Temat: Procesy, wątki oraz pamięć wirtualna.

1. Pojęcie procesu.
2. Wątek jako część procesu.
3. Pamięć wirtualna.

Pojęcie procesu

Proces - jedno z najbardziej podstawowych pojęć w informatyce, definiowane jako egzemplarz wykonywanego programu, jednak każdy nowo powstały proces otrzymuje unikalny numer, który go jednoznacznie identyfikuje, tzw. numer PID (ang. *process IDentifier*).

W celu wykonania programu system operacyjny przydziela procesowi zasoby (pamięć, czas procesora i inne - szczegółowa lista zasobów znajduje się dalej), ale także może być konieczne współbieżne wykonywanie pewnych fragmentów programu. Aby to zrealizować program może zażądać utworzenia określonej liczby wątków, wykonujących wskazane części programu - o ich współbieżne wykonanie dba system operacyjny (albo sam program, wówczas mówi się o zielonych wątkach). Wątki współdzielą prawie wszystkie zasoby zarezerwowane dla danego procesu, wyjątkiem jest **czas procesora**, który jest przydzielany indywidualnie każdemu wątkowi.

Za zarządzanie procesami odpowiada jądro systemu operacyjnego, sposób ich obsługi jest różny dla różnych systemów operacyjnych. W systemie operacyjnym każdy proces posiada proces nadrzędny, z kolei każdy proces może, poprzez wywołanie funkcji systemu operacyjnego, utworzyć swoje procesy potomne; w ten sposób tworzy się swego rodzaju drzewo procesów. Każdy proces otrzymuje od systemu operacyjnego odrębne zasoby, w tym odrębną przestrzeń adresową, listę otwartych plików, urządzeń itp.

W skład procesu wchodzi:

- kod programu,
- licznik rozkazów,
- stos,
- sekcja danych.

Każdemu procesowi przydzielone zostają zasoby, takie jak:

- procesor,
- pamięć,

- dostęp do urządzeń wejścia-wyjścia,
- pliki.

Użytkownik za pomocą powłoki zleca uruchomienie programu, proces wywołujący wykonuje polecenie *fork*, lub jego pochodną.

- System operacyjny tworzy przestrzeń adresową dla procesu oraz strukturę opisującą nowy proces w następujący sposób:
    - wypełnia strukturę opisującą proces,
    - kopiuje do przestrzeni adresowej procesu dane i kod, zawarte w pliku wykonywalnym,
    - ustawia stan procesu na działający,
    - dołącza nowy proces do kolejki procesów oczekujących na procesor (ustala jego priorytet),
    - zwraca sterowanie do powłoki użytkownika.

Dany proces rozpoczyna wykonywanie w momencie przełączenia przez Jądro systemu operacyjnego przestrzeni adresowej na przestrzeń adresową danego procesu oraz takie zaprogramowanie procesora, by wykonywał kod procesu. Wykonujący się proces może żądać pewnych zasobów, np. większej ilości pamięci. Zlecenia takie są na bieżąco realizowane przez system operacyjny.

Wykonanie procesu musi przebiegać sekwencyjnie. Może przyjmować kilka stanów:

- aktualnie wykonywany przez procesor,
- czekający na udostępnienie przez system operacyjny zasobów,
- Uśpiony
- przeznaczony do zniszczenia,
- proces zombie,
- właśnie tworzony itd.

Proces wykonuje ostatnią instrukcję - zwraca do systemu operacyjnego kod zakończenia. Jeśli proces zakończył się poprawnie zwraca wartość 0, w przeciwnym wypadku zwraca wartość kodu błędu.

- W momencie zwrotu do systemu operacyjnego kodu zakończenia, system operacyjny ustawia stan procesu na przeznaczony do zniszczenia i rozpoczyna zwalnianie wszystkich zasobów, które w czasie działania procesu zostały temu procesowi przydzielone.

- System operacyjny po kolei kończy wszystkie procesy potomne w stosunku do procesu macierzystego.
- System operacyjny zwalnia przestrzeń adresową procesu. Jest to dosłowna śmierć procesu.
- System operacyjny usuwa proces z kolejki procesów gotowych do uruchomienia i szereguje zadania. Jest to ostatnia czynność wykonywana na rzecz procesu.
- Procesor zostaje przydzielony innemu procesowi.

Wątek jako część procesu

Wątek (ang. *thread*) - część programu wykonywana współbieżnie w obrębie jednego procesu; w jednym procesie może istnieć wiele wątków.

Różnica między zwykłym procesem a wątkiem polega na współdzieleniu przez wszystkie wątki działające w danym procesie przestrzeni adresowej oraz wszystkich innych struktur systemowych (np. listy otwartych plików, gniazd, itp.) - z kolei procesy posiadają niezależne zasoby.

Ta cecha ma dwie ważne konsekwencje:

1. Wątki wymagają mniej zasobów do działania i też mniejszy jest czas ich tworzenia.
2. Dzięki współdzieleniu przestrzeni adresowej (pamięci) wątki jednego zadania mogą się między sobą komunikować w bardzo łatwy sposób, niewymagający pomocy ze strony systemu operacyjnego. Przekazanie dowolnie dużej ilości danych wymaga przesłania jedynie wskaźnika, zaś odczyt (a niekiedy zapis) danych o rozmiarze nie większym od słowa maszynowego nie wymaga synchronizacji (procesor gwarantuje atomowość takiej operacji).

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.

W systemach wieloprocesorowych, a także w systemach z wywłaszczaniem, wątki mogą być wykonywane równocześnie (współbieżnie). Równoczesny dostęp do wspólnych danych grozi jednak utratą spójności danych i w konsekwencji błędem działania programu.

Podręcznikowy przykład: ciąg instrukcji odczyt-zmiana-zapis.

Załóżmy że program ma dane do przetwarzania, umieszczone w N pierwszych komórkach tablicy X. Liczba N zapisana jest w odpowiedniej zmiennej. Algorytm przetwarzania mógłby wyglądać następująco:

1. odczytaj zmienną N i sprawdź, czy jest równa 0
2. jeśli tak (nie ma danych w X), przejdź do kroku 7.
3. (tu wchodzimy, gdy N równe 1 lub więcej) odczytaj wartość X[N]
4. zmniejsz wartość N o 1 (zaznacz, że N-ta dana została już zabrana)
5. zrób coś z tą odczytaną daną (tu następuje właściwe przetwarzanie)
6. (dana obsłużona - zajmij się następną) przejdź do kroku 1.
7. (koniec pracy)

Jest to najprostsza pętla opróżniająca stos X. W środowisku jednowątkowym działa zgodnie z oczekiwaniami, przetwarzając kolejno dane X[N], X[N-1], itd. aż do X[1], po czym zatrzymuje się z zerową wartością zmiennej N.

Jednak w środowisku wielowątkowym dwa równoczesne wątki mogą wykonać się w taki sposób (załóżmy N=2):

wątek 1		wątek 2	
krok	czynność	krok	czynność
1.	odczyt N=2		
2.	(nic; N > 0)	1.	odczyt N=2
3.	odczyt X[N], czyli **X[2]**	2.	(nic; N > 0)
		3.	odczyt X[N], czyli **X[2]**
4.	zapis N=1		
		4.	zapis N=1
5.	(przetwarzanie)	5.	(przetwarzanie)
6.	(przejście do 1.)	6.	(przejście do 1.)

Jak widać, oba wątki pobrały do przetwarzania tę samą daną X[2]. Jeśli nasz program jest systemem księgowym, a w X[2] było zapisane "dokonaj przelewu kwoty xxxx z rachunku nnnn na rachunek mmmm", to przelew zostanie zaksięgowany dwukrotnie.

W dalszym ciągu wykonania tego samego programu możliwy jest również inny przypadek. Przypuśćmy, że wątek nr 1 wolniej przetwarzał X[2] i teraz wątek nr 2 zaczyna kolejny cykl:

wątek 1		wątek 2	
krok	czynność	krok	czynność
		1.	odczyt N=1
1.	odczyt N=1	2.	(nic; N > 0)
		3.	odczyt X[N], czyli **X[1]**
2.	(nic; N > 0)		
		4.	zapis N=0
3.	odczyt X[N], czyli **X[0]**		
	....		....

Pomimo posłużenia się licznikiem N, wątek nr 1 usiłuje pobrać *nieistniejący* element danych spod nielegalnego indeksu - X[0]. W zależności od różnych czynników spowoduje to albo natychmiastowe awaryjne przerwanie działania programu, albo dowolne, nieprzewidywalne zaburzenia (błędy) w dalszym jego działaniu.
Do zapobiegania takim sytuacjom wykorzystuje się mechanizmy synchronizacji wątków: semafory, muteksy, sekcje krytyczne.
Wielowątkowość (ang. *multithreading*) – cecha systemu operacyjnego, dzięki której w ramach jednego procesu może wykonywać kilka wątków lub jednostek wykonawczych. Nowe wątki to kolejne ciągi instrukcji wykonywane oddzielnie. Wszystkie wątki tego samego procesu współdzielą kod programu i dane.
Cechy wielowątkowości

- wszystkie wątki wykonują się w ramach tylko jednego programu (procesu) - innymi słowy jeden proces posiada wiele instancji wykonawczych (wątków)
- wątki zostały wprowadzone aby umożliwić przetwarzanie współbieżne, np gdy zachodzi potrzeba wykonania wielu zadań jednocześnie. Może się to wiązać również ze zwiększeniem wydajności o ile istnieją odpowiednie zasoby sprzętowe (wiele procesorów lub obsługa wielowątkowości przez pojedynczy procesor). Należy pamiętać iż wprowadzenie wątków może obniżyć wydajność ponieważ najczęściej wymagane jest przy tym wprowadzenie odpowiednich mechanizmów synchronizacji;
- wszystkie wątki procesu współdzielą tę samą wirtualną przestrzeń adresową (mają dostęp do tych samych "egzemplarzy" zmiennych, obiektów i struktur) i korzystają z tych samych zasobów systemowych;
- komunikacja między wątkami w odróżnieniu od procesów jest bardzo łatwa do wykonania – w przypadku wątków wystarczy odwoływać się do tych samych

zmiennych i obiektów – komunikacja między procesami wymaga zastosowania mechanizmów IPC (ang. *InterProcess Communication*);

- współdzielenie wirtualnej przestrzeni adresowej niesie zagrożenie – jeden "wadliwy" wątek może zagrozić wykonaniu całego programu;
- każdy wielowątkowy system operacyjny zapewnia specyficzne metody synchronizacji wątków, które z wyżej wymienionych przyczyn należy bezwzględnie zaimplementować.

Systemy wielowątkowe to m.in. BeOS, Microsoft Windows 95, Windows NT, Unix, Linux.

Pamięć wirtualna

Pamięć wirtualna oznacza oddzielenie pamięci fizycznej od pamięci logicznej dostępnej użytkownikowi. U podstaw tej idei leży obserwacja, że w każdej chwili tylko część pamięci procesu musi znajdować się w pamięci operacyjnej komputera - ta część, która akurat jest używana przez proces. Jest to możliwe dzięki temu, że procesy charakteryzują się *lokalnością* - w konkretnej chwili proces nie potrzebuje całej przydzielonej mu pamięci, a jedynie jej część. Pamięć logiczna może być więc większa niż fizyczna, wystarczy że nie używane w danej chwili obszary pamięci logicznej zostaną przeniesione na pamięć drugorzędną, np. na dysk. Taka pamięć drugorzędna używana do "udawania" pamięci operacyjnej jest nazywana obszarem wymiany.

Adresy w pamięci wirtualnej są odwzorowywane na adresy pamięci fizycznej, jednak przestrzeń adresów logicznych może być większa niż przestrzeń adresów fizycznych. Konstrukcja MMU musi być rozszerzona tak, żeby było wiadomo, czy dany adres logiczny odpowiada adresowi fizycznemu i jakiemu, a jeżeli nie, to jakiemu miejscu w obszarze wymiany odpowiada.

Mechanizm pamięci wirtualnej stanowi rozszerzenie mechanizmów stronicowania lub segmentacji, omówionych w poprzednim wykładzie. Są to, odpowiednio, mechanizmy *stronicowania na żądanie* albo *segmentacji na żądanie*. W przypadku segmentacji na żądanie, tablica segmentów tłumaczy numer segmentu na spójny obszar w pamięci fizycznej lub w obszarze wymiany. W przypadku stronicowania na żądanie, tablica stron tłumaczy numery stron na numery ramek lub numery bloków dyskowych w obszarze wymiany. Stronicowanie na żądanie jest dużo bardziej elastyczne niż segmentacja na żądanie, dlatego też w praktyce wyparło segmentację na żądanie. W dalszej części wykładu ograniczymy się do

stronicowania na żądanie.
Mechanizm pamięci wirtualnej umożliwia też współdzielenie przestrzeni adresowej przez wiele procesów, oraz pozwala efektywniej tworzyć procesy.
W domyślnej konfiguracji Windows sam ustala rozmiar pliku stronicowania (pagefile.sys), który uzależniony jest od ilości zainstalowanej pamięci RAM. Obecnie nikogo nie dziwią wielkości RAM rzędu 4-8 GB, jest to standard jak na dzisiejsze czasy. Łatwo sobie wyobrazić, że przy takim poziomie pamięci system zawyży rozmiar pliku wymiany, co może przyczynić się do zmarnowania nawet kilku gigabajtów powierzchni dysku. Nie musimy godzić się z tą sytuacją, możemy ręcznie zmodyfikować ustawienia i dopasować je tak, aby maksymalnie zoptymalizować rozmiar.
System operacyjny ustawia plik stronicowania na zmienny rozmiar. Konkretniej mówiąc ustalany jest minimalny i maksymalny rozmiar swap. Początkowo rozmiar pliku pagefile.sys równa się wartości minimalnej, w trakcie użytkowania systemu i aplikacji następuje stopniowe zwiększenie rozmiaru pliku, niestety często powoduje to znaczące obciążenie dysku, co w niektórych sytuacjach może uniemożliwiać komfortową pracę.
Można temu zapobiec poprzez ustawienie stałego rozmiaru pamięci wirtualnej. Wtedy to bez względu na zmieniające się zapotrzebowanie, plik nie będzie podlegał rozszerzaniu. Zaletą takiego rozwiązania oprócz wyżej opisanego, jest także zmniejszenie stopnia fragmentacji plików.
Można by wymienić wiele zalet stosowania stałego pliku wymiany, jednakże takie rozwiązanie niesie ze sobą jeden problem. W przypadku wykorzystania całej pamięci wirtualnej system odmówi posłuszeństwa, wszelkie niezapisane dane zostaną utracone. Dlatego też niezmiernie istotną rzeczą jest ustalenie bezpiecznego pułapu, który zagwarantuje, że pamięć szybko się nie wyczerpie. Oczywiście przy większych pojemnościach RAM rzadko kiedy spotkamy się z tą krytyczną sytuacją, jednakże należy mieć na uwadze powyższą przestrogę.