Kolokwium z Programowania Współbieżnego 2020-12-09

Zadanie 1 (Semafory: Symultana Szachowa, 10p.)

Do Symultanicznej Symultany Szachowej zgłosiło się M arcymistrzów (`process Arcymistrz(int id: 0..M-1)`) oraz grupa amatorów (`process Amator()`). W sali gier rozłożono `K` szachownic. Liczba K może być mniejsza od liczby amatorów i od liczby arcymistrzów.

Każdy amator w pętli nieskończonej załatwia własne sprawy (`void sekcja_lokalna()`), po czym przystępuje do turnieju. Amator znajduje wolną szachownicę, siada przy niej, a następnie oczekuje na przyjście jednego z arcymistrzów. Po każdym ruchu arcymistrza amator sprawdza, czy partia już się skończyła (`bool czy_koniec(int szachownica)`). Jeśli partia się skończyła, amator odchodzi od szachownicy. Jeśli partia się nie skończyła amator wykonuje swój następny ruch (`void ruch(int szachownica)`).

Każdy arcymistrz w pętli nieskończonej załatwia własne sprawy (`void sekcja_lokalna()`), po czym przystępuje do symultany. W pojedynczej symultanie arcymistrz zaczyna grać z co najmniej $S > 1$ amatorami siedzącymi przy szachownicach, którzy nie grają jeszcze z żadnym arcymistrzem.

Następnie arcymistrz cyklicznie przechodzi między szachownicami, na których jeszcze trwa rozgrywka z nim, wykonując kolejne ruchy na przemian z amatorami (`void ruch(int szachownica)`). Arcymistrz może rozpocząć następną symultanę dopiero po zakończeniu partii ze wszystkimi amatorami, z którymi zaczął grać.

W ustalonym momencie tylko jeden gracz (amator lub arcymistrz) może wykonywać ruch na danej szachownicy. Pierwszy ruch w partii zawsze wykonuje amator, a każda partia kończy się po skończenie wielu ruchach. Wykonanie ruchu może zająć dowolnie długi, ale skończony czas.

Zapisz treść procesów `Amator()` i `Arcymistrz(int id: 0..M-1)` w języku C rozszerzonym o operacje semaforowe. Twoje rozwiązanie nie powinno dopuszczać do zagłodzenia żadnego procesu i powinno w wydajny sposób wykorzystywać dostępne zasoby.

Zadanie 2 (Monitory: Zakład produkcyjny, 10p.)

W zakładzie pracy jest $K > 1$ taśm produkcyjnych. Każda taśma ma dwa stanowiska: montażowe i lakiernicze. W zakładzie pracuje pewna (skończona) liczba nadzorców i pewna (skończona) liczba monterów. Nadzorca zajmuje się przyjmowaniem zleceń oraz lakierowaniem złożonych urządzeń, przy czym lakierowanie na kolor k ($0 \leqslant k < K$) odbywa się zawsze na taśmie o numerze k. Monterzy zajmują się składaniem urządzenia. W dowolnej chwili na każdej taśmie odbywa się produkcja co najwyżej jednego urządzenia.

Urządzenia mają różny stopień komplikacji i są składane przez zespoły monterów, formowane na bieżąco w kolejności zgłaszania się ich do pracy. Każdy monter ma parametr określający poziom jego kompetencji. Suma poziomów kompetencji członków zespołu składającego urządzenie musi być nie mniejsza niż stopień komplikacji tego urządzenia.

Nadzorca cyklicznie:

- przyjmuje do realizacji zlecenie na produkcję urządzenia o stopniu komplikacji `stKomplikacji` i kolorze `kolor` – predefiniowana funkcja `wezZlecenie(int &stKomplikacji, int &kolor)`
- czeka aż taśma o numerze `kolor` będzie wolna. Gdy to nastąpi kompletuje zespół monterów, którzy rozpoczynają składanie urządzenia i czeka aż wszyscy monterzy zakończą montaż – wszystko to odbywa się w funkcji monitorowej `czekamNaTasmeiUrzadzenie(int stKomplikacji, int kolor)`
- lakieruje urządzenie (predefiniowana funkcja `lakieruj(int kolor)`)
- zwalnia taśmę (funkcja monitorowa `zwalniam(int kolor)`)

Monter cyklicznie:

- zgłasza się do pracy podając swój poziom kompetencji i dowiadując się, do której taśmy ma się udać (funkcja monitorowa `chcePracowac(int pozKompetencji, int& tasma)`)
- gdy zbierze się cały zespół, zajmuje się składaniem urządzenia (predefiniowana funkcja `montaz(int tasma)`)
- czeka aż cały zespół zakończy montaż (funkcja monitorowa `czekajNaKoniecMontazu(int tasma)`)
- odpoczywa (predefiniowana funkcja `odpoczynek()`)

Napisz kod monitora `ZAKLAD` udostępniającego wymienione wyżej funkcje. Treść procesów `NADZORCA` i `MONTER` jest następująca:

```
process NADZORCA() {
   while (true) {
      int stKomplikacji, int kolor;
      wezZlecenie(&stKomplikacji, &kolor);
      ZAKLAD.czekamNaTasmeiUrzadzenie(stKomplikacji, kolor);
      lakierowanie(kolor);
      ZAKLAD.zwalniam(kolor);
  }
}
process MONTER (int pozKompetencji) {
    int tasma;
    while (true) {
      ZAKLAD.chcePracowac(pozKompetencji, &tasma);
      montaz(tasma);
      ZAKLAD.czekajNaKoniecMontazu(tasma);
      odpoczynek();
    }
}
```