Systemy operacyjne 2016

Lista zadań nr 7

Na zajęcia 24 listopada 2016

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Stallings (wydanie ósme): 7.1 – 7.4
- Tanenbaum (wydanie czwarte): 3.1 – 3.2

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytłuszczoną czcionką**. Dodatkowo należy zapoznać się z podręcznikami systemowymi oznaczonymi `czcionką maszynową`.

Zadanie 1. Zaprezentuj przykład występowania zjawiska **fragmentacji wewnętrznej** i **zewnętrznej**. Do czego służy **kompaktowanie** i kiedy można je stosować? Jakie problemy związane z **ciągłym przydziałem** pamięci fizycznej rozwiązuje stronicowanie?

Zadanie 2. Jakie właściwości mają **bloki pamięci** przydzielane i zwalniane z użyciem procedur[1] `malloc(3)` i `free(3)`? Czy bloki te mogą podlegać kompaktowaniu? Kiedy powstaje błąd zwany **wyciekiem pamięci** (ang. *memory leak*)? Jak implementacja bibliotecznego menadżera pamięci może wspomagać wykrywanie błędów **podwójnego zwolnienia** (ang. *double free*) i **przepełnienia bufora** (ang. *buffer overrun*)?

Zadanie 3. Systemy operacyjne udostępniają wywołania systemowe do przydziału stron na użytek bibliotecznego algorytmu zarządzania pamięcią. Opisz jak wykorzystać w tym celu wywołania `sbrk(2)` oraz parę `mmap(2)` i `munmap(2)`. Czemu implementacje `malloc` preferują drugą opcję?

Zadanie 4. Opisz listowy algorytm zarządzania pamięcią w przydziale ciągłym z polityką **first-fit**. W jakim celu wykorzystuje się technikę **złączania** wolnych bloków? Odpowiedz na pytania:

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

UWAGA! Algorytm zarządzania pamięcią może korzystać tylko i wyłącznie z uprzednio przydzielonego jednego ciągłego obszaru pamięci – również do przechowywania dynamicznych struktur danych. Jakiekolwiek mechanizmy przydziału pamięci poza tym obszarem są niedozwolone!

Zadanie 5. Wyjaśnij różnice między politykami **first-fit**, **next-fit**, **best-fit**, **worst-fit** stosowanymi w listowym algorytmie przydziału pamięci. W jakich warunkach poszczególne polityki mogą zapobiegać fragmentacji? Skorzystaj z obserwacji, że duże bloki mają dłuższy **czas życia**, niż małe bloki. Zauważ, że listę wolnych bloków możemy również przeszukiwać od końca.

[1] `malloc` i `free` są procedurami biblioteki standardowej języka `C`, a nie wywołaniami systemowymi!

Zadanie 6. Rozważmy przydział i zwalnianie bloków o ustalonej długości – np. węzłów drzewa binarnego. Opisz algorytm przydziału pamięci korzystający z **bitmapy** i odpowiedz na pytania z zadania 4. Jak przyspieszyć wyszukiwanie wolnych bloków z użyciem funkcji `ffs(3)` i dwupoziomowego opisu bitmapy? Należy zminimalizować ilość potencjalnych chybień w pamięć podręczną.

Zadanie 7. Jądro systemu operacyjnego implementuje wyspecjalizowany algorytm do zarządzania stronami pamięci fizycznej. Wyjaśnij działanie **systemu bliźniaków** (ang. *buddy systems*) i odpowiedz na pytania z zadania 4. Rozważ wyłącznie zarządzanie blokami 2^i (dla dowolnego i) stron pamięci. Nie można przechowywać danych wewnątrz stron, dlatego będziesz potrzebować dwóch statycznie przydzielonych obszarów – tablicy list bloków o danej długości i tablicy wszystkich stron.

Zadanie 8. Dlaczego algorytmy przydziału pamięci często cierpią na zjawisko **rywalizacji o blokady** (ang. *lock contention*)? Opisz pobieżnie struktury danych wykorzystywane przez bibliotekę `thread-caching malloc`[2]. Wyjaśnij techniki używane do efektywnej obsługi żądań przydziału bloków w aplikacjach wielowątkowych.

[2] `http://goog-perftools.sourceforge.net/doc/tcmalloc.html`