1. Rola pamięci operacyjnej

Pamięć operacyjna jest jedną z podstawowych części systemu komputerowego. Do niej trafiają niemal wszystkie dane programów (a także i same programy - zostanie to wyjaśnione później), które będą przetwarzane przez procesor. To z tej właśnie pamięci, procesor doczytuje potrzebne informacje i to w tej pamięci znajduje się system operacyjny, który nadzoruje działanie całego komputera. Ale nie tylko. Do pamięci operacyjnej załadowanych jest wiele sterowników urządzeń I/O (a w systemie MS-DOS także część BIOS'u komputera).

Pamięć operacyjną należy traktować jako zbiór komórek przechowujących informacje. Z dostępem do komórek pamięci ściśle wiąże się pojęcie szyny danych, której szerokość (ilość bitów, które można przesyłać jednocześnie - popularnie 32-bity) determinuje w jak duże słowa grupowane są dane (np. w słowa 32-bitowe). Z tym zaś wiąże się pojęcie organizacji pamięci, czyli przydzielania poszczególnym słowom konkretnych i unikatowych adresów fizycznych.

Podczas wykonywania programu, jednostka centralna zbiera rozkazy z pamięci, zależnie od wartości licznika rozkazów. Typowy cykl wykonania rozkazu zaczyna się pobraniem rozkazu z pamięci, później następuje dekodowanie rozkazu i jeśli to konieczne z pamięci pobierane są argumenty. Po wykonaniu rozkazu wyniki jego działania mogą zostać zachowane również w pamięci. Rozkazy mogą być wykonywane tylko i wyłącznie w pamięci operacyjnej i z jej użyciem (!). Stąd nazwa "pamięć operacyjna" - pamięć, w której wykonywane są operacje.

Pamięć operacyjna zwykle podzielona jest na dwie części. Jedną wykorzystuje system operacyjny, drugą programy użytkownika. W systemach wieloprogramowych pamięć przeznaczona na programy użytkownika musi być dalej podzielona tak, aby mogła współpracować z wieloma procesami [1]. Zadaniem tym zajmuje się właśnie system operacyjny i nosi ono nazwę **zarządzania pamięcią**.

NASTĘPNA

2. Adresowanie pamięci

W celu rozwiązania problemu związanego z faktem, że proces (składający się z rozkazów i danych) zawierać może rozkazy odwołujące się do adresów komórek pamięciowych dwóch rodzajów (adresów komórek danych oraz adresów rozkazów używanych w przypadku rozgałęzienia), pojawiło się rozgraniczenie pomiędzy adresem fizycznym i logicznym [1].

- **adres logiczny** definiowany jest jako położenie względem początku programu;
- **adres fizyczny** - jest to położenie w pamięci głównej o unikatowym numerze;

Realizacja przez procesor procesu wiąże się z konwersją adresów logicznych na fizyczne. Do każdego adresu logicznego w procesie dodawane jest aktualne położenie początku procesu - **adres bazowy**.

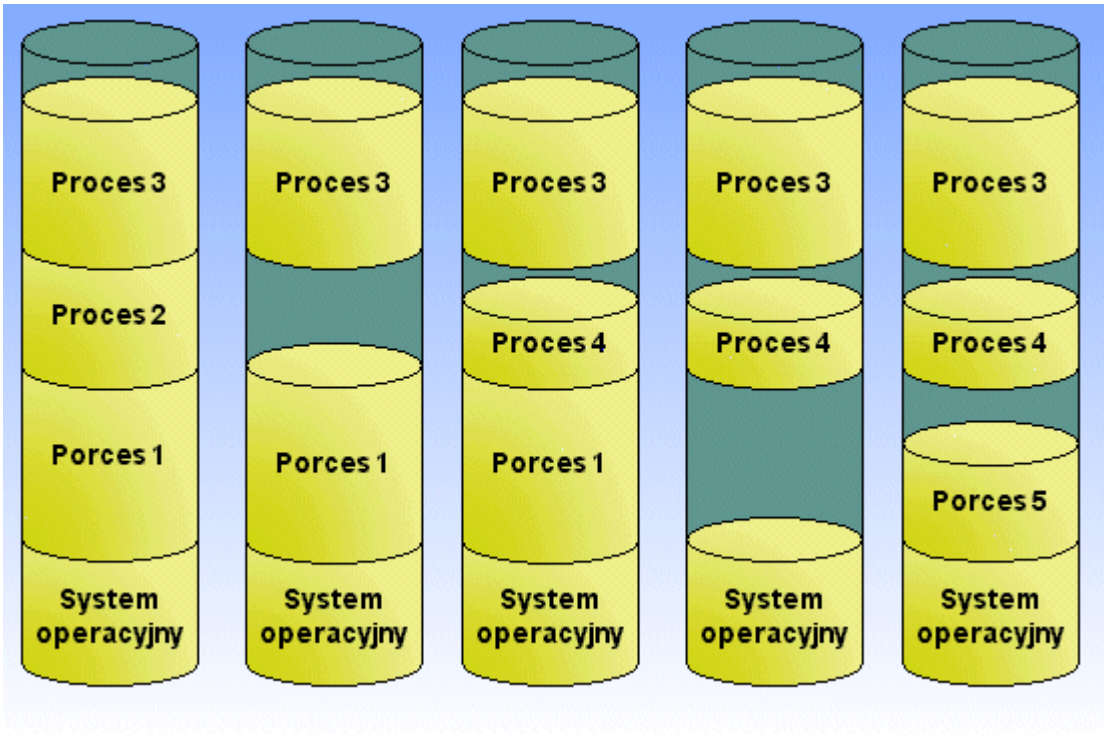
(2.1) Partycjonowanie

Terminem **partycjonowanie** określa się podział pamięci w celu udostępnienia jej wielu procesom [1]. Partycjonowaniu podlega ta część pamięci, której nie zajmuje system operacyjny. Wyróżniamy dwa rodzaje partycjonowania, którym odpowiadają dwa rodzaje partycji:

1. Partycje o **ustalonym rozmiarze** - wprowadzany do pamięci proces umieszczany jest w najmniejszej partycji, w której jest w stanie się pomieścić. Wyjaśnijmy, iż termin "ustalona" nie oznacza, że rozmiary partycji są równe, wręcz przeciwnie - nie są. Takie rozwiązanie sprawi jednak, że pojemność pamięci będzie niecałkowicie wykorzystana, ponieważ mało prawdopodobne jest zdarzenie (ale możliwe), w którym rozmiar procesu będzie równy rozmiarowi przydzielonej mu partycji.

2. Partycje o **zmiennych rozmiarach** - proces wprowadzany do pamięci jest umieszczany w partycji, o dokładnie takiej pojemności, jakiej ten proces wymaga. Metoda ta jest bardziej efektywna od opisanej powyżej.

Z partycjonowaniem wiążą się pewne problemy. Rozważmy na przykład sytuację przedstawioną na rysunku 4.1.



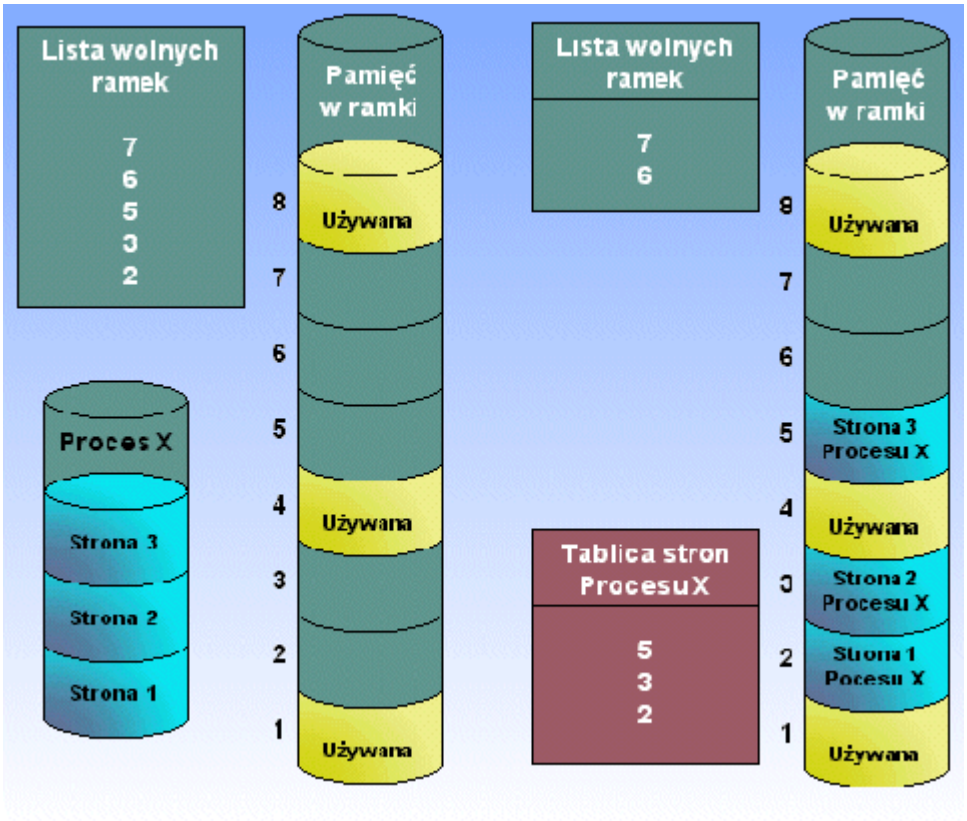
Rysunek 4.1. Hipotetyczny wynik partycjonowania [1]

Na początku proces 2 zostaje usunięty z pamięci, na jego miejsce wchodzi proces 4, następnie wychodzi proces 1 i wchodzi proces 5. Na przykładzie tym doskonale widać, jak schemat partycjonowania doprowadził do powstania luk w pamięci. Zauważmy, że im dłużej trwa partycjonowanie, tym więcej powstaje takich luk i tym mniejsze procesy mogą zostać do pamięci wprowadzane. Takie postępowanie z pewnością doprowadziłoby do zapełnienia pamięci, bez jednoczesnego wykorzystania całej jej pojemności. Dlatego też stosuje się **upakowywanie** (ang. *compaction*) , czyli okresowe przesuwanie przez system operacyjny procesów w pamięci, w celu skumulowania wolnej pamięci w jednym bloku [1].

(2.2) Stronicowanie

Wydajniejszym sposobem przydziału pamięci jest **stronicowanie**. Stronicowanie opiera się na koncepcji podziału pamięci na małe fragmenty, zwane **ramkami**, przy równoczesnym podziale procesów na takie same fragmenty - zwane **stronami**. Przy takim schemacie tracona pojemność stanowi jedynie część ostatniej strony. Jeżeli przypomnimy sobie, że strony mają małe rozmiary, zorientujemy się, że tracona pojemność również jest niewielkich rozmiarów.

Zasadę działania stron i ramek ilustruje rysunek 4.2., który przedstawia pewien jeszcze mało znany Proces X składający się z trzech stron. Gdy proces ów wczytywany jest do pamięci, system operacyjny dysponując listą wolnych ramek (którą sam utrzymuje), potrafi w odpowiedni sposób załadować strony do wolnych, niekoniecznie sąsiadujących, ramek.



Rysunek 4.2. Hipotetyczny wynik partycjonowania [1]

Ale skąd proces ma wiedzieć, w których ramkach znajdują się jego strony? Odpowiedź jest bardzo prosta. System tworzy dla każdego procesu **tablicę stron**. Pokazuje on gdzie są ramki pasujące do stron procesu. W przypadku stronicowania, tłumaczenie przez procesor adresów logicznych (numer strony, adres względny), umożliwia właśnie tablica stron. Procesor uzyskując do niej dostęp może, znając adres logiczny, utworzyć adres fizyczny (numer ramki, adres względny) [1].

Stronicowanie zapewnia bardzo efektywne wykorzystanie pamięci operacyjnej przy minimalnych stratach. Ułatwiony jest również dostęp do pamięci.

(2.3) Stronicowanie na żadanie

Pamięć wirtualna jest to pamięć znajdująca się na części obszaru dysku, jest ona antonimem dla pamięci głównej, czyli rzeczywistej (rzeczywista, gdyż proces tylko w tej pamięci może być wykonywany). Pamięć wirtualna wykorzystuje schemat **stronicowania na żądanie**, które tym różni się od omówionego wcześniej *stronicowania*, że strony procesu są wprowadzane do pamięci dopiero wówczas, gdy jest to konieczne tj. na żądanie.

Schemat stronicowania na żądanie umożliwia wczytanie tylko tych stron procesu, których realizacja (a więc tym samym obecność w pamięci operacyjnej) jest konieczna. Pozostałe strony procesu mogą przebywać w pamięci pomocniczej. W przypadku, gdy program rozgałęzi się do rozkazu znajdującego się poza pamięcią lub nastąpi odwołanie do danych spoza obszaru pamięci, sygnalizowany jest **błąd strony**, co zmusza system do dostarczenia wymaganej strony.

Ograniczenie ilości stron procesu, znajdujących się w pamięci powoduje, że możliwa staje się współpraca z większą liczbą tychże. Ponadto nieużywane strony nie zajmują niepotrzebnie czasu procesora. System operacyjny musi jednak w odpowiedni sposób pilnować, aby wprowadzając do pamięci jedną stronę, wyrzucić jednocześnie inną. W przeciwnym wypadku może dojść do **szamotania** (ang. *trashing*), polegającego na wykorzystaniu większości czasu procesora na wymianę

stron, a nie na przetwarzaniu poleceń.

Technika stronicowania na żądanie pozwala na stworzenie procesu o rozmiarach większych niż pojemność pamięci głównej oraz dała programistom ogromne zaplecze - pamięć wirtualną.

(2.4) Segmentacja

Segmentacja dzieli przestrzeń adresową na segmenty o zmiennym, dynamicznym rozmiarze, do których przypisane są programy i dane, zwykle przez system operacyjny. Możliwe jest istnienie wielu segmentów programów przeznaczonych dla różnych rodzajów programów, a także wiele segmentów danych. Ponadto każdy segment może mieć ustawione prawa dostępu i użytkownika, zaś odniesienia do tej pamięci stanowi adres, na który składa się numer segmentu i adres względny.

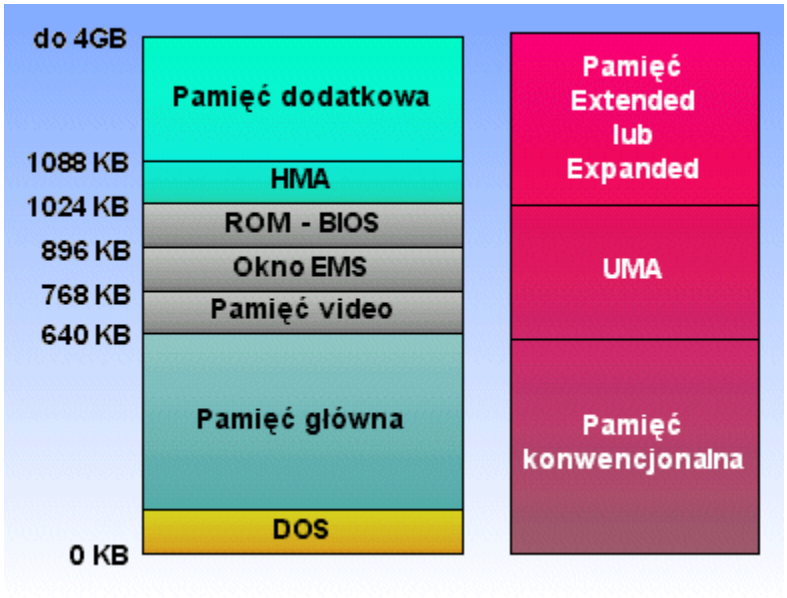
Segmentacja jest widzialna dla programisty i w przeciwieństwie do stronicowania umożliwia wygodniejsze organizowanie programów i danych oraz może być wykorzystywana jako środek kojarzenia atrybutów przywileju i ochrony z rozkazami i danymi [1]. Ponadto zapewnia min. ochronę zasobów i wykorzystanie zbioru danych przez kilka procesów. Mimo wielu zalet segmentacja nie zapewnia tak efektywnego zarządzania pamięcią. Dlatego też łączy się zalety obu schematów, stosując segmentację i stronicowanie.

NASTĘPNA

3. Mapa pamięci w MS-DOS

Podział pamięci ściśle wiąże się z wykorzystywanym systemem operacyjnym. System MS-DOS ogranicza pamięć dostępną dla programów do 640 KB. Ograniczenie to jest spowodowane architekturą pra-peceta, czyli procesora 8086, który wymuszał takie właśnie ograniczenie pamięci. Kolejne procesory potrafiły zaadresować większy obszar pamięci. Chcąc jednak zachować kompatybilność z 8086 zachowano początkowe ograniczenia.

Mapę pamięci dla systemu MS-DOS przedstawia rysunek 4.3.:



Rysunek 4.3. Mapa pamięci operacyjnej dla systemu MS-DOS

Poniżej zostaną opisane pokrótce wyżej wymienione bloki pamięci.

(3.1) Pamięć konwencjonalna

Pamięć konwencjonalna, nazywana również podstawową, to obszar położony poniżej granicy pierwszych 640 KB-ów. W obszarze tym działają wszystkie programy. W pamięci konwencjonalnej, znajduje się także część instrukcji i danych systemu operacyjnego MS-DOS, dlatego też dla programów użytkownika pozostaje mniej miejsca.

(3.2) Upper Memory Area i UMB

Wszystkie komputery kompatybilne z IBM-PC posiadają dostęp do 1 MB pamięci. MS-DOS rezerwuje 640 KB dla programów, zostają więc 384 KB nazywane UMA (ang. *Upper Memory Area*). Część tej pamięci zajmują programy BIOS'u i obsługi monitora. Pozostają jednak wolne bloki tej pamięci, nazywane UMB (ang. *Upper Memory Blocks*) [3].

Programy MS-DOS pozwalają na umieszczenie w tych górnych blokach sterowników, a nawet uruchamianie niektórych programów. W UMB znajdują się najczęściej programy rezydentne pracujące przez cały czas działania komputera.

(3.3) Pamięć rozszerzona Extended i HMA

Pamięć typu Extended znajduje się powyżej 1MB. Aby korzystać z niej w systemach MS-DOS musi być zainstalowany specjalny sterownik, który zarządza tą pamięcią i czuwa nad tym, aby np. dwa programy nie korzystały z tego samego obszaru pamięci. Program ten nosi nazwę "himem.sys".

Pierwsze 64 KB pamięci Extended stanowi pamięć HMA (ang. *High Memory Area*). Pozwala ona na umieszczanie używanych przez system tabel z danymi i buforów (min. strony kodowe z polskimi literkami) [3].

(3.4) Pamięć rozszerzona Expanded

Ta pamięć również pozwala programom na sięgnięcie poza granicę 1 MB. Pamięć Expanded jest umownie podzielona na małe, 16 KB bloki, zwane stronami. Jeśli jakiś program sięga po dane do tej pamięci, specjalny "zarządca" kopiuje całą stronę do pamięci posiadającej adres mniejszy niż 1 MB, czyli do UMA. Ze względu na zastosowane rozwiązania, pamięć tego typu jest wolniejsza i mniej efektywna od pamięci Extended [3]. W MS-DOS istnieje specjalny sterownik - "emm386.exe" - emulujący działanie tej pamięci. Sterownik ten pozwala na utworzenie 64 KB okna w pamięci UMA, poprzez które można korzystać z pamięci rozszerzonej. Okno podzielone jest na 4 strony po 16 KB każda.

NASTĘPNA

4. Proces - pojęcia podstawowe

(4.1) Proces

Przez **Proces** rozumiemy program będący w trakcie wykonywania. Wykonanie procesu musi przebiegać w sposób sekwencyjny - tzn. w danej chwili na żądanie procesu może być wykonany tylko jeden rozkaz kodu programu [2].

Na pojęcie procesu, oprócz kodu programu, składają się również: bieżąca czynność reprezentowana przez licznik rozkazów (ang. *program counter*)oraz zawartość rejestrów procesora, zwykle także stos procesu (ang. *process stack*) i sekcja danych (ang. *data section*) zawierająca zmienne globalne.

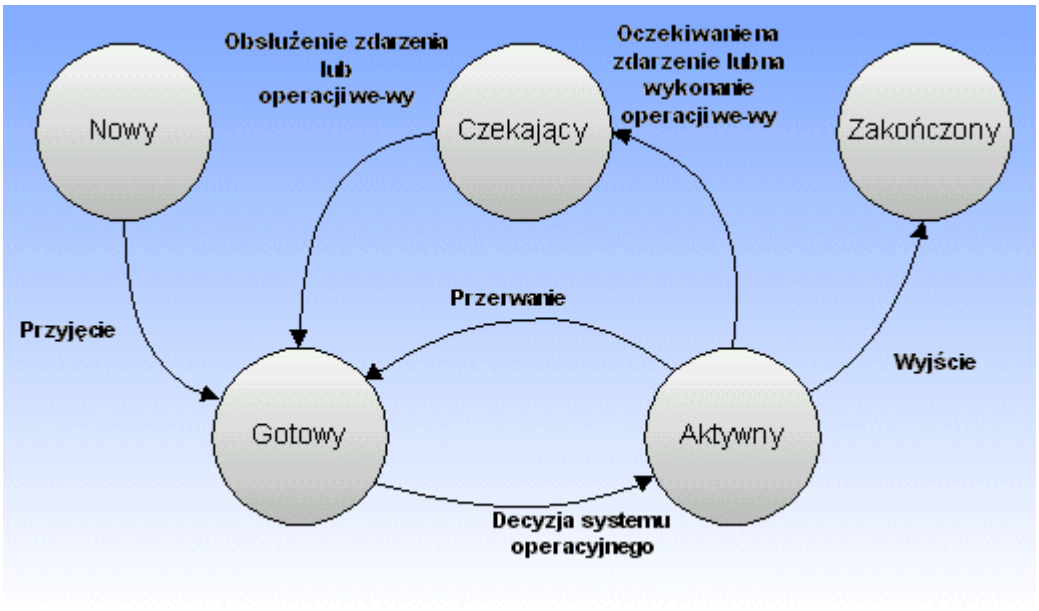
Ważne jest rozgraniczenie pojęcia programu i procesu. Pamiętajmy, że program jest obiektem pasywnym, a wszelka aktywność należy do procesów [2].

(4.2) Stan procesu

Proces, który się wykonuje zmienia również swój stan (ang. *state*). Stan ten jest częściowo określany przez to co robi proces w chwili bieżącej. Możliwe są następujące stany procesów [2]:

- **nowy** - pierwszy stan procesu
- **aktywny** - proces wykonuje instrukcje
- **oczekiwanie** - proces czeka na wystąpienie jakiegoś zdarzenia (np. zakończenie operacji wejścia-wyjścia).
- **gotowy** - proces czeka na przydział procesora
- **zakończony** - proces zakończył działanie

W różnych systemach operacyjnych stany noszą różne nazwy. W niektórych omówione stany są bardziej rozdrobnione, jednak przedstawione powyżej stany występują zawsze. Pamiętajmy, że w każdej chwili, w określonym procesorze, tylko jeden proces może być aktywny, zaś wiele procesów może być gotowych lub oczekujących.



Rysunek 4.4. Diagram stanów procesu [2].

Rysunek 4.4. przedstawia wszelkie możliwe przejścia pomiędzy poszczególnymi stanami procesu.

(4.3) Blok kontrolny procesu

Każdy proces reprezentowany jest w systemie operacyjnym przez **blok kontrolny procesu** (ang. *process control block* - PCB). Blok kontrolny procesu zawiera następujące pola [2]:

- **Stan procesu** - jeden ze stanów procesu
- **Licznik rozkazów** - adres następnego rozkazu do wykonania w procesie
- **Rejestry procesora** - Lista i typ rejestrów zależą od procesora.

Rysunek 4.5. Blok kontrolny procesu [2]

- **Informacje o planowaniu przydziału procesora** - należy do nich priorytet procesu, wskaźnik do kolejek porządkujących zamówienia i inne parametry planowania.
- **Informacje o zarządzaniu pamięcią** - zawartości rejestrów granicznych, tablice stron i segmentów.
- **Informacje do rozliczeń** - należą do nich: ilość zużytego czasu procesora i czasu rzeczywistego, ograniczenia czasowe, numery kont, numery zadań lub procesów.
- **Informacje o stanie wejścia-wyjścia** - występują tu informacje o urządzeniach I/O przydzielonych do procesu, wykaz otwartych plików, itp.

Blok kontrolny procesu jest więc magazynem przechowującym wszelkie informacje o procesie.

(4.4) Proces macierzysty i potomny

Proces może utworzyć kolejne procesy. Mówimy, że proces "stwórca" - to **proces macierzysty** (ang. *parent proces*), zaś proces przez niego utworzony - to **proces potomny** lub **potomek** (ang. *child*).

Gdy proces potomny zostaje utworzony, może stać się kopią procesu macierzystego lub otrzymać nowy program [2]. Proces macierzysty natomiast, może kontynuować działanie współbieżnie ze swoimi "potomkami", ale może też oczekiwać na zakończenie działań wszystkich, bądź niektórych swoich dzieci. Ponadto proces macierzysty może zakończyć działanie któregoś ze swoich procesów

potomnych, gdy potomek nadużyje któregoś z przydzielonych mu zasobów, wykonanie zadania przez potomka jest już zbędne, proces macierzysty zakończy się a system operacyjny nie pozwoli potomkowi na dalsze działanie [2]. Więcej informacji na ten temat znajduje się w treści kolejnego rozdziału.

NASTĘPNA

5. Nakładki

Technikę zwaną **nakładkami** (ang. *overlays*) stosuje się, aby zwiększyć wymiary procesu ponad ilość przydzielonej mu pamięci [2]. Idea nakładek opiera się na zasadzie przechowywania w pamięci tylko tych informacji, które są niezbędne. Pozostałe dane/rozkazy wprowadzane są na miejsce już (w danej chwili) niepotrzebnych.

Aby zrozumieć sens stosowania nakładek przeprowadźmy pewien eksperyment myślowy. Niech więc będzie dany program, który musi wykonać sekwencyjnie dwie operacje: Operację 1 i Operację 2. Kod tego programu zajmuje 250 KB, ale my dysponujemy jedynie przestrzenią 200 KB. Ponadto załóżmy, że na całkowity kod programu składają się:

- kod Operacji 1 : 100 KB
- kod Operacji 2 : 90 KB
- Tablica z danymi : 20 KB
- Wspólne podprogramy : 40 KB

Ponieważ operacje mają wykonać się sekwencyjnie, kody obu operacji nie muszą znajdować się w tym samym czasie w pamięci. Tworzymy więc nakładki.
Nakładka 1 to: kod Operacji 1, Tablica z danymi i Wspólne podprogramy.
Nakładka 2 to: kod Operacji 2, Tablica z danymi i Wspólne podprogramy.
Dodajemy do tego moduł obsługi nakładek (np. 10 KB) i dysponując naszymi 200 KB pamięci jesteśmy w stanie sekwencyjnie wykonać obie operacje.

Konstruowanie nakładek nie jest jednak tak proste, jak mogłoby to wynikać z powyższego eksperymentu. Do ich konstrukcji używane są specjalne algorytmy konsolidacji i przemieszczania [2]. Nakładki nie wymagają wsparcia od systemu operacyjnego, jednakże stworzenie nakładki na duży program na pewno nie jest zadaniem trywialnym.

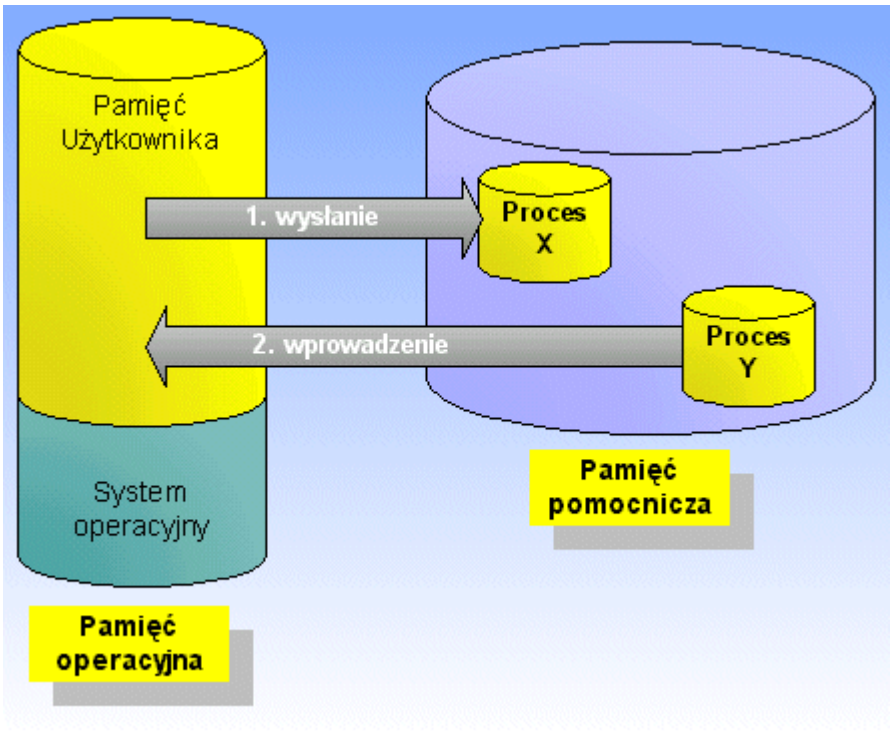
Warto wspomnieć, że w chwili obecnej, użycie nakładek ogranicza się głównie do systemów o ograniczonej pamięci fizycznej, których środki sprzętowe nie pozwalają na zastosowanie bardziej zaawansowanych technik.

NASTĘPNA

6. Swapping

Współczesne systemy operacyjne (a także sprzęt komputerowy) pozwalają na implementację zaawansowanych metod zarządzania pamięcią. Do tej grupy zalicza się **wymiana** (ang. *swapping*). Wymieniany jest proces znajdujący się w pamięci operacyjnej (wykonywany), na proces znajdujący się w pamięci pomocniczej, zaś proces znajdujący się w pamięci pomocniczej, wymieniany jest na proces z pamięci operacyjnej. Przy czym proces może być wysłany do pamięci pomocniczej, a następnie przywołany z powrotem w celu kontynuowania działania [2].

Dla środowiska sterowanego rotacyjnym algorytmem planowania przydziału procesora cykl wymiany wygląda następująco:



Rysunek 4.6. Wymiana dwóch procesów pomiędzy pamięcią operacyjną a pomocniczą [2]

Po wyczerpaniu przez proces X przeznaczonego mu kwantu czasu procesora, a więc tymczasowego zakończenia działania, system operacyjny wymienia go na proces Y, któremu również przydzielany jest kwant czasu procesora. Po upływie tego czasu również ten proces zostanie wymieniony.

Przy procedurach wymiany istotną rolę odgrywa pamięć pomocnicza (ang. *backing store*). Mousi być ona dostatecznie pojemna, aby pomieścić kopie obrazów pamięci wszystkich użytkowników oraz umożliwić bezpośredni dostęp do tych obrazów pamięci. Wszystkie procesy gotowe do działania, których obrazy pamięci znajdują się w pamięci pomocniczej lub operacyjnej, przechowywane są przez system operacyjny w **kolejce procesów gotowych** (ang. *ready queue*). Za każdym razem, gdy system operacyjny chce wykonać proces, wywołuje ekspedytora. **Ekspedytor** (ang. *dispatcher*) sprawdza dostępność procesu (nazwijmy go X) w pamięci operacyjnej. Jeżeli proces (X) nie znajduje się w pamięci operacyjnej oraz brak jest wolnego miejsca w tejże pamięci, to *dispatcher* wysyła na dysk jeden z procesów znajdujących się w pamięci operacyjnej (np. Y). Na zwolnione w ten sposób miejsce wprowadzany jest potrzebny proces (X). Na koniec ekspedytor uaktualnia rejestry i przekazuje sterowanie do wprowadzonego procesu (X).

Wadą wymiany jest dość długi czas potrzebny na przełączenie zadania, w stosunku do krótkiego

czasu na ich wykonanie. Z tego względu klasyczna technika wymiany jest stosowana coraz rzadziej przez systemy operacyjne.

NASTĘPNA

7. Programy rezydentne

Pojęcie programu rezydentnego (ang *Terminate and Stay Ready* - TSR) jest ściśle związane z systemem operacyjnym MS-DOS. Programem rezydentnym nazywamy proces wykonujący swe zadania "w tle" systemu operacyjnego, niezależnie od programów działających "na pierwszym planie". W systemach Win32 konstrukcja tych programów jest nieco inna.

Funkcje rezydentne programu mogą być wywoływane poprzez:

- przerwania użytkownika;
- przerwania systemowych IRQ;
- stałe adresy procedur;
- przypadkowe odwołania systemu;

Programy rezydentne z reguły zajmują się usługami związanymi z pracą systemu operacyjnego lub zainstalowanych urządzeń wejścia-wyjścia, ale są nimi również wirusy. Programami rezydentnymi są: sterownik myszy, nakładki na system (np. popularny Norton Commander).

Drugą grupę programów, stanowią programy rezydentne systemu operacyjnego, niezbędne do jego prawidłowego działania. Do wystartowania (uruchomienia pierwszego procesu) system operacyjny wykorzystuje pamięć. W tej pamięci przez cały czas działania systemu operacyjnego musi znajdować się rdzeń systemu operacyjnego, tablice wykorzystywane przy komunikacji z urządzeniami we/wy, itd. To właśnie programy rezydentne systemu operacyjnego zajmują się wykonaniem tych zadań.

Programy rezydentne stanowiły ważną rolę w systemie MS-DOS i pierwszych systemach spod znaku Windows. Jednakże, ze względów bezpieczeństwa, w nowych systemach operacyjnych zmieniono nieco konstrukcję programów tego typu.

NASTĘPNA