Proces można uznać za działający program, któremu system przeznaczył zasoby (pamięć, system plików, procesor, urządzenia wejścia-wyjścia) natomiast wątek wchodzi w skład aktualnie działającego procesu i korzysta z zasobów nadanych procesowi.

* wątki uruchamiane są w ramach jednego procesu (programu)
* procesy posiadają niezależne zasoby - wątki współdzielą zasoby nadane procesowi (przestrzeń adresowa, gniazda itp)
* tworzenie wątków zajmuje mniej czasu niż tworzenie procesów
* wątki zużywają więcej zasobów w trakcie działania

[](http://4.bp.blogspot.com/_-5zc7bx-GmI/S8tbsic8hTI/AAAAAAAAAIY/_oVhDWpsDhc/s1600/3.png)

ThreadPool

Pula wątków umożliwia w elegancki sposób zarządzać wieloma małymi zadaniami. Tworzenie obiektów Thread powoduje zużycie zasobów, jednak daje możliwość zarządzania wątkiem (wstrzymywanie, wznawianie, przerwanie). ThreadPool zmniejsza zużycie zasobów, jednak po uruchomieniu wątku nie możemy nim jakkolwiek zarządzać. Innym problemem jest to, że wątki w ThreadPool są uruchomiane jako wątki w tle. Dla nas oznacza to, że w momencie zakończenia głównego wątka, niewykonane wątki z puli zostaną przerwane.

Mutex

 Mutex w swoim działaniu jest bardzo podobny do lock. W odróżnieniu jednak od locka pozwala na działanie w obrębie „komputera” a nie tylko aplikacji. Należy też zaznaczyć, że lock jest dużo szybszy od Mutexu. Jest jednak jedno szczególne zastosowanie Mutexa, które znalazło szerokie zastosowanie w programowaniu. Dzięki temu, że Mutex nie jest zależny tylko od aplikacji, można dzięki niemu zaimplementować możliwość uruchomienia tylko jednej instancji naszej aplikacji.

Semafor

 Internecie można znaleźć takie porównanie: „Semafor jest jak nocny klub – ma określoną pojemność pilnowaną przez bramkarza. Kiedy klub jest pełny, nikt więcej nie wejdzie dopóki ktoś nie wyjdzie. Podczas tworzenia obiektu tej klasy, konstruktor wymaga dwóch parametrów – pojemności klubu oraz ilość wolnych miejsc.

Task vs Threads + ContinueWith

<http://www.pzielinski.com/?p=192>