Programowanie współbieżne - ćwiczenia 13
(monitory cz.2)

Zadanie 1 (Biuro)

W pewnym biurze grupa urzędników obsługuje grupę klientów. Każdy urzędnik ma unikatowy identyfikator z zakresu od 1 do N. Algorytmy urzędnikow i klientów są następujące:

```
process Urzędnik (u:integer)
begin
 repeat
   BIURO.CHCĘ_PRACOWAĆ(u);
   rozmawiam_z_klientem;
   BIURO.SKOŃCZYŁEM;
   odpoczywam;
 until false;
end;
```

```
process Klient
var u:integer;
begin
 repeat
   BIURO.CHCĘ_ZAŁATWIĆ_SPRAWĘ(u);
   rozmawiam_z_urzędnikiem(u);
   BIURO.ZAŁATWILEM;
   własne_sprawy;
 until false;
end;
```

- W biurze pracuje co najmniej jeden urzędnik wyższej rangi lub co najmniej dwóch urzędników niższej rangi.
- Na wyposażenie biura składa się K (K $>$ 2) krzeseł, z których korzystają urzędnicy i klienci podczas rozmowy.
- Do rozmowy dochodzi, jeżeli jest zainteresowany klient i chętny do pracy urzędnik wyższej rangi lub dwóch urzędników niższej rangi oraz wystarczająca liczba krzeseł (każda osoba zajmuje jedno krzesło).
- Urzędnicy w ramach każdej z grup pracują według kolejności zgłaszania się do pracy.
- Klienci muszą dowiedzieć się z jakmi urzędnikami mają rozmawiać (muszą poznać ich identyfikatory).
- Po skończonej rozmowie urzędnicy i klienci mogą opuszczać krzesła pojedynczo.
- Biuro powinno pracować w ten sposób, aby obsłużyć jak największą liczbę klientów i jest to priorytetem w tym zadaniu.

Rozwiązanie

```
monitor BIURO
var
  ile_krzeseł: integer;
  ilu_u2: integer;                { ilu jest gotowych urzędników II }
  u: array [1..2] of condition;   { tu czekają urzędnicy I i II }
  k: condition;                   { tu czekają klienci }
```

```
  z_kim1, z_kim2: integer;        { identyfikatory urzędników }

export procedure CHCĘ_PRACOWAĆ(ranga: 1..2, id: integer);
begin
  if ranga = 1 then begin
       { czeka, jeżeli nie ma klientów lub wystarczającej liczby krzeseł }
     if empty(k) or ile_krzeseł < 2 then wait u[1];
     ile_krzeseł := ile_krzeseł - 2;
     z_kim1 := id;          { zapisuje swój identyfikator }
     z_kim2 := 0;           { informuje, ze nie będzie drugiego urzędnika }
     signal(k);             { zwalnia klienta }
  end
  else begin { czeka, jeżeli nie ma klientów lub wystarczającej liczby
               krzeseł lub oczekującego urzędnika II }
         if empty(k) or ile_krzeseł < 3 or empty(u[2]) then begin
                                                          inc(ilu_u2)
                                                          wait u[2];
                                                          dec(ilu_u2);
                                                        end;
             { jeżeli jest pierwszym urzędnikiem, to zapisuje
               swój identyfikator i budzi drugiego urzędnika }
         if z_kim1 = 0 then begin
                             ile_krzeseł := ile_krzeseł - 3;
                             z_kim1 := id;
                             signal(u[2]);
                            end
             { jeżeli jest drugim urzędnikiem, to zapisuje
               swój identyfikator i budzi klienta }
         else begin
                z_kim2 := id;
                signal(k);
              end;
end;

export procedure CHCĘ_ZAŁATWIĆ_SPRAWĘ(var id1, id2: integer);
begin
    { klient czeka jeżeli }
  if (empty(u[1]) or ile_krzeseł < 2) and { nie ma urzędnika I lub 2 krzeseł }
  (ilu_u2 < 2 or ile_krzeseł < 3) { nie ma dwóch urzędników II lub 3 krzeseł }
  then wait(k)
  else if not empty(u[1]) then signal(u[1]) { budzi urzędnika I }
       else signal(u[2]);                   { budzi pierwszego urzędnika II }
  id1 := z_kim1;                            { odczytuje identyfikatory }
  id2 := z_kim2;
  z_kim1 := 0;
end;
```

```
export procedure SKOŃCZYŁEM/ZAŁATWIŁEM;
begin
  inc(ile_krzeseł);
    { jeżeli może dojść do rozmowy, to zwalnia odpowiedniego urzędnika }
  if not empty(k) then
    if not empty(u[1]) and ile_krzeseł >= 2 then signal(u[1])
    else if ilu_u2 >= 2 and ile_krzeseł >= 3 then signal(u[2]);
end;

begin
  ile_krzeseł := K;
  ilu_u2 := 0;
  z_kim1 := 0;
end.
```

Uwagi

Schemat rozwiązania jest następujący: kiedy nie może dojść do rozmowy (z powodu braku jednej ze stron lub miejsc do siedzenia), wtedy proces czeka w odpowiedniej kolejce. Jeżeli pojawi się brakujący urzędnik pierwszej rangi, to zwalnia klienta. Jeżeli pojawi się brakujący urzędnik drugiej rangi, to zwalnia drugiego urzędnika, który zwalnia klienta. Jeżeli pojawi się brakujący klient, to zwalnia urzędnika i jeżeli jest to urzędnik drugiej rangi, to zwalnia swojego kolegę, który zwolniłby klienta, ale tego nie robi, bo kolejka klientów jest pusta (gdyby nie była, to nie brakowało by klienta – patrz: początek zdania). Jeżeli pojawiają się brakujące krzesła, to zwalniani są odpowiedni urzędnicy, którzy zwalniają klienta.

Warto zwrócić uwagę na przekazywanie identyfikatorów klientowi. Jeżeli to klient zwalnia urzędników, to identyfikatory zostają przekazane w momencie, kiedy klient wykonuje `signal`.

Priorytetowym wymaganiem w tym zadaniu jest obsłużenie jak największej ilości klientów. Aby je spełnić należy w pierwszej kolejności wybierać urzędników pierwszej rangi, ponieważ zajmują oni mniej krzeseł. Oczywiście może dojść do zagłodzenia drugiej grupy urzędników, ale jest to zagłodzenie wynikające z treści zadania.

Zadanie 2 (Zasoby)

W systemie są dwie grupy procesów korzystające z N zasobów typu A i M zasobów typu B ($N + M > 1$). Procesy z pierwszej grupy cyklicznie wykonują własne sprawy, po czym wywołują procedurę `zamieńAB`, która konsumuje jeden zasób A i produkuje jeden zasób B. Procesy z grupy drugiej cyklicznie wykonują własne sprawy, po czym wywołują procedurę `zamień`, która konsumuje jeden zasób dowolnego typu i produkuje zasób przeciwny. Zsynchronizuj procesy za pomocą monitora tak, aby:

- procedury `zamieńAB` i `zamień` były wywoływane przez procesy jedynie pod warunkiem dostępności odpowiednich zasobów

- procesy z grupy pierwszej wykonywały procedurę `zamieńAB` parami, tzn. proces z grupy pierwszej może rozpocząć jej wykonanie jedynie wtedy, gdy jest inny proces z grupy pierwszej, gotowy do jej wykonania (i oczywiście niezbędne zasoby)
- jednocześnie mogło odbywać się wiele operacji na zasobach, ale nie doszło do zagłodzenia żadnej grupy procesów

Monitor powinien udostępniać jedynie procedury wywoływane przez procesy przed i po rozpoczęciu korzystania z zasobów.

Rozwiązanie

```
monitor Zasoby;
var
  ileA, ileB, iluNaA: Integer;
  zasóbA, zasób, partner: condition;
export procedure ChceA;
begin
  inc(iluNaA);
  if iluNaA mod 2 = 1 then    { pierwszy z pary }
    wait (partner);           { czeka na partnera }
  else begin                  { drugi z pary }
    if ileA < 2 then          { jeżeli nie ma dwóch zasobów A }
      wait (zasóbA);          { czeka na co najmniej dwa zasoby A }
    ileA := ileA - 2;
    signal (partner)          { zwalnia partnera }
  end
  dec(iluNaA);
end;

export procedure ChceZasób (var z: Zasob);
begin
  if IleB + IleA = 0 then
    wait (zasób);             { czeka na cokolwiek }
  if ileB > 0 then            { najpierw B, żeby produkować A }
  begin
    z := B; dec (IleB)
  end else
  begin
    z := A; dec (ileA)
  end
end;

export procedure Wyprodukowalem (z: Zasób);
begin
  if z = B then              { jeżeli wyprodukowano B }
  begin
```

```
    inc (ileB);  signal (Zasób)
  end
  else
  begin
    inc (ileA);
    if empty (ZasóbA) then   { tylko jeżeli nie ma par w pierwszej grupie }
      signal (Zasób)         { to zwalniamy swoją grupę }
    else                     { czeka para na zasób A }
      if ileA > 1 then       { zwalniamy ja jeśli można lub }
        signal (ZasóbA)      { czekamy na drugi egzemplarz zasobu A }
  end
end;

begin
  iluNaA := 0;
  ileA := N;
  ileB := M;
end.
```

Uwagi

Aby nie zagłodzić procesów pierwszej grupy, proces który wyprodukował A, zwalnia procesy drugiej grupy tylko wtedy, gdy nie ma pary gotowych procesów z grupy pierwszej, nawet jeżeli jest tylko jeden zasób, który na razie jest dla tej pary bezużyteczny.