Wątki

Wątek (thread) lub **proces lekki** (Lightweight process - LWP) – podstawowa jednostka wykorzystywana przez proces.

Wątek posiada: identyfikator wątku, zbiór rejestrów, osobne stosy użytkownika i jądra, obszar prywatnej pamięci danych. Współużytkuje wraz z innymi równorzędnymi wątkami sekcję kodu, sekcję danych oraz takie zasoby systemu operacyjnego, jak otwarte pliki i sygnały, co łącznie określa się jako *zadanie* (*task).*

Proces wielowątkowy to taki, w którym jest wiele kontrolowanych przepływów sterowania.

W procesie jednowątkowym wątkiem jest przepływ sterowania (control flow).

Korzyści

- Responsiveness (zdolność do reagowania, np. serwer wielowątkowy)
- Resource sharing (dzielenie zasobów) powoduje, że przełączanie procesora między równorzędnymi wątkami, jak również tworzenie wątków, jest tanie w porównaniu z przełączaniem kontekstu między tradycyjnymi procesami ciężkimi. Choć przełączanie kontekstu między wątkami nadal wymaga przełączania zbioru rejestrów, jednak nie trzeba wykonywać żadnych prac związanych z zarządzaniem pamięcią.
- Economy (ekonomika, tworzenie i przełączanie wątków jest tańsze, ponadto zob. wyżej)
- Utilization of MP architectures (spożytkowanie architektur wieloprocesorowych)

Wątki użytkownika i jądra

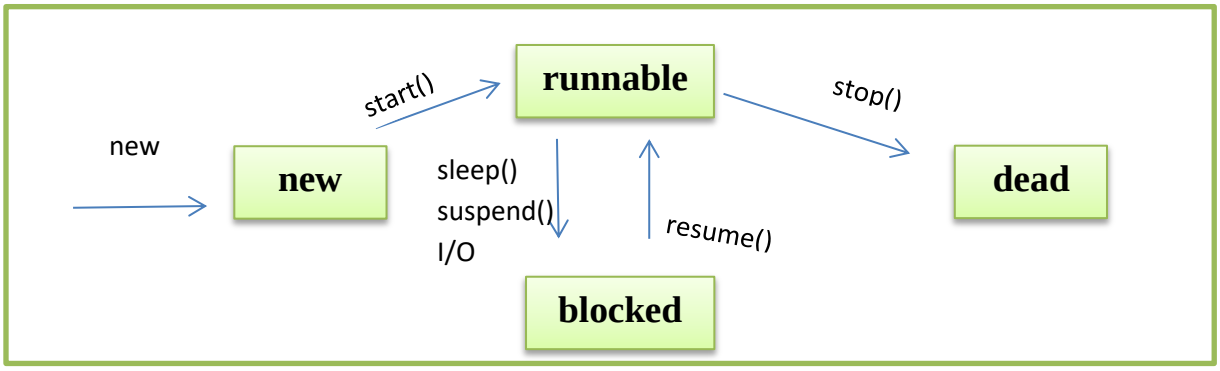
Wątki użytkownika (user-level threads), z których korzysta się za pośrednictwem wywołań bibliotecznych zamiast odwołań do systemu, dzięki czemu przełączanie wątków nie wymaga wzywania systemu operacyjnego i przerwań związanych z przechodzeniem do jego jądra. Wadą jest na przykład to, że jeśli jądro jest jednowątkowe, to każdy wątek poziomu użytkownika odwołujący się do systemu będzie powodował oczekiwanie całego zadania na zakończenie wywołania systemowego.

Wątki jądra (kernel-level threads) są obsługiwane przez jadro. Wątki poziomu użytkownika nie angażują jądra, więc dają się przełączać szybciej niż wątki realizowane przez jądro. Jednakże dowolne wywołanie systemu operacyjnego może powodować oczekiwanie całego procesu, ponieważ jądro planuje tylko procesy (nic nie wiedząc o wątkach), a proces czekający nie otrzymuje przydziału czasu procesora. Planowanie może odbywać się też niesprawiedliwie. Rozważmy dwa procesy, z których jeden ma tylko jeden wątek (proces a), a drugi ma 100 wątków (proces *b).* Oba procesy otrzymają na ogół tę samą liczbę kwantów czasu, wskutek czego wątek w procesie *a* będzie działał 100 razy szybciej niż wątek w procesie *b.* W systemach, w których wątki są organizowane przez jądro, przełączanie wątków zabiera więcej czasu, gdyż zajmuje się tym jądro (za pośrednictwem przerwań). Każdy wątek może jednak podlegać indywidualnemu planowaniu, dzięki czemu proces *b* otrzyma 100 razy więcej czasu procesora niż proces *a.*

Wątki są udostępniane wprost przez system operacyjny MS Windows, w systemach Linux, BSD i innych dostępna jest biblioteka **pthread**, dająca jednolity interfejs, ukrywający

szczegóły implementacji. W językach programowania używających maszyn wirtualnych (Python, Java itp.) są dostępne również tzw. **zielone wątki**, które nie są obsługiwane przez system operacyjny, ale samą maszynę wirtualną - to pozwala m.in. na realizację współbieżności nawet wtedy, gdy docelowy system operacyjny nie udostępnia wątków.

Stany wątku



Działanie wątków pod wieloma względami przypomina działanie procesów. Podobnie jak procesy, wątki użytkują wspólnie jednostkę centralną i w danej chwili tylko jeden wątek jest aktywny (wykonywany). Wykonanie wątku w procesie przebiega sekwencyjnie, a każdy wątek ma własny stos i licznik rozkazów. Wątki mogą tworzyć wątki potomne i mogą blokować się do czasu zakończenia wywołań systemowych. Jeśli jeden wątek jest zablokowany, to może działać inny wątek. Jednak w odróżnieniu od procesów wątki nie są niezależne od siebie. Ponieważ wszystkie wątki mają dostęp do każdego adresu w zadaniu, dany wątek może czytać i zapisywać stosy dowolnych innych wątków. Struktura taka nie umożliwia ochrony na poziomie wątków. Ochrona ta nie powinna jednakże być konieczna. Podczas gdy procesy pochodzą od różnych użytkowników i mogą być do siebie wrogo usposobione, zadanie z wieloma wątkami może należeć tylko do jednego użytkownika. W tym przypadku wątki zostaną prawdopodobnie tak zaprojektowane, aby pomagać sobie wzajemnie, więc nie będą wymagały wzajemnej przed sobą ochrony.

Modele wielowątkowości

- „wiele na jeden” (Many-to-One) - wiele wątków poziomu użytkownika jest odwzorowywanych na jeden wątek jądrowy (czyli na wątek wykonania procesu), stosowane w systemach bez wątków jądrowych.
- „jeden na jeden” (One-to-One) - każdy wątek poziomu użytkownika jest odwzorowywany
- na wątek jądrowy.
- „wiele na wiele” (Many-to-Many) - polega na odwzorowywaniu wielu wątków poziomu użytkownika, Umożliwia systemowi operacyjnemu tworzenie odpowiedniej liczby wątków jądrowych.

Zagadnienia

Kasowanie wątku (*thread cancellation*) - wymuszone kończenie wątków.

- kasowanie asynchroniczne (*asynchronous cancellation*) powoduje natychmiastowe zakończenie wskazanego
- kasowanie odroczone (*deferred cancellation*): zezwala się, aby docelowy wątek okresowo sprawdzał, czy powinien być skasowany.

Obsługa sygnałów (*signal handling*)

- sygnał powstaje wskutek wystąpienia pewnego zdarzenia,
- sygnał jest dostarczany do procesu,
- sygnał jest obsługiwany.

Możliwości:

- dostarczyć sygnał wątkowi, do którego sygnał się odnosi,
- dostarczyć sygnał każdemu wątkowi w procesie,
- dostarczyć sygnał wybranym wątkom w procesie,
- przydzielić specjalny wątek do odbierania wszystkich sygnałów dotyczących danego procesu.

Pule wątków (*thread pools*) - utworzenie pewnej puli (liczby) wątków, w której oczekują one na zatrudnienie. Zalety:

- z użyciem istniejącego wątku można zwykle nieco szybciej obsłużyć zamówienie niż za pomocą wątku, który trzeba najpierw utworzyć;
- liczbę wątków aplikacji można ograniczyć do rozmiaru puli.

Dane charakterystyczne wątku (*thread specific data*) - Zezwala się na posiadanie przez każdy wątek własnej kopii danych. Przydatne, gdy nie mamy kontroli nad procesem tworzenia wątków (tzn. gdy korzystamy z puli wątków).

Uaktywnianie planisty (*scheduler activations*) - powoduje **wezwania** (*upcalls*) — jest to mechanizm komunikacji prowadzącej z jądra do biblioteki wątków (rzadki przypadek naruszenia zasad warstwowości).