SYSTEMY OPERACYJNE

Procesy, wątki, wielozadaniowość

I. PROCESY

1. Koncepcja procesu w informatyce.

a) Proces jest elementarną jednostką pracy (aktywności) zarządzaną przez system operacyjny, która ubiega się o zasoby systemu komputerowego w celu wykonania programu.

b) Proces = wykonujący się program.

2. Elementy składowe procesu:

a) program - definiuje zachowanie procesu,

b) dane - zbiór wartości przetwarzanych oraz wyniki,

c) zbiór zasobów tworzących środowisko wykonawcze,

d) blok kontrolny procesu (PCB, deskryptor) - opis bieżącego stanu procesu.

Należy odróżnić jednak proces od wątku - każdy proces posiada własną przestrzeń adresową, natomiast wątki posiadają wspólną sekcję danych.

Program (ang. program) jest to kod źródłowy aplikacji skompilowany do pliku wykonywalnego.
Służy do wykonania zadania określonego z poziomu tworzenia kodu.
Programem może być też zestaw instrukcji napisany w kodzie maszynowym, zawarty również w pliku wykonalnym

Program jest to zapis algorytmu, w jakimś języku programowania.

Program komputerowy (ang. *computer program*) – sekwencja symboli opisująca realizowanie obliczeń zgodnie z pewnymi regułami zwanymi językiem programowania[1]. Program jest zazwyczaj wykonywany przez komputer (np. wyświetlenie strony internetowej), zwykle bezpośrednio, jeśli wyrażony jest w języku zrozumiałym dla danej maszyny lub pośrednio – gdy jest interpretowany

przez inny program (interpreter). Program może być ciągiem instrukcji opisujących modyfikacje stanu maszyny, ale może również opisywać obliczenia w inny sposób

Każdemu procesowi przydzielone zostają **zasoby**, takie jak:

- procesor,
- pamięć,
- dostęp do urządzeń wejścia-wyjścia,
- pliki.

Każdy proces posiada tzw. "rodzica". W ten sposób tworzy się swego rodzaju drzewo procesów.
Proces może (ale nie musi) mieć swoje procesy potomne.
Za zarządzanie procesami odpowiada jądro systemu operacyjnego.

Wykonanie procesu musi przebiegać sekwencyjnie. Może przyjmować kilka stanów:

- **Nowy** — formowanie procesu, czyli gromadzenie zasobów niezbędnych do rozpoczęcia wykonywania procesu, z wyjątkiem procesora (kwantu czasu procesora), a po zakończeniu formowania oczekiwanie na przyjęcie do kolejki procesów gotowych.
- **Wykonywany** — wykonywanie instrukcji programu danego procesu i wynikająca z ich wykonywania zmiana stanu odpowiednich zasobów systemu.
- **Oczekujący** — zatrzymanie wykonywania instrukcji programu danego procesu ze względy na potrzebę przydziału dodatkowych zasobów, konieczność otrzymania danych lub osiągnięcia odpowiedniego stanu przez otoczenie procesu (np. urządzenia zewnętrzne lub inne procesy).

- **Gotowy** — oczekiwanie na przydział kwantu czasu procesora (dostępność wszystkich niezbędnych zasobów z wyjątkiem procesora).
- **Zakończony** — zakończenie wykonywania programu, zwolnienie większości zasobów i oczekiwanie na możliwość przekazania informacji o zakończeniu innym procesom lub jądru systemu operacyjnego.

Cykl zmian stanów procesu

Pozostawanie procesu z stanie *zakończony* (w systemach uniksopodobnych zwany *zombi*) spowodowane jest przetrzymywaniem pewnych informacji o procesie po jego zakończeniu (np. statusu zakończenie). Całkowite usunięcie procesu mogłoby oznaczać zwolnienie pamięci i utratę tych informacji.

3. **Elementy składowe procesu:**
   - **kod programu,**
   - **licznik rozkazów,**
   - **stos,**
   - **sekcja danych.**

Proces zombie to wpis w tablicy procesów opisujący program, którego wykonanie w systemie operacyjnym zostało zakończone, ale którego zamknięcie nie zostało jeszcze obsłużone przez proces rodzica.

Termin ten odnosi się zazwyczaj do systemów z rodziny UNIX, gdzie pozostawienie wpisu *zombie* tymczasowo zajmującego pozycję w tablicy procesów zapobiega ponownemu wykorzystaniu danego PIDa i możliwym na skutek tego pomyłkom programistycznym. Wpisy takie nie dają się wyeliminować poleceniem kill, czemu prawdopodobnie zawdzięczają swoją złowrogą nazwę.

Chociaż wpisy *zombie* nie obciążają znacząco komputera, nieprawidłowo napisany program, który nie obsługuje zakończenia pracy potomków, może doprowadzić do destabilizacji pracy systemu. Dzieje się tak gdy cała tablica procesów, zwykle posiadająca sztucznie ograniczony rozmiar, zostanie zajęta przez wpisy zombie.
Hierarchia powiadamiania o procesach *zombie* może zostać czasowo zmieniona przez mechanizm **ptrace, czasem prowadząc do utrudnień przy debuggowaniu programów.**

4. **Tworzenie procesów**

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.

5. Wykonywanie procesów

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.

II. WĄTKI

1. **Wątek (**ang. *thread*) - to jednostka wykonawcza w obrębie jednego procesu, będąca kolejnym ciągiem instrukcji wykonywanym w obrębie tych **samych danych (w tej samej przestrzeni adresowej).**
   Wątki tego samego procesu korzystają ze wspólnego kodu i danych, mają jednak oddzielne stosy.
   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

2. **Wielowątkowość to 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. W systemach nie obsługujących wielowątkowości pojęcia procesu i wątku utożsamiają się.

. **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)

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 X z rachunku(A) na rachunek (B)", 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]**		

III. WIELOZADANIOWOŚĆ

1. **Wielozadaniowość – cecha** systemu operacyjnego umożliwiająca mu równoczesne wykonywanie więcej niż jednego procesu. Zwykle za poprawną realizację wielozadaniowości odpowiedzialne jest jądro systemu operacyjnego.

Równoczesność jest pozorna, gdy system ma dostępnych mniej procesorów niż zadań do wykonania. Wówczas dla uzyskania wrażenia wykonywania wielu zadań jednocześnie, konieczne staje się dzielenie czasu.

Systemy wielozadaniowe można podzielić na oferujące i nie oferujące wywłaszczania.
W systemach **z wywłaszczaniem** może nastąpić przerwanie wykonywania procesu, odebranie mu procesora i przekazanie sterowania do planisty.

Pełne wywłaszczanie zapewniają mechanizmy sprzętowe działające niezależnie od oprogramowania (na przykład przerwanie zegarowe).
W systemach **bez wywłaszczania** procesy powinny same dbać o sprawiedliwy podział czasu, co często uzyskuje się pośrednio – proces dokonując wywołania systemowego, oddaje sterowanie procesowi jądra, lub jednemu z procesów systemowych i w ten sposób zrzeka się procesora. Program nie wykonywany pozostaje "w uśpieniu" do momentu, gdy znów zostanie mu przydzielony czas procesora.

2. **Wielozadaniowość w Windows 10**

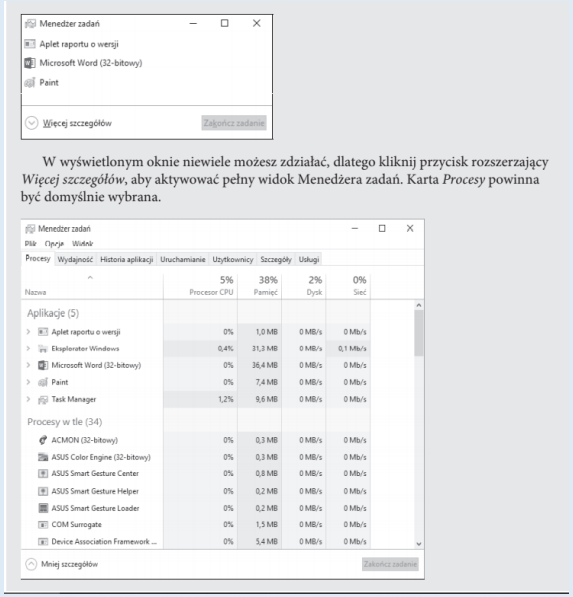
Poznaj trzy różne sposoby na jednoczesne wykonywanie wielu zadań i korzystanie z kilku pulpitów w systemie Windows 10.

ZADANIE

Wyświetlanie informacji o procesach za pomocą narzędzia Menedżer zadań.

Wbudowane narzędzie Menedżer zadań Windows zapewnia uproszczoną listę procesów systemowych. Narzędzie możesz uruchomić na jeden z następujących czterech sposobów:

1. Użycie kombinacji klawiszy Ctrl+Shift+Esc.
2. Kliknięcie paska zadań prawym przyciskiem myszy, a następnie kliknięcie pozycji Menedżer zadań.
3. Użycie kombinacji klawiszy Ctrl+Alt+Delete i kliknięcie przycisku Menedżer zadań.
4. Uruchomienie pliku wykonywalnego Taskmgr.exe. Przy pierwszym uruchomieniu Menedżer zadań znajduje się w trybie mniejszej liczby szczegółów, w przypadku którego pokazane są wyłącznie procesy z widocznym oknem najwyższego poziomu. Prezentuje to poniższy zrzut ekranu.



Zadanie należy wykonać do 05.05.2020r. i wysłać na mojego e-maila.