4;

    xr Requeue_T1; xr Requeue_T4; xs Requeue_T1; xs Requeue_T4;

boucle:
    if

        /* ----- */
        /* entree Demande_T1
        /* ----- */
        :: (!Garde_Int_T1) && (!Garde_Int_T4) && nempty(Demande_T1) ->

            Demande_T1 ? X, Y;

            if

                :: (Est_En_Attente_T4[Y]) ->
                    d_step{
                        Sert_QUI[Y] = X;
                        Garde_Int_T4 = true;
                        Reponse_T1[X] ! true;
                    }
                :: !(Est_En_Attente_T4[Y]) ->
                    if

                        :: (Est_En_Attente_T1[Y]) ->
                            Reponse_T1[X] ! false;

                        :: !(Est_En_Attente_T1[Y]) ->
                            d_step{
                                Est_En_Attente_T1[X] = true;
                                Attent_QUI[X] = Y;
                                Requeue_T1 ! X,Y;
                            }

                    fi;

            fi;

        fi;
    fi;
}

```

```

/* ----- */
/* entree Demande_T4 */
/* ----- */
:: (!Garde_Int_T1) && (!Garde_Int_T4) && nempty(Demande_T4) ->

  Demande_T4 ? X;

  atomic{
    k = 0;
    do
      :: (k==N) -> break;
      :: (k<N)&&(Est_En_Attente_T1[k])&&(Attent_QUI[k]==X) ->
        d_step{
          Y = k;
          Sert_QUI[X] = k;
          Garde_Int_T1 = true;
          Reponse_T4[X] ! true, k;
        }
        goto boucle;

      :: (k<N)&&!((Est_En_Attente_T1[k])&&(Attent_QUI[k]==X)) ->
        k++;
    od;
  }
  if
    :: (len(Requeue_T4) == N-1) ->
      Reponse_T4[X] ! false, 0;
    :: (len(Requeue_T4) < N-1) ->
      d_step{
        Est_En_Attente_T4[X] = true;
        Sert_QUI[X] = X;
        Requeue_T4 ! X;
      }
  fi;

/* ----- */
/* entree Attente_Interne_T1 */
/* ----- */
:: (Garde_Int_T1) && nempty(Requeue_T1) ->

  Requeue_T1 ? X, Y;

  if
    :: (Sert_QUI[Y] == X) ->
      d_step{
        Garde_Int_T1 = false;
        Est_En_Attente_T1[X] = false;
        Reponse_T1[X] ! true;
      }
    :: !(Sert_QUI[Y] == X) ->
      Requeue_T1 ! X,Y;
  fi;

/* ----- */
/* entree Attente_Interne_T4 */
/* ----- */
:: (Garde_Int_T4) && nempty(Requeue_T4) ->

  Requeue_T4 ? X;

  if
    :: (Sert_QUI[X] != X) ->
      d_step{
        Y = Sert_QUI[X];
        Est_En_Attente_T4[X] = false;
        Garde_Int_T4[X] = false;
        Reponse_T4[X] ! X, Y;
      }
    :: !(Sert_QUI[X] != X) ->
      Requeue_T4 ! X;
  fi;

  fi;

  goto boucle;
}

```

03 jui 02 16:00

prog2_promela.spec

Page 3/3

```

/* ----- */
/* Les clients */
/* ----- */
proctype Client(byte Ego)
{
    byte Y;
    bit Ok;

loop:
    /* choix d'un nombre */
    if
        :: Y = 0;
        :: Y = 1;
        :: Y = 2;
    fi;
    if
        :: (Ego != Y) ->
            Demande_T1 ! Ego, Y;
            Reponse_T1[Ego] ? Ok;
            if
                :: (Ok) -> printf("%d va en rdv avec %d\n", Ego, Y);
                :: (!Ok) -> printf("refus pour %d de rdv avec %d\n", Ego, Y);
            fi;
        :: (Ego == Y) ->
            Demande_T4 ! Ego;
            Reponse_T4[Ego] ? Ok, Y;
            if
                :: (Ok) -> printf("%d va en rdv avec %d\n", Ego, Y);
                :: (!Ok) -> printf("refus pour %d de servir\n", Ego, Y);
            fi;
    fi;
    goto loop;
}

/* ----- */
/* Le processus init */
/* ----- */
init{
    int id = 0;

    atomic{
        run Serveur();
        do
            :: (id < N) ->
                run Client(id);
                id++;
            :: (id == N) ->
                break;
        od
    }
}

/* ----- */
/*
/* La verification de ce modele avec l'outil Spin et les options de
/* compilation -DMA=110 (permettant de reduire la taille physique du graphe*/
/* ainsi que l'option "Use partial order reduction" donne les resulats
/* suivants (on remarque errors: 0 qui signifie "pas d'interblocage"
/*
/* (Spin Version 3.2.4 -- 10 January 1999)
/* + Partial Order Reduction
/* + Graph Encoding (-DMA=110)
/*
/* Full statespace search for:
/* invalid endstates +
/*
/* State-vector 136 byte, depth reached 573638, errors: 0
/*
/* Minimized Automaton: 344644 nodes and 1.09625e+06 edges
/*
/* 1.00163e+07 states, stored
/* 5.11985e+06 states, matched
/* 1.51362e+07 transitions (= stored+matched)
/* 3.72422e+06 atomic steps
/* ----- */

```