Teoria współbieżności

Labolatorium 2

Andrzej Ratajczak

1. Zaimplementować semafor binarny za pomocą metod wait i notify, użyć go do synchronizacji programu Wyścig.

Celem ćwiczenia był przygotowanie semafora na podstawie przygotowanego szkieletu oraz zastosowanie go w celu synchronizacji poprzedniego ćwiczenia, zgodnie z poniższymi warunkami.

Opuszczenie, s.P(): Jeśli S==1 to S=0, w przeciwnym wypadku wstrzymaj działanie procesu wykonującego tę operację.

Podniesienie, s.V(): Jeśli są procesy wstrzymane w wyniku opuszczania semafora S, to wznów jeden z nich, w przeciwnym razie S=1.

```
class Semafor {
    private boolean _stan = true;
    private int _czeka = 0;

    public Semafor(boolean stan) {
        _stan = stan;
    }

    public synchronized void P() {
        _czeka++;
        if(_stan == false) {
            try{
                this.wait();
            } catch(InterruptedException e) {
                System.out.println("Ooops, something went wrong " +
                                   e.getMessage());
            }
        }
        _czeka--;
        _stan = false;
    }

    public synchronized void V() {
        if(_czeka > 0) {
            this.notify();
        }
        _stan = true;
    }
}
```

Kod źródłowy 1. Implementacja semafora binarnego.

Następnie otoczyłem wywołania metody w sekcji krytycznej operacjami semafora.

```
class IThread extends Thread {

    private Counter counter;
    private Semafor semafor;

    public IThread(Counter c, Semafor s) {
        counter = c;
        semafor = s;
    }

    public void run() {
        for (int i = 0; i < 1000000; i++) {
            System.out.println("i " + i);
            semafor.P();
            counter.inc();
            semafor.V();
        }
    }
}
```

Kod źródłowy 2. Przykładowa sekcja krytyczna jednego z wątków.

Jednak po uruchomieniu programu okazuje się, że nie działa wszystko jak powinno. Nie otrzymujemy spodziewanej wartości 0. Dlaczego tak się dzieje odpowiem w kolejnym paragrafie.

2. Pokazać, że do implementacji semafora za pomocą metod wait i notify nie wsytarczy instrukcja *if*, tylko ptorzeba użyć *while*. Wyjaśnić teoretycznie dlaczego i potwierdzić eksperymentem w praktyce.

W poprzednim paragrafie stworzyliśmy binarny semafor, który nie działa. Powodem jest warunek oczekiwania. W obecnej sytuacji jest on zaimplementowany przy użyciu isntrukcji if. Dlaczego jest on niebezpieczny? Problem leży w działaniu metod wait i notify w czasie wykonania. Metoda notify wybudza wątek, ale nie gwarantuje nam tego, że to nastąpi natychmiast. Istnieje szansa, że jeden z wątków uśpiony w monitorze P otrzyma polecenie wybudzenia od innego wątku, który znajduje się w monitorze V. Jednakże zajmie mu to na tyle długo, że inny wątek (możliwe, że ten sam, który był jeszcze przed chwilą w kontekście monitora V, zajmie monitor P. Wtedy ten wątek, który dotąd był uśpiony, będzie już obudzony, będzie za instrukcją wait, i będzie oczekiwał na otrzymanie monitora P. Jeżeli jednak zmienimy instrukcję if, na instrukcję while, nasz dopiero co obudzony wątek będzie musiał przed kontynuację sprawdzić, czy warunek jednego wątku w sekcji krytycznej jest spełniony. Oczywiście nie jest, bo inny wątek ustawił flagę na zajętą, więc wątek powróci do metody wait i zacznie oczekiwać na nowe wybudzenie.

```
class Semafor {
    private boolean _stan = true;
    private int _czeka = 0;

    public Semafor(boolean stan) {
        _stan = stan;
    }

    public synchronized void P() {
        _czeka++;
        while(_stan == false) {
            try{
                this.wait();
            } catch(InterruptedException e) {
                System.out.println("Ooops, something went wrong " +
                                   e.getMessage());
            }
        }
        _czeka--;
        _stan = false;
    }

    public synchronized void V() {
        if(_czeka > 0) {
            this.notify();
        }
        _stan = true;
    }
}
```

Kod źródłowy 3. Implementacja semafora binarnego przy użyciu instrukcji while.

Rzeczywiście tak zaimplementowany semafor deterministycznie zwraca za każdym razem oczekiwaną wartość, czyli 0.

3. Zaimplementować semafor licznikowy (ogólny) za pomocą semaforów binarnych. Czy semafor binarny jest szczególnym przypadkiem semafora ogólnego?

Celem ćwiczenia było zaimpelementować semafor ogólny przy użyciu binarnych. Do takiego mechanizmu wystarczą tylko dwa semafory binarne.

Jak taki semafor ma działać. Również będziemy potrzebować metody zajęcia i opuszczenia sekcji krytycznej. Nasz semafor powinien również mieć dwa oddzielne semafory binarne oraz zostać zainicjalizowany z licznikiem całkowitym dodatnim, który oznacza ile różnych wątków może wejść do sekcji krytycznej.

Zasada działania jest bardzo prosta. Jeden z semaforów będzie wyznaczał sekcję krytyczną ciał funkcji P i V, tak aby mieć pewność że tylko jeden wątek będzie mógł używać globalnie wejścia do lub wyjścia z sekcji krytycznej. W metodzie P każdy wątek musi zmniejszyć licznik o 1 oraz sprawdzić warunek. Jeżeli licznik jest >= 0 to znaczy że może po prostu opuścic metodę P i wpuścić inny wątek. Jeżeli jest ujemna to znaczy że weszło już n innych wątków i należy poczekać. Taki wątek powinien zablokować się na naszym drugim semaforze.

Tak samo z drugiej strony w metodzie V należy zwiększać licznik oraz sprawdzać czy jest ujemny, jeżeli tak, znaczy to że jakieś wątki utknęły na drugim semaforze. Należy jeden obudzić oraz opuścić sekcję krytyczną w V.

Najlepiej obrazuje to poniższa implementacja.

```
class CountingSemafor {
    private boolean _stan = true;
    private int _size = 0;

    Semafor s1 = new Semafor(true);
    Semafor s2 = new Semafor(false);

    public CountingSemafor(int size) {
        _size = size;
    }

    public void P() {
        s1.P();
        _size--;
        if(_size < 0 ) {
            s1.V();
            s2.P();
        } else {
            s1.V();
        }
    }

    public void V() {
        s1.P();
        _size++;
        if(_size <= 0) {
            s2.V();
            s1.V();
        } else {
            s1.V();
        }
    }
}
```

Kod źródłowy 4. Implementacja semafora ogólnego.

Czy semafor binarny jest szczególnym przypadkiem ogólnego? Taki semafor ogólny można użyć jako binarny, jeżeli zainicjalizujemy go z licznikiem równym 1.