Wątki i procesy

Maciej Dawid

Proces

Proces – program w trakcie wykonywania, jest sekwencją zmian systemu komputerowego, które odbywają się zgodnie z zapisanym w programie algorytmem.

Realizacja procesu zajmuje się procesor.

Procesor i pamięć operacyjna to urządzenia niezbędne do wykonania każdego procesu.

Proces jest ciągiem czynności, zaś program jest ciągiem instrukcji.

Współbieżność

Jeśli w jednym komputerze istnieje w tej samej chwili czasowej wiele procesów, to mówimy, że są one wykonywane współbieżnie

Komputer na którym jest jeden procesor centralny – współbieżność osiąga się przez przeplatanie porcji poszczególnych procesów

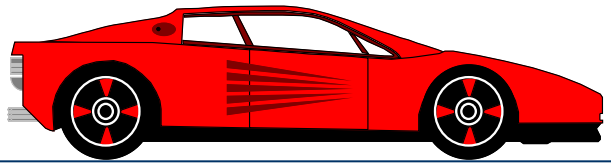
Komputer ma N procesorów – N procesów może być wykonywanych równolegle (realnie jednocześnie)

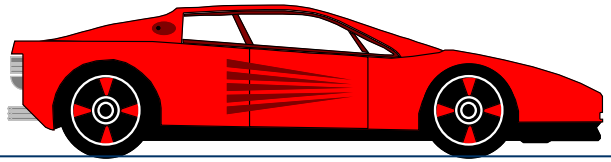
Programowanie rozproszone

Programowanie rozproszone (ang. *distributed processing*) to wykonywanie określonych zadań użytkownika z wykorzystaniem współdzielonych zasobów połączonych siecią komputerową.

Procesy współbieżne

Dwa procesy są współbieżne, jeśli jeden z nich rozpoczyna się przed zakończeniem drugiego.





Zasób dzielony

Wspólny obiekt, z którego może korzystać w sposób wyłączny wiele procesów nazywa się zasobem dzielonym (może być wykorzystywany tylko przez jeden proces lub ograniczoną ich liczbę, mniejszą od liczby chętnych).

Współzawodnictwo procesów

O współzawodnictwie mówimy, gdy dwa procesy ubiegają się o ten sam zasób (np. dostęp do tej samej komórki pamięci).

Współzawodnictwo wymaga synchronizacji, gdyż akcja procesu musi być wstrzymana, jeśli zasób potrzebny do jej wykonania jest w danej chwili zajęty przez inny proces.

Komunikacja

Komunikacja miedzy parą procesów obejmuje działania po stronie procesu nadawczego oraz odbiorczego, dające w wyniku:

- przenoszenie danych za środowiska procesu nadawczego do środowiska procesu odbiorczego
- synchronizację czynności odbiorczych z czynnościami nadawczymi

Program współbieżny

Procesy interakcyjne komunikują się i synchronizują w sposób określony przez programistę – program opisujący zachowanie się zbioru takich współbieżnych procesów nazywa się programem współbieżnym

Program współbieżny składa się z kilku (co najmniej dwóch) współbieżnych procesów sekwencyjnych, które muszą się ze sobą komunikować lub synchronizować swoje działania

Wątek

Wątek jest podstawową jednostką wykorzystującą procesor

Różnica między procesem a wątkiem polega przede wszystkim na sposobie wykorzystania dostępnych zasobów.

Każdy proces ma przydzielony odrębny obszar pamięci operacyjnej

Grupa równorzędnych wątków współdzieli tą samą przestrzeń adresową, kod i zasoby systemu operacyjnego

Wielowątkowość

Wątki działają praktycznie równolegle. Równoległość działań w ramach procesu osiągamy przez uruchamianie kilku różnych wątków.

Każdy proces posiada co najmniej jeden wątek nazywany wątkiem głównym.

Zmiany wątków mogą dokonywać się według następujących mechanizmów:

współpracy - wątek sam decyduje kiedy oddać czas procesora innym wątkom

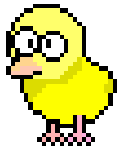
wywłaszczania – o dostępie wątków do procesora decyduje systemowy zarządca wątków (przydział z przeplotem)

Wielowątkowość w JAVA

- uruchamianiem wątków i zarządzaniem nimi zajmuje się klasa ***Thread***
- aby uruchomić wątek należy stworzyć obiekt klasy ***Thread*** i użyć metody *start()* wobec tego obiektu
- metoda *run()* określa co ma robić wątek

Stany wątku

Wątek może znajdować się w czterech stanach:



- New Thread

- Runnable

- NotRunnable

- Dead

Pierwszy sposób tworzenia i uruchomienia wątku

- zdefiniować własną klasę dziedziczącą ***Thread***
- przedefiniować odziedziczoną metodę *run()* podając w niej działania, które ma wykonywać wątek
- stworzyć obiekt własnej klasy
- wysłać mu komunikat *start()*

Przykład

```
public class Timer extends Thread{
public void run(){
…
while(true){
   try{
          this.sleep(1000);
}catch (InterruptedException exc){
…}
}
}
```

kod wątku jest opisany w metodzie *run()*

w pętli *while* usypiamy wątek wykonujący metodę *run()* na 1 sek

w trakcie uśpienia wątek jest odsuwany od procesora

```
…….
Timer tm= new Timer();
tm.start
….
```

utworzenie i uruchomienie wątku w programie

Drugi sposób tworzenia i uruchamiania wątku

- zdefiniować klasę implementującą interfejs *Runnable* *(np. class C implements Runnable)*
- dostarczyć w niej definicji metody *run()* (co ma robić watek)
- utworzyć obiekt tej klasy (np. C c=new C();)
- utworzyć obiekt klasy *Thread* przekazując w konstruktorze referencje do obiektu utworzonego (np.Thread th=new Thread(c);)
- wywołać na rzecz nowoutworzonego obiektu klasy *Thread* metodę *start()* (th.start();)

Przykład

```
class Timer implements Runnable {
public void run(){
int time=0;
while(true) {
try {Thread.sleep(1000);
} catch(InterruptedException exc) {
System.out.println(" watek zostal przerwany");
return;}
time++;
}}}
```

Koniec pracy wątku

- Wątek kończy pracę, wtedy gdy zakończy się jego metoda *run()*
- Programowo zakończyć jego pracę – należy sprawdzić w metodzie *run()* warunki zakończenia
- Gdy są spełnione spowodować wyjście z *run()* (np. *return*)
- W klasie *Thread* znajduje się metoda *stop()*

Blokada

Zbiór procesów znajduje się w stanie blokady (zastoju, martwego punktu), jeśli każdy z tych procesów jest wstrzymywany w oczekiwaniu na zdarzenie, które może być spowodowane tylko przez inny proces z tego zbioru.

Zagłodzenie

Zagłodzenie (zakleszczenie) występuje wówczas, gdy proces nie zostaje wznowiony, mimo, że zdarzenie na które czeka, występuje dowolną ilość razy, ale za każdym razem jest wybierany jakiś inny czekający proces.