10. Zarządzanie pamięcią

Wstęp

Podstawowym zadaniem każdego systemu operacyjnego jest wykonywanie programów. Przed wykonaniem, kod programu i jego dane muszą być umieszczone przynajmniej częściowo w pamięci operacyjnej. Zwykle w każdym systemie dochodzi do sytuacji, gdy zaczyna brakować pamięci dla wykonywanych programów lub nawet dla jednego programu. Z tego względu pamięć operacyjna należy do najważniejszych zasobów każdego systemu komputerowego, a zarządzanie tą pamięcią stanowi jedno z głównych zadań systemu operacyjnego.

W wykładzie 10 prezentujemy podstawowe zagadnienia dotyczące zarządzania pamięcią operacyjną. Opisujemy różne strategie zarządzania pamięcią ze szczególnych uwzględnieniem pamięci wirtualnej. Szczegółowo omawiamy strategię stronicowania na żądanie zastosowaną w systemie Linux do implementacji pamięci wirtualnej.

10.1. Podstawowe zagadnienia

Podstawowym zadaniem każdego systemu operacyjnego jest wykonywanie programów. Przed wykonaniem, kod programu i jego dane muszą być umieszczone przynajmniej częściowo w pamięci operacyjnej. Zwykle w każdym systemie dochodzi do sytuacji, gdy zaczyna brakować pamięci dla wykonywanych programów lub nawet dla jednego programu. Z tego względu pamięć operacyjna należy do najważniejszych zasobów każdego systemu komputerowego, a zarządzanie tą pamięcią stanowi jedno z głównych zadań systemu operacyjnego.

Celem zarządzania pamięcią operacyjną jest:

- przydział pamięci fizycznej poszczególnym procesom,
- odwzorowanie logicznej przestrzeni adresowej procesu na fizyczną przestrzeń adresową pamięci,
- ochrona zawartości pamięci,
- współdzielenie obszarów pamięci przez różne procesy.

Realizacja wieloprogramowości

Systemy wieloprogramowe dają możliwość jednoczesnego uruchamiania wielu programów. Każdy z działających programów musi mieć przydzielony odpowiedni obszar pamięci operacyjnej. System może to zapewnić poprzez:

- podział pamięci operacyjnej pomiędzy działające procesy,
- wymianę procesów w pamięci.

Podział pamięci na określoną liczbę obszarów o stałym rozmiarze określany jest jako **podział statyczny**. Proces dostaje kilka obszarów, których łączny rozmiar jest równy lub przekracza rozmiar procesu. Wystarczy przechowywać jedynie numery przydzielonych obszarów, gdyż ich położenie w pamięci można łatwo obliczyć na podstawie stałego rozmiaru.

Podział dynamiczny polega na przydzielaniu obszarów o zmiennym rozmiarze w zależności od potrzeb uruchamianych procesów. Proces przechowuje dwa parametry każdego z przydzielonych obszarów:

- adres początkowy (bazowy) obszaru,
- rozmiar (granicę) obszaru.

Z podziałem pamięci wiąże się nieodłącznie problem **fragmentacji** pamięci. Polega on na pozostawianiu niewykorzystanych fragmentów pamięci wewnątrz lub pomiędzy przydzielonymi obszarami.

Fragmentacja zewnętrzna występuje poza przydzielonymi obszarami o zmiennej długości, gdy pozostałe fragmenty wolnej pamięci są zbyt małe dla innych procesów.

Fragmentacja wewnętrzna występuje wewnątrz przydzielonych obszarów o stałej długości, gdy rozmiar procesu jest mniejszy niż suma przydzielonych mu obszarów. Niewykorzystany pozostaje wtedy fragment ostatniego z przydzielonych obszarów.

Wymiana procesów (ang. *swapping*) polega na okresowym przesłaniu procesu z pamięci operacyjnej do pamięci pomocniczej, aby umożliwić sprowadzenie innego procesu z pamięci pomocniczej do zwolnionego obszaru pamięci operacyjnej.

W systemach z podziałem pamięci wymiana umożliwia wykonywanie większej liczby procesów niż może pomieścić pamięć operacyjna. Powoduje to zwiększenie stopnia wieloprogramowości, co zwykle poprawia efektywność systemu. W systemach bez podziału pamięci wymiana procesów stwarza jedyną możliwość uzyskania wieloprogramowości.

Przestrzenie adresowe

Adres wytworzony przez procesor w wyniku wykonania rozkazu programu nosi nazwę **adresu logicznego** (ang. logical address). Zbiór wszystkich adresów logicznych generowanych przez program tworzy **logiczną przestrzeń adresową** procesu.

Adres fizyczny wskazuje konkretną komórkę pamięci operacyjnej. Zbiór wszystkich adresów fizycznych tworzy **fizyczną przestrzeń adresową**.

Wiązanie adresów polega na odwzorowaniu adresów logicznych generowanych w programie na adresy fizyczne w pamięci operacyjnej. Może następować w dowolnym z poniższych etapów:

- w czasie kompilacji, czyli tworzenia programu,
- w czasie ładowania programu do pamięci,
- w czasie wykonania programu.

Wiązanie podczas kompilacji wymaga dokładnej znajomości początkowego adresu, pod którym program zostanie załadowany do pamięci. Tylko w takiej sytuacji program może posługiwać się bezpośrednio adresami bezwzględnymi w pamięci. W czasie wykonywania programu adresy logiczne i fizyczne są już takie same. Wiązanie podczas kompilacji można zastosować jedynie w prostych systemach przeznaczonych dla jednego użytkownika (np. MS-DOS).

Jeśli początkowy adres w pamięci nie jest znany w czasie tworzenia programu, to wiązanie adresów musi być opóźnione do momentu ładowania programu do pamięci. Po załadowaniu program nie może być już przemieszczany w pamięci podczas wykonywania, ponieważ adresy logiczne i fizyczne są już takie same. Zmiana adresu początkowego pociąga za sobą konieczność ponownego załadowania całego programu, począwszy od nowego adresu w pamięci.

Największe możliwości swobodnego przemieszczania programu w pamięci daje wiązanie adresów w czasie wykonywania programu. Adresy logiczne muszą być jednak tłumaczone na bieżąco na adresy fizyczne, co wymaga zastosowania specjalnego sprzętu. Zajmuje się tym **jednostka zarządzania pamięcią** (ang. *memory managament unit - MMU*). Złożoność tej jednostki warunkuje wybór strategii zarządzania pamięcią.

Zwiększenie logicznej przestrzeni adresowej procesu

Ograniczenie logicznej przestrzeni adresowej procesu do rozmiaru pamięci operacyjnej stanowiło przez długi czas poważne ograniczenie w systemach operacyjnych, uniemożliwiając wykonywanie dużych programów. Opracowane zostały dwie metody rozwiązania tego problemu:

- nakładki,
- pamięć wirtualna.

Stosowanie nakładek wymaga wyróżnienia w programie kilku funkcjonalnych modułów. Program nie jest w całości wprowadzany do pamięci. W pamięci przechowywane są tylko te moduły, które są stale wykorzystywane. Pozostałe moduły, zwane nakładkami, mogą być wprowadzane zamiennie w miarę potrzeb. Stosowanie nakładek odbywa się w zasadzie bez udziału systemu operacyjnego. Programista musi zadbać o dokonanie odpowiedniego podziału swojego programu na moduły oraz zawrzeć w programie fragmenty kodu odpowiedzialne za ładowanie kolejnych nakładek w odpowiednich momentach. Rozwiązanie to jest zatem bardzo uciążliwe dla programistów. W związku z tym, nie jest obecnie stosowane w systemach operacyjnych ogólnego przeznaczenia działających na typowych komputerach. Polem dla stosowania nakładek pozostają natomiast urządzenia o specyficznym przeznaczeniu dysponujące ograniczonymi zasobami pamięci, takie jak kalkulatory programowalne, komputery kieszonkowe.

Znacznie większe możliwości i większą wygodę oferuje technika **pamięci wirtualnej**. Pamięć wirtualna umożliwia wykonywanie programów, które nie znajdują się w całości w pamięci operacyjnej. Udostępnia procesom dużą ciągłą przestrzeń adresową nieograniczoną rozmiarem fizycznej pamięci operacyjnej. Separuje w ten sposób pamięć logiczną procesu od pamięci fizycznej. System operacyjny zajmuje się odwzorowaniem pamięci wirtualnej używanej przez procesy, częściowo na pamięć operacyjną a częściowo na pamięć pomocniczą.

Ponieważ procesy nie muszą znajdować się w całości w pamięci operacyjnej, każdy z nich zajmuje mniejszy obszar tej pamięci. Dzięki temu w pamięci można umieścić więcej procesów, zwiększając stopień wieloprogramowości. Zmniejsza się liczba operacji wejścia-wyjścia związanych z ładowaniem lub wymianą procesów. W efekcie wzrasta przepustowość systemu.

10.2. Podstawowe strategie zarządzania pamięcią

Systemy bez pamięci wirtualnej wykorzystują jedną ze strategii zarządzania pamięcią wymienionych w tablicy 10.1:

- przydział ciągły pojedynczego obszaru,
- przydział ciągły wielu obszarów,
- stronicowanie,
- segmentacja,
- segmentacja stronicowana.

Tablica 10.1 Charakterystyka różnych strategii zarządzania pamięcią

Strategia	Charakterystyka
przydział ciągły pojedyńczego obszaru	1. W pamięci fizycznej wyodrębniony jest obszar dla systemu operacyjnego i obszar dla procesów użytkownika. 2. Rozmiar obszaru systemu może ulegać zmianom, więc proces przechowuje adres bazowy obszaru użytkownika. 3. Adres logiczny jest przekształcany na fizyczny przez dodanie adresu bazowego obszaru użytkownika.
przydział ciągły wielu obszarów	1. Pamięć fizyczna dzielona jest dynamicznie na obszary o zmiennym rozmiarze, przydzielane kolejnym procesom. 2. Jeden obszar zajmowany jest przez system operacyjny. 3. Pomiędzy przydzielonymi obszarami pozostają obszary wolne, zwane dziurami. 4. Każdy nowy proces dostaje tylko jeden ciągły obszar pamięci o odpowiednim rozmiarze, wybrany przez system spośród dostępnych dziur. 5. Proces przechowuje adres bazowy i granicę przydzielonego obszaru. 6. Adres logiczny określa położenie wewnątrz przydzielonego obszaru. Jest porównywany z granicą obszaru w celu sprawdzenia poprawności, a następnie dodawany do adresu bazowego obszaru.
stronicowanie	1. Pamięć fizyczna podzielona jest na obszary o stałym rozmiarze, zwane ramkami. 2. Pamięć logiczna podzielona jest na obszary o stałym rozmiarze, zwane stronami. 3. Przed wykonaniem wszystkie strony procesu muszą być umieszczone w dowolnych ramkach, które mogą tworzyć nieciągły obszar pamięci. 4. Proces przechowuje tablicę stron zawierającą adresy przydzielonych ramek. 5. Adres logiczny składa się z numeru strony i odległości na stronie. Numer strony jest indeksem pozycji w tablicy stron, która zawiera adres bazowy ramki przechowującej daną stronę procesu.
segmentacja	1. Pamięć fizyczna podzielona jest na obszary o zmiennym rozmiarze, odpowiadające segmentom logicznym procesów. 2. Pamięć logiczna podzielona jest na obszary o zmiennym rozmiarze i określonej nazwie lub numerze.

	3. Przed wykonaniem wszystkie segmenty procesu muszą być sprowadzone i rozmieszczone w pamięci operacyjnej, przy czym mogą tworzyć nieciągły obszar pamięci. 4. Proces przechowuje tablicę segmentów zawierającą adresy bazowe i granice przydzielonych segmentów. 5. Adres logiczny składa się z numeru segmentu i odległości w segmencie. Numer segmentu jest indeksem pozycji w tablicy segmentów.
segmentacja stronicowana	1. Pamięć fizyczna podzielona jest na ramki. 2. Pamięć logiczna podzielona jest na segmenty. 3. Segmenty logiczne podlegają stronicowaniu. 4. Tablica segmentów procesu zawiera adresy bazowe tablic stron poszczególnych segmentów.

Wszystkie wymienione wyżej strategie wymagają, żeby cały proces został umieszczony w pamięci operacyjnej przed rozpoczęciem wykonywania. Ograniczenie to nie występuje w systemach wykorzystujących pamięć wirtualną. Do implementacji pamięci wirtualnej może posłużyć jedna ze strategii:

- stronicowanie na żądanie,
- segmentacja na żądanie.

Większość współczesnych systemów operacyjnych wykorzystuje stronicowanie na żądanie do realizacji pamięci wirtualnej. Zostanie ona omówiona w dalszej części niniejszego wykładu wraz z przykładową implementacją w systemie Linux. Segmentację na żądanie zastosowano w systemie OS/2.

Dzielenie i ochrona pamięci

Współdzielenie obszarów pamięci przez różne procesy jest możliwe do realizacji w systemach, które przydzielają po kilka lub więcej obszarów każdemu procesowi. Wówczas obszary, które zawierają kod programu lub dane przeznaczone wyłącznie do odczytu, mogą być użytkowane wspólnie z innymi procesami. Dotyczy to strategii stronicowania i segmentacji pamięci. Określone pozycje tablic stron lub segmentów kilku procesów wskazują wtedy na te same ramki lub segmenty w pamięci operacyjnej.

Podstawowa ochrona pamięci obejmuje sprawdzanie:

- poprawności adresów,
- praw dostępu do poszczególnych obszarów pamięci, zwłaszcza obszarów współdzielonych.

Sprawdzanie poprawności adresów powinno być realizowane we wszystkich strategiach zarządzania przez sprzęt komputerowy. Opisujemy to w następnym punkcie.

W zaawansowanych strategiach zarządzania pamięcią ochrona może być powiązana ze stronami lub segmentami. Dla każdej strony lub segmentu zdefiniowany jest zestaw bitów, przechowywanych w tablicy wraz z adresem bazowym. Zestaw ten zawiera zwykle:

bit poprawności - określa, czy obszar należy do przestrzeni adresowej procesu oraz czy znajduje się obecnie w pamięci operacyjnej,

bity ochrony - opisują prawa dostępu do danego obszaru pamięci: prawo do czytania, pisania i wykonywania.

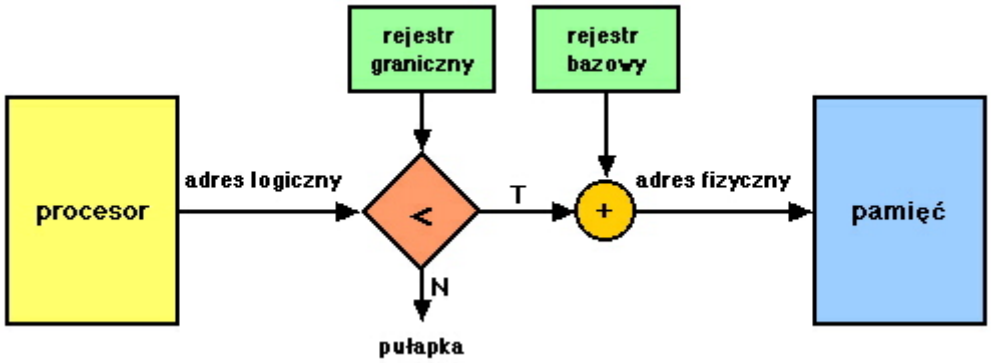
Wspomaganie sprzętowe

Wybór strategii zarządzania pamięcią operacyjną jest silnie uzależniony od możliwości sprzętu komputerowego. Wspomagania sprzętowego wymaga przede wszystkim:

- przekształcanie adresów logicznych na fizyczne,
- sprawdzanie poprawności adresu.

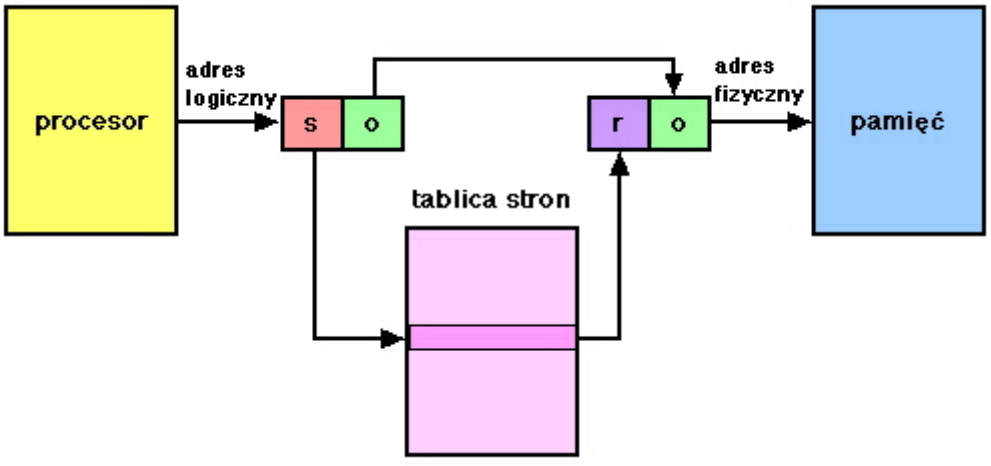
Zajmuje się tym jednostka zarządzania pamięcią MMU. Złożoność tej jednostki decyduje o możliwości wyboru strategii zarządzania pamięcią.

W najprostszym przypadku przydziału pojedynczego obszaru, potrzebny jest tylko jeden **rejestr bazowy**, przechowujący adres bazowy obszaru użytkownika. Przydział wielu obszarów wymaga już użycia dwóch rejestrów: **rejestru bazowego** i **rejestru granicznego**. Po wybraniu procesu przez planistę, dyspozytor ładuje do tych rejestrów odpowiednio adres początkowy i rozmiar przydzielonego procesowi obszaru. Porównanie adresu logicznego z wartością w rejestrze granicznym umożliwia sprawdzenie poprawności adresu, a przez dodanie wartości z rejestru bazowego uzyskuje się adres fizyczny w pamięci. Mechanizm ten ilustruje rys. 10.1. W ten sposób sprzęt realizuje wiązanie adresów i jednocześnie ochronę innych obszarów pamięci.



Rys. 10.1 Sprzęt do obsługi wielu ciągłych obszarów

Znacznie większego wsparcia sprzętowego wymaga stronicowanie pamięci. Na rys. 10.2 przedstawiono podstawowy sprzęt stronicujący.

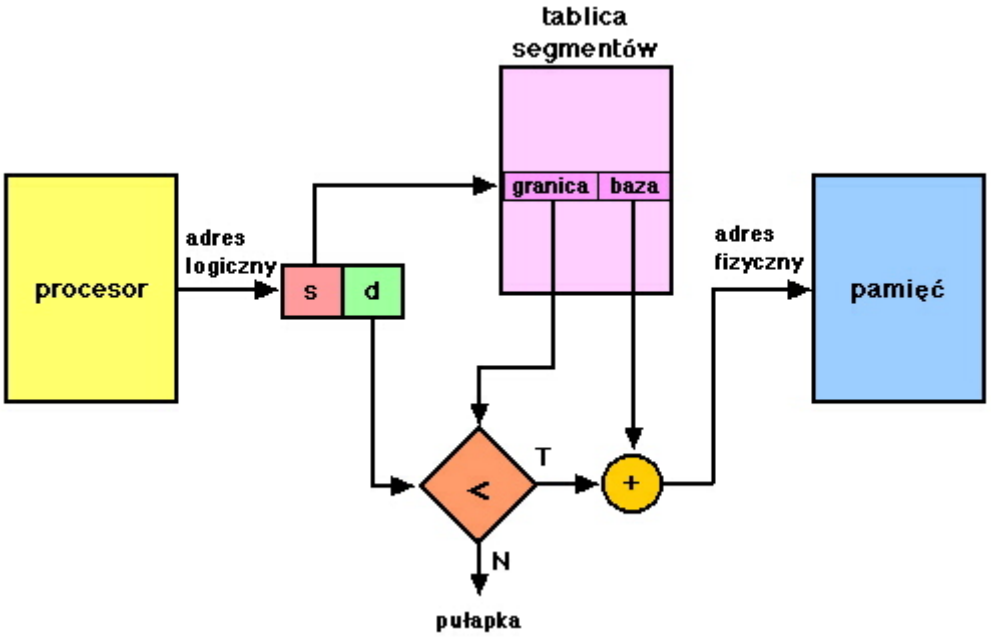


Rys. 10.2 Sprzęt do stronicowania

Podczas przełączania kontekstu procesu, jego tablica stron musi być załadowana do sprzętowej tablicy stron systemu. Adres logiczny składa się z numeru strony **s** i odległości na stronie **o**. Numer strony wybiera odpowiednią pozycję w tablicy stron, zawierającą adres bazowy ramki w pamięci fizycznej. Sprzęt stronicujący sprawdza ponadto bit poprawności, aby stwierdzić, czy strona o poda-

nym numerze należy do logicznej przestrzeni adresowej procesu i czy jest sprowadzona do pamięci operacyjnej. Odległość na stronie nie wymaga sprawdzania, gdyż rozmiar tej części adresu jest odpowiednio dobrany do rozmiaru strony (i ramki). Przykładowo, przy rozmiarze adresu N-bitów i rozmiarze strony 2^n słów, numer strony zajmuje *N-n* bardziej znaczących bitów, a odległość na stronie - *n* mniej znaczących bitów.

Najbardziej złożony sprzęt, przedstawiony na rys. 10.3, jest potrzebny do realizacji segmentacji pamieci.



Rys. 10.3 Sprzęt do segmentacji

Podczas przełączania kontekstu procesu, jego tablica segmentów musi być załadowana do sprzętowej tablicy segmentów systemu. Adres logiczny zawiera numer segmentu **s**, który jest indeksem w tablicy segmentów. Wybrana pozycja tablicy zawiera adres bazowy i granicę segmentu w pamięci fizycznej. Zadaniem sprzętu jest jeszcze porównanie odległości **d** w segmencie z jego granicą oraz dodanie odległości do adresu bazowego segmentu, aby otrzymać adres fizyczny.

Sprzętowa tablica stron lub segmentów może być zrealizowana na kilka sposobów:

1. zbiór rejestrów specjalnego przeznaczenia, przechowujących całą tablicę stron procesu,
2. rejestr bazowy tablicy stron wskazujący jej położenie w pamięci operacyjnej,
3. rejestry asocjacyjne, zwane również buforami translacji bliskiego otoczenia TLB, przechowujące tylko wybrane pozycje tablicy stron i uzupełniane w miarę potrzeb z tablicy stron w pamięci.

W pierwszym przypadku cała tablica stron procesu jest ładowana do rejestrów procesora podczas przełączania kontekstu. Dostęp do tablicy jest bardzo szybki, ale rozmiar rejestrów ogranicza rozmiar tablicy stron.

W drugim przypadku tablica stron jest przechowywana w pamięci. Uzupełnienie stanowią dwa rejestry:

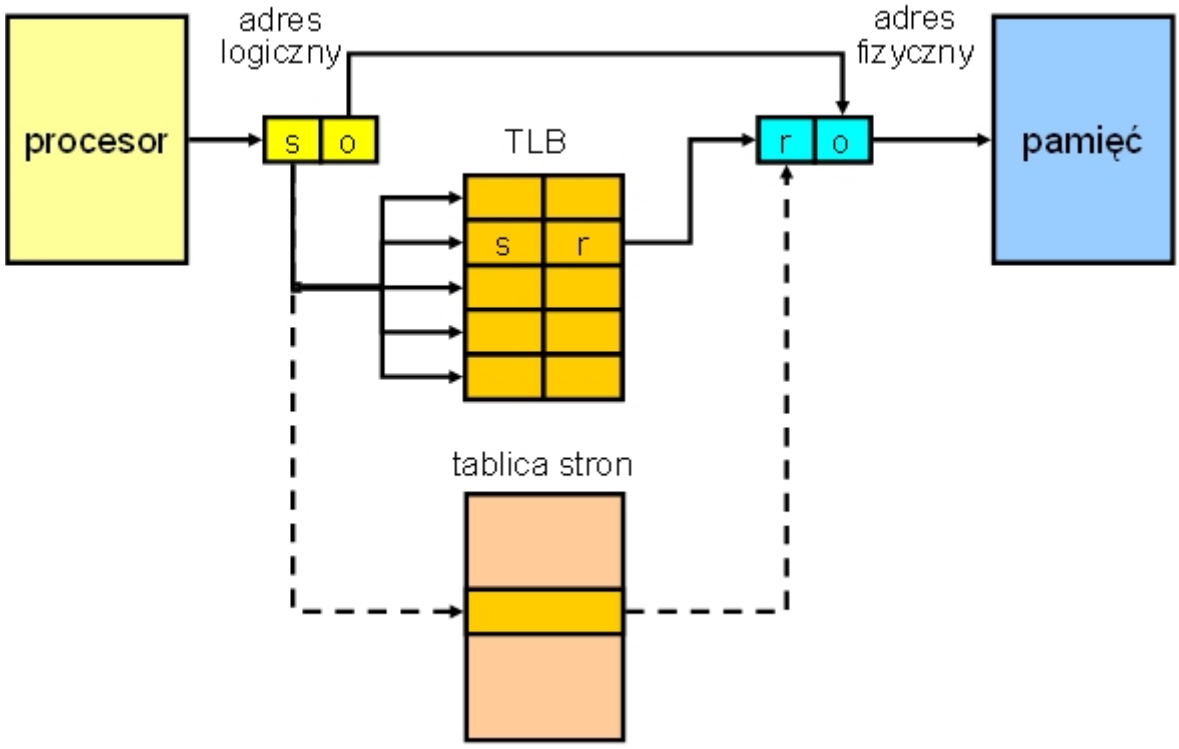
- rejestr bazowy tablicy stron PTBR wskazuje położenie tablicy w pamięci operacyjnej,
- rejestr graniczny tablicy stron PTLR przechowuje rozmiar tablicy.

Takie rozwiązanie znacząco wydłuża czas dostępu do danych, ponieważ konieczne są dwie operacje dostępu do pamięci: pobranie adresu ramki z tablicy stron i pobranie instrukcji lub danych programu.

Rozwiązaniem pośrednim jest zastosowanie rejestrów asocjacyjnych (Rys. 10.4). Przechowują one tylko wybrane pozycje z tablicy stron. Każdy rejestr składa się z dwóch części:

- klucza, przechowującego numer strony w pamięci logicznej,
- wartości, określającej numer przydzielonej ramki.

Bieżący adres logiczny porównywany jednocześnie ze wszystkimi kluczami, co daje możliwość szybkiego odszukania adresu fizycznego. Brakujące adresy odnajdowane są w tablicy stron w pamięci i wpisywane do rejestrów.



Rys. 10.4 Sprzęt stronicujący z rejestrami asocjacyjnymi

10.3. Pamięć wirtualna

W większości współczesnych systemów operacyjnych, w tym w systemie Linux, stosuje się strategię stronicowania na żądanie do realizacji pamięci wirtualnej. Punktem wyjścia dla takiej strategii było stronicowanie pamięci z wymianą całych procesów. Podstawowa zmiana polega na tym, że zamiast wymieniać całe procesy pomiędzy pamięcią operacyjną i pomocniczą, system wymienia w miarę potrzeb tylko pojedyncze strony tych procesów.

Stronicowanie na żądanie wykorzystuje następujące mechanizmy:

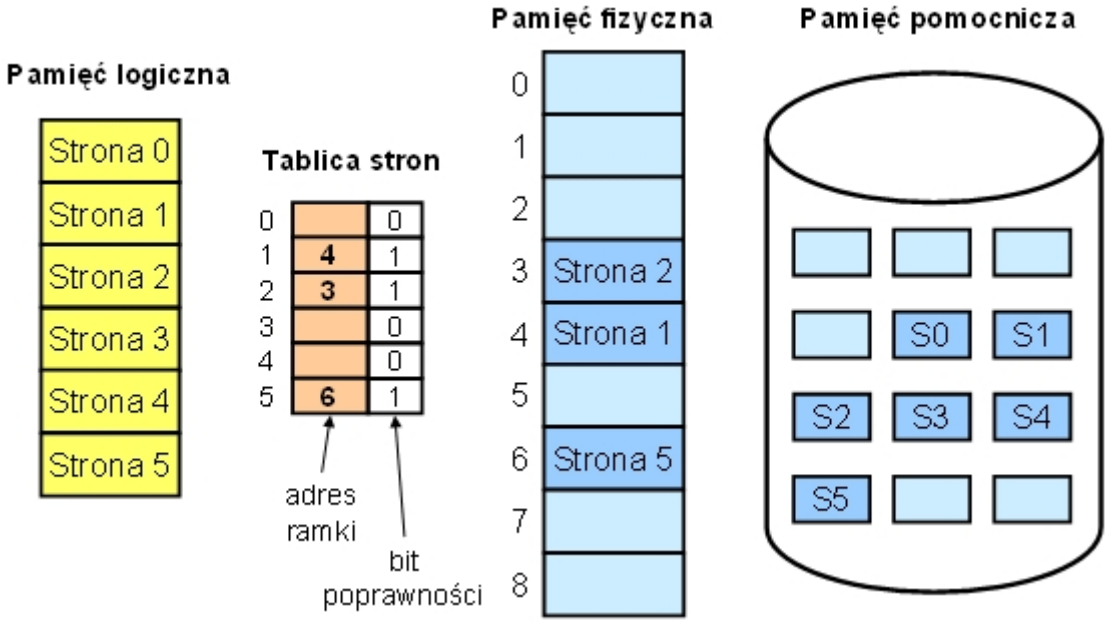
- stronicowanie pamięci z możliwością wykonywania procesu wprowadzonego tylko częściowo do pamięci operacyjnej,
- sprowadzanie brakującej strony do pamięci operacyjnej na żądanie procesu, gdy wystąpi odwołanie do tej strony,
- zastępowanie stron, gdy brakuje wolnych ramek pamięci na nowe strony potrzebne procesowi,
- wymiana całych procesów jako mechanizm uzupełniający.

Model stronicowania na żądanie

Pamięć fizyczna podzielona jest na ramki, czyli obszary o ustalonym rozmiarze. Rozmiar ramki jest uzależniony od platformy sprzętowej i zwykle wynosi 4096 B (4 KB) lub 8192 B (8 KB).

Pamięć logiczna procesu, zwana również pamięcią wirtualną, podzielona jest na strony o tym samym rozmiarze. W pamięci wirtualnej musi być odwzorowany kod programu wykonywanego przez proces. Cały kod nie musi jednak znajdować się w pamięci operacyjnej. Wystarczy, że znajdą się tam tylko niektóre strony zawierające początek kodu, aby proces mógł rozpocząć wykonywanie programu. Strony zawierające pozostałą część kodu oraz inne dane będą sprowadzane do pamięci operacyjnej, gdy proces będzie próbował się do nich odnosić.

Pamięć pomocnicza jest również podzielona na obszary o rozmiarze strony i przeznaczona do przechowywania stron czasowo usuniętych z pamięci operacyjnej w wyniku zastępowania. Pamięć pomocnicza jest określana jako **przestrzeń wymiany** i może być zrealizowana na urządzeniu wymiany lub w pliku wymiany. Rys. 10.5 przedstawia model stronicowania na żądanie.



Rys. 10.5 Model stronicowania na żądanie

Oprócz adresu bazowego ramki, każda pozycja w tablicy stron zawiera bity przechowujące dodatkowe informacje o stronie, m.in.:

bit poprawności - określa, czy strona znajduje się w pamięci operacyjnej,

bit modyfikacji - określa, czy strona była modyfikowana od momentu sprowadzenia do pamięci operacyjnej,

bit odniesienia - określa, czy nastąpiło ostatnio jakieś odniesienie do strony,

bity ochrony - opisują prawa dostępu do danego obszaru pamięci: prawo do czytania, pisania i wykonywania,

wiek strony - określa, kiedy strona została sprowadzona do pamięci.

Sprowadzanie stron na żądanie

Pamięć wirtualna daje możliwość wykonywania procesu, który nie znajduje się w całości w pamięci operacyjnej. Stwarza to jednak możliwość odwołania do strony, która nie została jeszcze sprowadzona do pamięci. Sytuację taką wykrywa sprzęt stronicujący na podstawie bitu poprawności w tablicy stron. Generowany jest wtedy błąd braku strony, czyli pułapka. Błąd taki musi być niezwłocznie obsłużony przez jądro systemu.

Obsługa błędu braku strony obejmuje następujące czynności:

1. zapamiętanie kontekstu procesu,
2. sprawdzenie poprawności adresu wirtualnego, ewentualne wysłanie sygnału SIGSEGV do procesu,
3. sprawdzenie, czy typ dostępu do strony był dozwolony, ewentualne wysłanie sygnału SIGSEGV do procesu,
4. znalezienie i przydział wolnej ramki,
5. sprowadzenie potrzebnej strony do przydzielonej ramki,
6. modyfikacja odpowiedniej pozycji w tablicy stron,
7. wznowienie wykonania instrukcji kodu programu.

Zastępowanie stron

Brakująca strona może zostać sprowadzona bezpośrednio z pliku na dysku lub z przestrzeni wymiany na dysku (z pamięci pomocniczej), jeśli była już wcześniej używana i została zastąpiona.

Przed sprowadzeniem strony na żądanie system musi przydzielić procesowi nową ramkę. Jeżeli brakuje wolnych ramek pamięci, to pojawia się konieczność wykorzystania jednej z ramek wykorzystywanych dotychczas i zastąpienia przechowywanej strony nową stroną.

Zastępowanie stron obejmuje następujące czynności:

1. znalezienie ofiary, czyli ramki, której zawartość zostanie zastąpiona,
2. wysłanie przechowywanej w ramce strony do przestrzeni wymiany, jeżeli strona była modyfikowana (ustawiony bit modyfikacji),
3. modyfikacja odpowiedniej pozycji w tablicy stron procesu, do którego należała strona,
4. sprowadzenie potrzebnej strony z pamięci pomocniczej do zwolnionej ramki,
5. modyfikacja odpowiedniej pozycji w tablicy stron procesu, dla którego sprowadzono stronę.

Najtrudniejszym etapem jest wybór ramki-ofiary. Opracowano w tym celu wiele algorytmów zastępowania stron. Podstawowym kryterium wyboru algorytmu jest zwykle minimalizacja częstości błę-

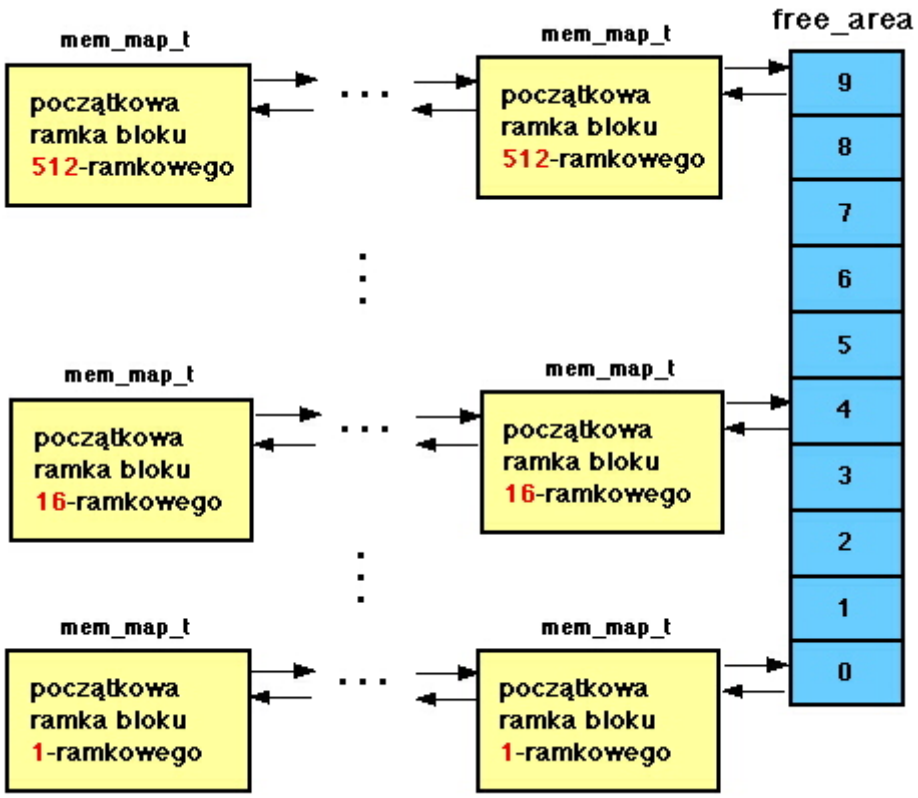
dów braku strony. Należy również brać pod uwagę możliwość implementacji wybranego algorytmu na konkretnej platformie sprzętowej.

10.4. Stronicowanie na żądanie w systemie Linux

Przydział i zwalnianie ramek

Wszystkie ramki pamięci fizycznej w systemie opisane są przez listę struktur **struct page** (typ **mem_map_t)** inicjowaną w czasie startu systemu. Każda struktura **page** reprezentuje pojedynczą ramkę pamięci.

Informacje o wolnych ramkach przechowywane są w tablicy **free_area**[]. Sąsiednie wolne ramki grupowane są w bloki o rozmiarze będącym potęgą liczby 2. Element tablicy **free_area**[i] wskazuje listę bloków o rozmiarze 2^i ramek, gdzie i=0, 1,..., N-1. Zatem element **free_area**[0] wskazuje listę bloków składających się z jednej ramki, element **free_area**[1] - listę bloków składających się z dwóch ramek, element **free_area**[2] - listę bloków składających się z czterech ramek, itd. Rozmiar tablicy wynosi N=10 na większości platform sprzętowych, co oznacza że największe bloki mogą składać się z 2^9 ramek. Ze względu na to, że ramki w bloku tworzą ciągły obszar pamięci, każdy blok jest reprezentowany na liście tylko przez początkową ramkę. Rysunek 10.6 ilustruje opisany sposób reprezentacji wolnych ramek.



Rys. 10.6 Reprezentacja wolnych ramek

Przydziałem i zwalnianiem ramek zajmuje się podprogram jądra nazywany dyspozytorem ramek. Wykorzystuje w tym celu **algorytm bliźniaków** (ang. *buddy algorithm*). Dyspozytor przydziela procesom ciągłe bloki pamięci składające się z 2^i ramek.

Po otrzymaniu zlecenia przydziału **n** ramek pamięci, dyspozytor przegląda tablicę **free_area**[] w celu znalezienia najmniejszego bloku o rozmiarze nie mniejszym od zamówionego 2^i >=n. Najpierw poszukuje bloku o rozmiarze dokładnie pasującym do zamówienia, potem bloku dwukrotnie większego i kolejnych. Jeśli znajdzie wolny blok o rozmiarze 2^{i+1} ramek, to dokonuje podziału na dwa mniejsze bloki o rozmiarze 2^i . Jeden z tych bloków przydziela procesowi, a drugi dołącza do listy wolnych bloków wskazywanych przez **free_area**[i].

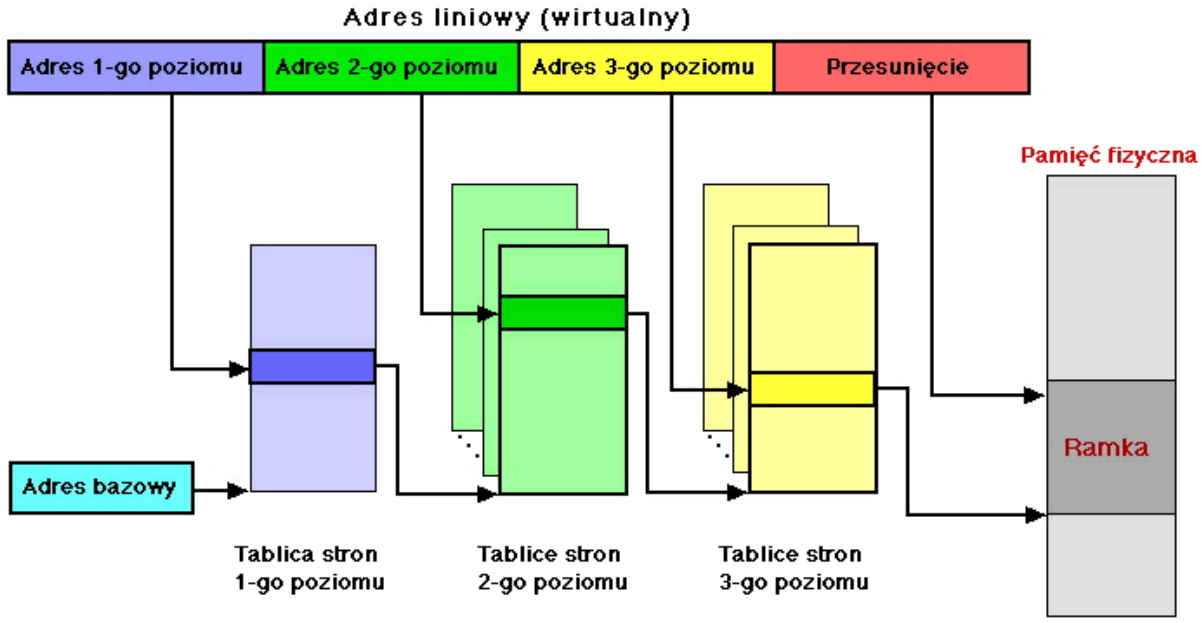
Zwalniane bloki ramek są dołączane do odpowiednich list w tablicy **free_area**[]. Dyspozytor stara się przy tym łączyć każde dwa sąsiednie bloki ramek o rozmiarze 2^i w jeden większy blok o rozmiarze 2^{i+1} .

Tablice stron i wiązanie adresów

W zależności od platformy sprzętowej Linux dopuszcza adresowanie pamięci 32-bitowe (np. i386) lub 64-bitowe (np. x86_64, PowerPC, Alpha). Rozmiar wirtualnej przestrzeni adresowej wynosi wtedy odpowiednio 2^{32} B (4 GB) lub 2^{64} B. Dla tak dużej przestrzeni adresowej potrzebna jest wielka tablica stron. Już przy 32-bitowym adresie i stronie 4 KB tablica stron powinna zawierać 2^{20} pozycji. Przechowywanie takiej tablicy w pamięci wymaga zaalokowania dużego ciągłego obszaru.

Z tego powodu system Linux stosuje stronicowanie wielopoziomowe, dzieląc tablicę stron na mniejsze części. Rysunek 10.7 przedstawia 3-poziomową reprezentację tablicy stron procesu. Adres bazowy tablicy stron wskazuje na tablicę 1-go poziomu, nazywaną katalogiem stron. Każda z pozycji w katalogu wskazuje na jedną z tablic 2-go poziomu, określaną jako pośredni katalog stron. Pozycja w tym katalogu wskazuje na jedną z tablic 3-go poziomu, nazywaną po prostu tablicą stron. Nie wszystkie platformy sprzętowe wykorzystują 3 poziomy stronicowania. Przy adresowaniu 32-bitowym wystarcza na ogół tablica 2-poziomowa, w której pomija się katalogi pośrednie. Każda pozycja w katalogu stron interpretowana jest wtedy jako pośredni katalog stron o rozmiarze 1.

Adres wirtualny zawiera numer strony w przestrzeni wirtualnej procesu i odległość na stronie. Jednostka MMU procesora przekształca ten adres na adres liniowy, wyróżniając odpowiednią liczbę poziomów adresu (2 lub 3). Następnie adres liniowy przekształcany jest na adres fizyczny w sposób przedstawiony na rysunku 10.7.



Rys. 10.7 Przekształcanie adresu linowego na adres fizyczny

Adres 1-go poziomu wybiera pozycję w katalogu stron (tablicy 1-go poziomu), czyli jeden z katalogów pośrednich. Adres 2-go poziomu wybiera pozycję w katalogu pośrednim, wskazującą jedną z tablic stron. Adres 3-go poziomu wybiera już konkretną stronę w tablicy stron. Uzyskuje się w ten sposób adres bazowy ramki w pamięci operacyjnej, a odległość na stronie wskazuje konkretny bajt.

Algorytm zastępowanie stron

Demon wymiany stron **kswapd** jest uruchamiany przez proces **init** w czasie startu systemu jako wątek jądra, który działa w fizycznej przestrzeni adresowej jądra systemu. Większość czasu spędza w uśpieniu, ale jest budzony przez dyspozytora ramek, gdy ilość wolnej pamięci w systemie spadnie poniżej dozwolonego poziomu (liczba wolnych ramek spadnie poniżej progu **pages_low**).

W takiej sytuacji demon kswapd stara się zwolnić tyle ramek, żeby liczba wolnych ramek w systemie osiągnęła **pages_high**.

System Linux stosuje obecnie udoskonaloną wersję algorytmu LRU (Split LRU) do zastępowania stron. Funkcja realizująca klasyczny algorytm LRU sprawdza kolejno ramki wszystkich procesów i wybiera do zastąpienia stronę najdawniej używaną. W zmodyfikowanej wersji Split LRU wprowadza się rozróżnienie na strony przechowujące dane procesów i strony przechowujące dane systemu plików (pamięć podręczna), aby zastosować do nich różne algorytmy wyboru strony.

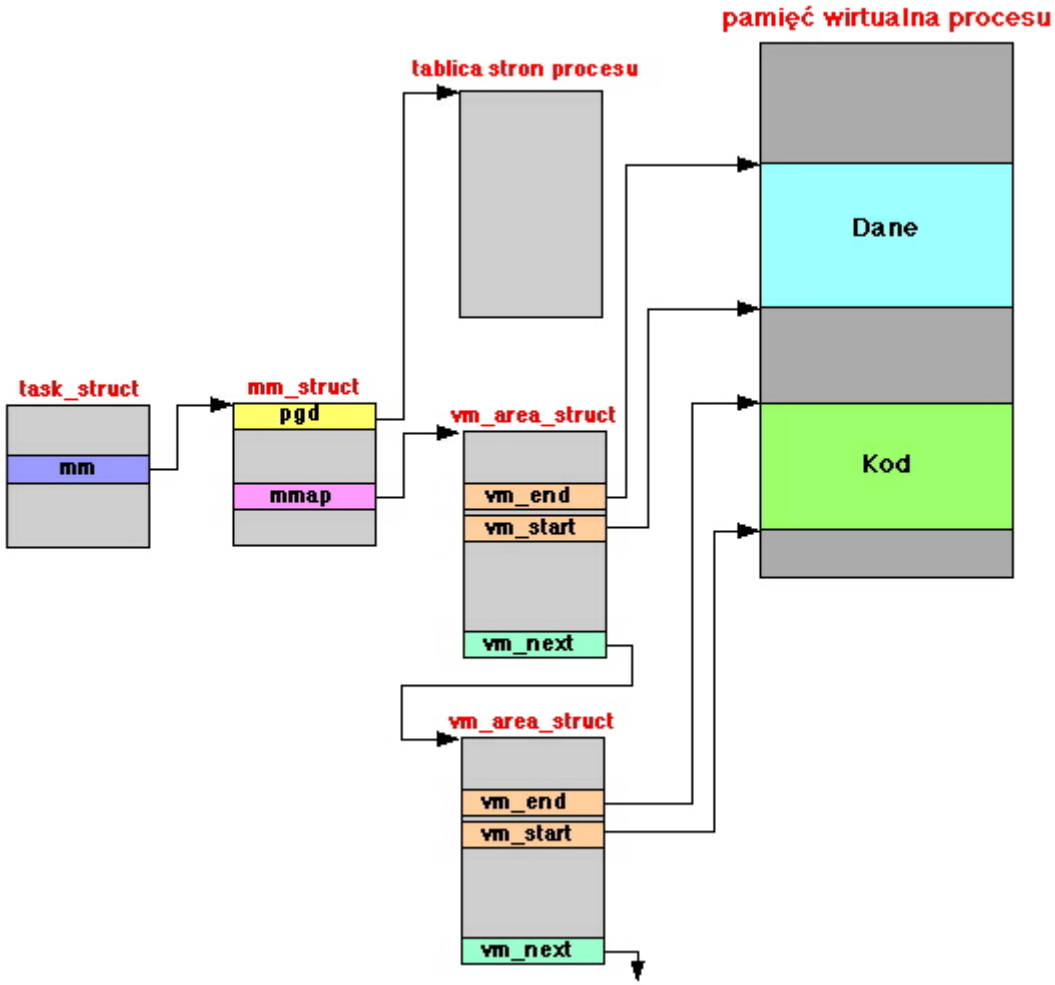
Funkcja zastępowania stron może być wołana w dwóch sytuacjach:

1. przez demona **kswpd** po stwierdzeniu, że liczba wolnych ramek w systemie jest zbyt mała,
2. bezpośrednio przez dyspozytora ramek, gdy zabrakło wolnych ramek do przydziału a kswapd nie nadąża z ich zwalnianiem.

Wybrana strona musi być wysłana do przestrzeni wymiany tylko wtedy, gdy była modyfikowana w pamięci i ma ustawiony bit modyfikacji. W przeciwnym przypadku strona może być zapomniana, ponieważ można ją ponownie odczytać z pliku lub z przestrzeni wymiany, gdzie była zapisana wcześniej. Modyfikacji wymaga jeszcze pozycja strony w tablicy stron procesu. Zerowany jest bit poprawności strony, a zamiast adresu ramki może być zapisany wskaźnik położenia strony w przestrzeni wymiany.

10.5. Wirtualna przestrzeń adresowa procesu w systemie Linux

Każdy proces w systemie dysponuje niezależną wirtualną przestrzenią adresową. Jej rozmiar jest uzależniony od rozmiaru adresu w konkretnej implementacji systemu Linux. Rysunek 10.8 przedstawia strukturę pamięci wirtualnej procesu.



Rys. 10.8 Struktura pamięci wirtualnej procesu

Zawartość pamięci wirtualnej procesu opisuje struktura **mm_struct**. Jądro utrzymuje równolegle dwie reprezentacje tej pamięci:

1. podział na strony opisany w wielopoziomowej tablicy stron procesu,
2. podział na obszary reprezentowane przez struktury **vm_area_struct**.

Każda struktura **vm_area_struct** zawiera m.in.:

- adresy początku i końca obszaru w pamięci wirtualnej,
- i-węzeł pliku odwzorowanego na ten obszar,
- atrybuty ochrony obszaru,
- wskaźniki umożliwiające powiązanie struktur wszystkich obszarów na dwa sposoby:
    1. w listę jednokierunkową,
    2. w drzewo RB (ang. *red-black tree*), umożliwiające szybkie przeszukiwanie.

Bibliografia

[1] Silberschatz A., Galvin P.B.: Podstawy systemów operacyjnych, WNT 2000

[2] Glass G., Ables K.: Linux dla programistów i użytkowników, Wydawnictwo Helion 2007 (rozdział 13.4)

Słownik

Termin	Objaśnienie
adres bazowy	adres początkowy obszaru w pamięci operacyjnej
adres fizyczny	wskazanie konkretnej komórki pamięci operacyjnej
adres logiczny	adres wytworzony przez procesor w wyniku wykonania rozkazu programu
dynamiczny podział pamięci	podział pamięci na obszary o zmiennym rozmiarze w zależności od potrzeb uruchamianych procesów
fizyczna przestrzeń adresowa	zbiór wszystkich adresów fizycznych
fragmentacja pamięci	pozostawianie niewykorzystanych fragmentów pamięci wewnątrz lub pomiędzy przydzielonymi obszarami
fragmentacja wewnętrzna	pozostawianie wolnych fragmentów pamięci wewnątrz przydzielonych obszarów o stałej długości, gdy rozmiar procesu jest mniejszy niż suma przydzielonych mu obszarów
fragmentacja zewnętrzna	pozostawianie pomiędzy przydzielonymi obszarami wolnych fragmentów pamięci, które są zbyt małe dla innych procesów
granica obszaru	długość obszaru pamięci
logiczna przestrzeń adresowa	zbiór wszystkich adresów logicznych generowanych przez proces
pamięć wirtualna	technika umożliwiająca wykonywanie programów, które nie znajdują się w całości w pamięci operacyjnej poprzez udostępnienie procesom dużej ciągłej przestrzeni adresowej nieograniczonej rozmiarem fizycznej pamięci operacyjnej
segmentacja pamięci	strategia zarządzania pamięcią operacyjną polegająca na niecią-głym przydziale wielu obszarów o zmiennym rozmiarze
statyczny podział pamięci	podział pamięci na określoną liczbę obszarów o stałym rozmiarze
stronicowanie pamięci	strategia zarządzania pamięcią operacyjną polegająca na niecią-głym przydziale wielu obszarów o stałym rozmiarze
wiązanie adresów	odwzorowanie adresów logicznych na adresy fizyczne
wymiana procesów	okresowe przesłanie procesu z pamięci operacyjnej do pamięci pomocniczej, aby umożliwić sprowadzenie innego procesu z pamięci pomocniczej do zwolnionego obszaru pamięci operacyjnej