Systemy operacyjne

Ćwiczenia 5

Zadanie 1

Zmienne w programie źródłowym są abstrakcyjną reprezentacją komórek pamięci fizycznej. W podobny, symboliczny (tj. tekstowy) sposób, reprezentowane są adresy procedur. Wiązanie adresów to proces przekształcenia takiego symbolicznego adresu na pewien adres w przestrzeni adresowej. Podaj trzy przykłady, kiedy ten proces może nastąpić? Odpowiedz na to pytanie osobno dla wiązania adresów zmiennych i adresów procedur. Dlaczego wiązanie adresów podczas wykonania procesu bywa przydatne?

Zadanie 2

Opisz pobieżnie mechanizmy segmentacji oraz stronicowania (z uwzględnieniem translacji adresów i deskryptorów segmentów / stron). Jakie problemy występujące przy ciągłym przydziale pamięci rozwiązują te mechanizmy? Porównaj, w szczególności, segmentację z ciągłym przydziałem obszarów o zmiennej długości (ang. variable partitions). Dlaczego i w tym przypadku segmentacja jest lepsza?

Zadanie 3

Rozważmy zarządzanie pamięcią użytkownika - tj. odpowiedniki funkcji bibliotecznych `malloc` i `free`. Systemy operacyjne pozwalają procesom na przydział dodatkowej pamięci na użytek bibliotecznego alokatora pamięci. Jakie mechanizmy należy udostępnić celem implementacji tych funkcji dla: ciągłego przydziału pamięci, segmentacji i stronicowania. Wyjaśnij czym jest fragmentacja zewnętrzna i wewnętrzna w kontekście wyżej wymienionych metod przydziału.

Zadanie 4

Wyjaśnij trzy strategie zarządzania pamięcią w ciągłym przydziale: bazujące na listach (best-fit, first-fit, next-fit), mapie bitowej zajętych bloków (bitmap) i systemach bliźniaków (buddy systems). Porównaj je ze względu na aplikowalność do różnych rozmiarów przydzielanych bloków i wydajność operacji `malloc` / `free`. Zaprezentuj nietrywialne przykłady działania przydziału i zwalniania bloków.

Zadanie 5

Załóżmy, że proces użytkownika posiada pewną ilość wolnej pamięci na stercie. Pamięć ta może być zarządzana przez alokator typu `malloc` / `free`, będący funkcjonalnością dostarczoną przez bibliotekę standardową. Wykorzystując wymienione w poprzednim zadaniu strategie, zaproponuj hybrydową implementację takiego alokatora. Możesz wykorzystać również dodatkowe struktury danych typu zbalansowane drzewa (np. `splay`, `red-black`, `b-tree`) i tablice mieszające, do tego by efektywnie przechowywać informacje o nieużytkach określonej wielkości.

Zadanie 6

Porównaj mechanizm translacji adresów wykorzystujący hierarchiczną tablicę stron i odwróconą tablicę stron. Wytłumacz jak przebiega proces tłumaczenia adresu z logicznej przestrzeni adresowej do fizycznej. Jakiej wielkości są struktury opisujące przestrzeń adresową pojedyńczego procesu? Wytłumacz znaczenie dodatkowych bitów w deskryptorze strony.

Zadanie 7

Rozważ stronicowanie z hierarchiczną tablicą stron. Czemu służy pamięć podręczna TLB (ang. Translation Lookaside Buffer)? Jakie korzyści wynikają z jej użycia. Biorąc pod uwagę pamięć podręczną (indeksowaną wirtualnie) i TLB, jakie problemy zauważasz przy zmianie przestrzeni adresowej przy przełączaniu procesów? Czemu służą tzw. duże strony (ang. huge pages)?

Zadanie 8

Rozważmy system ze stronicowaniem. Ogólnie proces nie może korzystać z pamięci, której nie jest właścicielem. Jednak czasami chcielibyśmy komunikować się z innymi procesami poprzez pamięć dzieloną. Jak zrealizować ją w systemie ze stronicowaniem? Pamiętaj, że dzielony obszar może być widoczny pod różnymi adresami logicznymi. Jakie problemy tu dostrzegasz?

Zadanie 9

Zaprezentuj hybrydowy mechanizm translacji adresów (segmentowanie i stronicowanie). Opisz jak zmieniły się algorytm translacji i struktury danych opisujące przestrzeń adresową. Podaj zalety i wady tego schematu w stosunku do czystego segmentowania i stronicowania. Czy rozwiązuje on problem synonimów w pamięci podręcznej indeksowanej wirtualnie?

Zadanie 10

Scharakteryzuj systemy operacyjne należące do klasy SASOS (ang. single address space operating system). Podaj przykłady takich systemów i ich zastosowania. Jakie są ich przewagi nad systemami z lokalną przestrzenią adresową per proces? Pokaż w jaki sposób rozwiązać ochronę dostępu i dzielenie danych między procesami, korzystając z hybrydowego mechanizmu translacji adresów.