

Zarządzanie pamięcią

Jakub Robaczewski

Funkcje:

Do wykonania tego zadania należało zaimplementować 2 funkcje:

```
int hole_map( void *buffer, size_t nbytes )
int worst_fit( int w )
```

Zostały one zaimplementowane w `usr/src/mm/alloc.c`, a odwołania do nich zostały dodane do `usr/src/mm/table.c` i `usr/src/mm/proto.h`. Dodatkowo należało dodać odwołania puste do `usr/src/fs/table.c` oraz zwiększyć ilość odwołań i dodać ich nazwy do `/usr/include/minix/callnr.h`

Algorytm worst-fit:

Dodatkowo należało zaimplementować algorytm worst-fit jako alternatywną metodę przydzielania pamięci, w tym celu stworzyłem zmienną globalną worstFit, która jest zmieniana przez funkcję worst_fit() i pozwala na wybranie aktywnego algorytmu.

Algorytm iteruje po tablicy pamięci szukając największego bloku, a gdy go znajdzie, „odrywa” z niego małą część dla nowego procesu.

Testowanie:

Testowanie programu odbywa się przez uruchomienie skryptu test.sh, który wywołuje programy: x, t oraz w. Ostateczny rezultat przedstawiony jest w postaci tabeli alokowanej pamięci:

[std]	[worst]
[5] 22 12 38 62 257784	[6] 22 12 38 62 62 257722
[5] 22 12 38 62 257722	[7] 22 12 38 62 62 62 257529
[5] 22 12 38 62 257524	[8] 22 12 38 62 62 62 62 257338
[5] 22 12 38 62 257395	[9] 22 12 38 62 62 62 62 257147
[5] 22 12 38 62 257266	[10] 22 12 38 62 62 62 62 256956
[5] 22 12 38 62 257137	[11] 22 12 38 62 62 62 62 256765
[5] 22 12 38 62 257008	[12] 22 12 38 62 62 62 62 256574
[5] 22 12 38 62 256879	[13] 22 12 38 62 62 62 62 256383
[5] 22 12 38 62 256750	[14] 22 12 38 62 62 62 62 256192
[5] 22 12 38 62 256621	[15] 22 12 38 62 62 62 62 256001
[5] 22 12 38 62 256492	[16] 22 12 38 62 62 62 62 255810
[5] 22 12 38 62 256492	[16] 22 12 38 62 62 62 62 255810
[6] 22 12 38 62 129 256492	[16] 22 12 38 62 62 191 62 62 255810
[6] 22 12 38 62 252 256627	[14] 22 12 38 62 62 573 62 62 255810
[6] 22 12 38 62 381 256627	[13] 22 12 38 62 62 764 62 62 255810
[6] 22 12 38 62 510 256627	[12] 22 12 38 62 62 955 62 62 255810
[6] 22 12 38 62 639 256627	[11] 22 12 38 62 62 1146 62 62 255810
[6] 22 12 38 62 768 256627	[10] 22 12 38 62 62 1337 62 62 255810
[6] 22 12 38 62 897 256627	[9] 22 12 38 62 62 1528 62 62 255810
[6] 22 12 38 62 1026 256627	[8] 22 12 38 62 62 1719 62 255810
[5] 22 12 38 62 257784	[6] 22 12 38 62 62 257722

Testy możemy podzielić na 2 etapy: alokację i dealokację. W pierwszym etapie tworzone jest 10 procesów, które są usypiane na 10 sekund (etap A). Następnie pamięć jest uwalniana i konsolidowana. Jak możemy zauważyć, algorytm worst-fit „odrywa” małe fragmenty z największego bloku i nowe mniejsze natomiast algorytm domyślny próbuje znaleźć najlepsze dopasowanie.

Możemy też zauważyć, że stan początkowy i końcowy są identyczne co świadczy o braku „wyciekania pamięci” podczas alokacji i dealokacji.