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

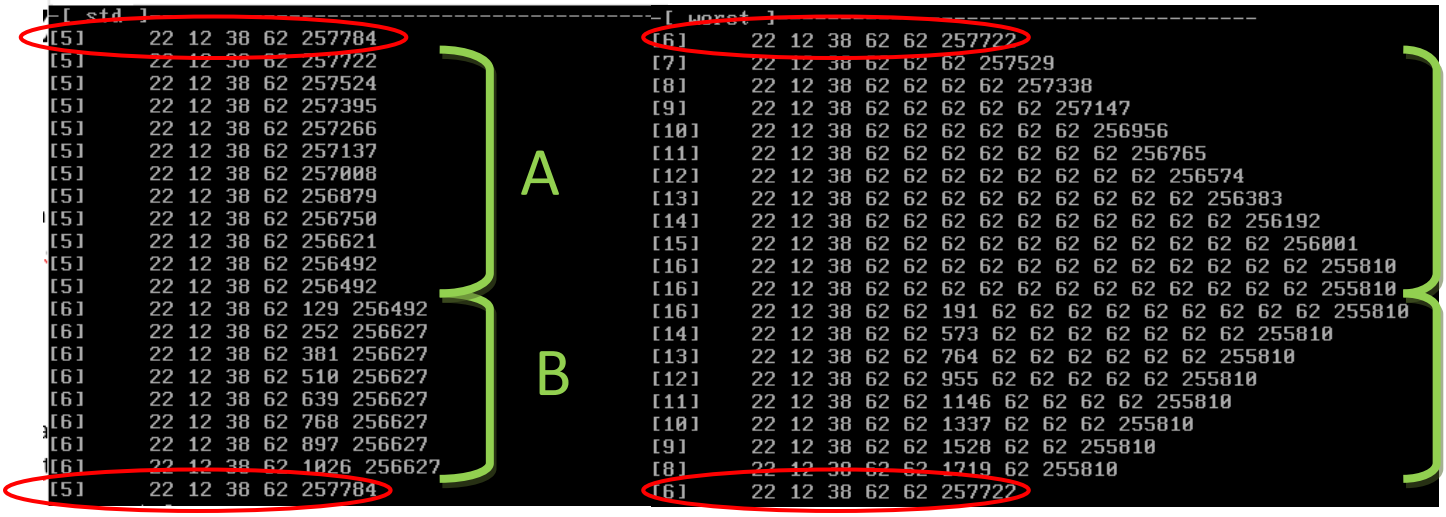
Algorytm worst-fit:

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:



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.