Przetwarzanie Równoległe – zadanie 2A

12.03.2013

Zaprojektuj i zaimplementuj synchronizację dla poniższego problemu. Należy użyć mechanizmów biblioteki POSIX.

Problem.

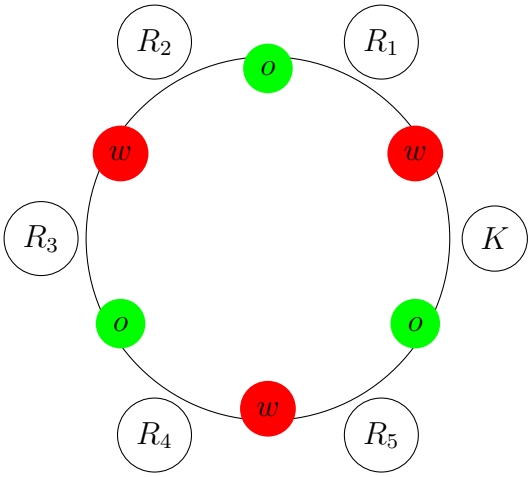
1. Przy okrągłym stole siedzi $n > 4$ rycerzy i król Artur (czyli razem $n+1$ osób). Przyjmijmy, że n jest nieparzyste.

2. Uczestnicy uczty na zmianę opowiadają o swoich zwycięstwach i popijają wino bezalkoholowe.

3. Opowiadanie trwa od 100 do 500 ms (losowo). Jeśli rycerz opowiada, żaden z jego sąsiadów nie może opowiadać ani pić w tym samym czasie (rycerze są kulturalni i słuchają tego co mówią ich koledzy). Jeśli król opowiada, nikt przy stole nie może opowiadać.

   Jeśli rycerz/król A aktualnie pije albo chce pić i czeka na kielich, a jego sąsiad B zacznie opowiadać, rycerz A może dokończyć picie i dopiero wtedy zacząć słuchać.

4. Pomiędzy rycerzami naprzemiennie stoją kielichy z winem (bezalkoholowym) i talerzyki na ogórki (patrz rysunek). Na początku kielichy i talerzyki są puste.



5. Na środku stołu stoi dzban z winem i misa z ogórkami. W dzbanie mieści się $n/2$ kielichów wina, a w misie mieści się $n/3$ ogórków. Początkowo dzban i misa są pełne.

6. Jeśli rycerz/król chce pić, chwyta stojący przy nim kielich i talerzyk, nalewa wina, nakłada ogórka, a następnie je wypija/zjada (musi mieć w ręku oba).

7. Między rycerzami nie ma żadnych priorytetów. Król ma priorytet nad rycerzami (tj. jeśli król i rycerz jednocześnie chcą złapać ten sam kielich/ogórka, rycerz ustępuje królowi).

8. Po wypiciu wina kielich jest pusty (nie można z niego pić). Analogicznie nie można zjeść drugi raz tego samego ogórka.

9. Co pewien (losowy – częstotliwość trzeba dobrać tak, żeby coś było widać) czas pojawia się kelner, który uzupełnia wino w dzbanie i donosi ogórki (za jednym przyjściem uzupełnia oba do pełna).

10. Po wypiciu dziesięciu kolejek, rycerz/król kończy uczestnictwo w uczcie.

Wymagania techniczne.

1. W kodzie powinny być komentarze objaśniające synchronizację.

2. W konsoli powinny pojawiać się komunikaty informujące o aktualnym stanie procesów.

3. Aplikacja powinna być oddana w postaci kodu źródłowego i dołączonego pliku makefile. Polecenie `make compile` powinno kompilować aplikację, a polecenie `make run [n]` powinno wywołać program z parametrem n (liczbą całkowitą). Domyślna wartość dla n to 9 (tj. polecenie `make run` jest równoważne z `make run 9`).

4. **Termin wgrania rozwiązania: 31.03.2013 (23:59)**