Systemy operacyjne

Ćwiczenia 5

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Stallings: 7.1 - 7.4, 7A, 8.1
- Tanenbaum: 3.1 - 3.3
- Silberschatz: 8

Zadanie 1

Przypomnij mechanizmy używane do implementacji pamięci wirtualnej – **segmentację** oraz **stronicowanie**. Jakie problemy występujące przy **ciągłym przydziale pamięci fizycznej** rozwiązują te mechanizmy? Porównaj segmentację z **ciągłym przydziałem obszarów o zmiennej długości**. Dlaczego i w tym przypadku segmentacja jest lepsza? Wyjaśnij czym jest **fragmentacja zewnętrzna** i **wewnętrzna**. Do czego służy **kompaktowanie**?

Zadanie 2

Rozważmy zarządzanie pamięcią w przestrzeni użytkownika – tj. funkcje biblioteczne `malloc` i `free` (to nie są wywołania systemowe!). Jakie szczególne właściwości mają tak przydzielane bloki? Czy przydzielona pamięć może podlegać kompaktowaniu? Systemy operacyjne pozwalają procesom na przydział dodatkowej pamięci na użytek bibliotecznego zarządcy pamięci. Opisz wywołania systemowe **sbrk** oraz **mmap** – czemu z tego drugiego korzysta się częściej? Jednym z częstych błędów związanych z zarządzaniem stertą są **wycieki pamięci**. Jakie inne błędy znasz?

Zadanie 3

Wyjaśnij algorytmy zarządzania wolną pamięcią w ciągłym przydziale: **lista bloków**, **mapa bitowa bloków**. Na czym polega polityka **first-fit**, **next-fit**, **best-fit**, **worst-fit**? Jakie argumenty przemawiają za użyciem pierwszego, a jakie za drugiego z nich. W jakim celu wykorzystuje się technikę **złączania nieużytków**? Dla obydwu algorytmów opisz:

- jak wygląda struktura danych przechowująca listę wolnych / zajętych bloków?
- jak przebiegają operacje `alloc` i `free`?
- jaka jest oczekiwana złożoność czasowa?
- jaki jest narzut (ang. *overhead*) pamięciowy struktur danych?

Zadanie 4

Wewnątrz jądra systemu operacyjnego istnieją dwa dodatkowe typy algorytmów zarządzania pamięcią – **system bliźniaków** (ang. *buddy systems*) i **algorytm płytowy** (ang. *slab allocator*). Wyjaśnij do czego są stosowane i odpowiedz na pytania wypunktowane w poprzednim zadaniu.

Zadanie 5

Bazując na metodzie **quick-fit** zaproponuj hybrydową implementację zarządcy pamięci. Możesz wykorzystać dodatkowe struktury danych (np. `splay`, `red-black`, `b-tree`) oraz **gorliwe** lub **leniwe złączanie** nieużytków. Jak będzie działać `malloc` i `free`? Jaka będzie ich oczekiwana złożoność obliczeniowa? Jak będą wyglądały struktury danych przechowujące wolne bloki?

Zadanie 6 [bonus]

Efektywny przydział pamięci jest kłopotliwy w aplikacjach wielowątkowych, zatem i w jądrze systemu operacyjnego. Algorytmy przydziału cierpią ze względu na często występujące zjawisko **rywalizacji o blokady** (ang. *lock contention*) – dlaczego? Opisz pobieżnie strukturę zarządcy pamięci thread-caching malloc i wyjaśnij jakich technik używa, aby efektywnie obsługiwać żądania o przydział bloków pamięci od wielu wątków.

Zadania do wykonania w domu

Rozwiązania poniższych zadań należy zapisać na kartce formatu A4, podpisać imieniem i nazwiskiem, oraz zdać na początku zajęć prowadzącemu.

Zadanie 7

Biblioteczny algorytm przydziały pamięci dysponuje `N` bajtowym obszarem pamięci. Dostarcza funkcji o sygnaturach `malloc: size -> #id`[1] i `free: #id -> void`. Implementacja bazuje na dwukierunkowej liście wolnych bloków posortowanych względem adresu i zadanej polityce. Algorytm gorliwie złącza bloki. Pomijając narzut związany z utrzymywaniem nagłówka bloków, jak będzie wyglądać zarządzany blok pamięci po wykonaniu żądań dla polityki: `best-fit`, rozmiaru obszaru: `50`, ciągu żądań: `5 14 20 4 #2 #1 #3 7`.

Zadanie 8

Wykonaj polecenia z poprzedniego zadania dla polityki: `first-fit`, rozmiar obszaru: `40`, ciągu żądań: `5 4 8 13 #2 8 #4 #1 3 #3`.

[1]Unikalny identyfikator bloku (w praktyce adres pierwszej komórki bloku). Na potrzeby zadania identyfikatory to kolejne liczby naturalne nadawane blokom według kolejności przydziału w czasie.