

Операционные Системы

Алгоритмы аллокации памяти

April 20, 2017

Аллокация памяти

- ▶ Рассмотрим простейшую постановку задачи:
 - ▶ есть непрерывный участок логической памяти;
 - ▶ функция аллокации: `void *alloc(int size);`
 - ▶ функция освобождения: `void free(void *free).`

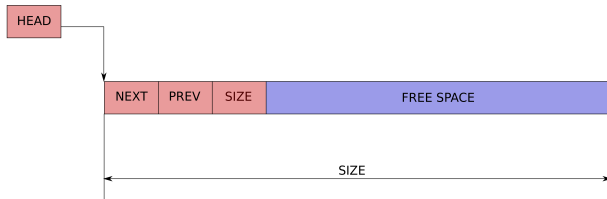
Выравнивание

- ▶ Некоторые процессоры требуют выравненных указателей
 - ▶ 2 байта - выравнивание 2 байта;
 - ▶ 4 байта - выравнивание 4 байта;
 - ▶ 8 байт - выравнивание 8 байт...

Простой подход к аллокации

- ▶ Создадим связный список свободных блоков
 - ▶ каждый узел списка описывает непрерывный свободный участок;
 - ▶ узлы списка хранятся в начале каждого свободного блока.

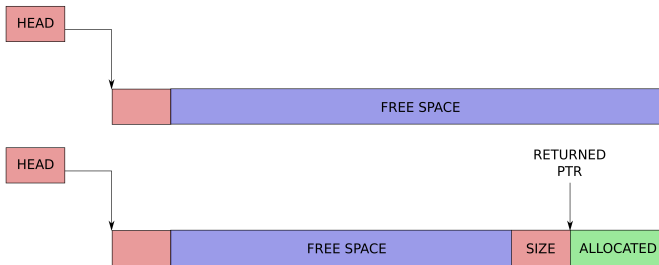
Связный список свободных блоков



Аллокация свободного блока

- ▶ Пройдемся по списку и найдем блок достаточного размера
 - ▶ если блок слишком большой, то отрезаем от него часть;
 - ▶ если подходящего блока не нашлось, то возвращаем ошибку.

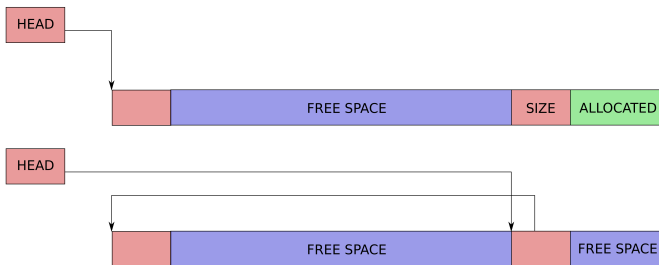
Аллокация свободного блока



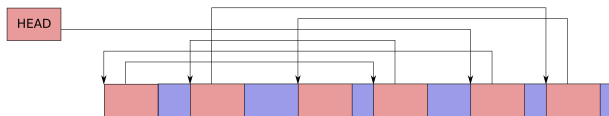
Освобождение занятого блока

- ▶ Чтобы освободить свободный блок, его нужно вернуть в список
 - ▶ например, можно добавить в список новый элемент.

Освобождение занятого блока



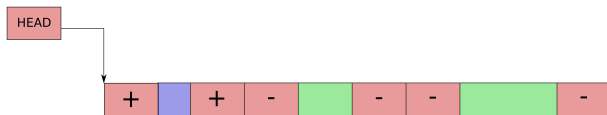
Фрагментация свободной памяти



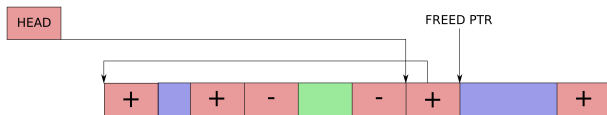
Боремся с фрагментацией

- ▶ Как избежать подобной фрагментации?
 - ▶ искать смежные блоки проходом по списку ($O(N)$);
 - ▶ поддерживать список упорядоченным ($O(N)$);
 - ▶ использовать упорядоченную структуру вместо списка ($O(\log N)$);
 - ▶ использовать граничные маркеры (Border Tags, $O(1)$).

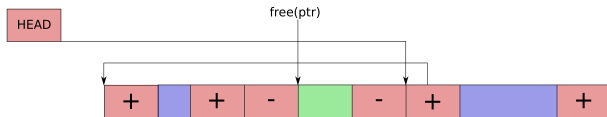
Граничные маркеры



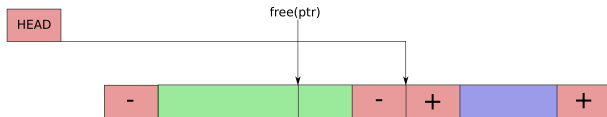
Граничные маркеры



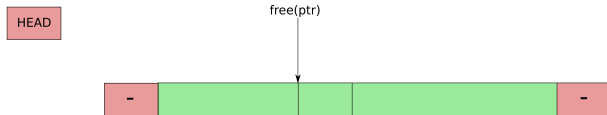
Граничные маркеры



Граничные маркеры



Граничные маркеры



Граничные маркеры



Buddy аллокатор

- ▶ Buddy аллокатор предназначен для аллокации больших участков памяти
 - ▶ buddy аллокатор аллоцирует память блоками;
 - ▶ блок - 2^i последовательных страниц;
 - ▶ например, 1 страница, 2 страницы, 4 страницы и т. д.;
 - ▶ но не 3 страницы или 5 страниц.

Дескрипторы страниц

- ▶ Buddy аллокатор не будет работать с памятью напрямую
 - ▶ вместо страниц памяти будем использовать в алгоритме дескрипторы;
 - ▶ просто массив дескрипторов - по адресу страницы легко получить дескриптор и наоборот.

Дескрипторы страниц

- ▶ Что будет храниться в дескрипторах?
 - ▶ указатели, чтобы связать дескрипторы в двусвязный список;
 - ▶ признак свободности/занятости;
 - ▶ уровень (указание на размер блока).

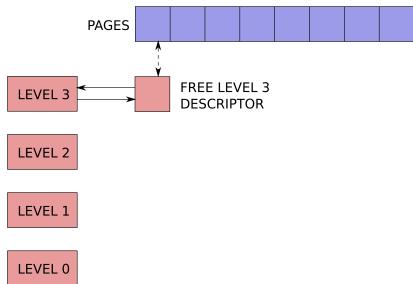
Списки свободных блоков

- ▶ Пусть у нас изначально есть 2^N последовательных свободных страниц:
 - ▶ заведем $N + 1$ изначально пустой двусвязный список;
 - ▶ i -ый список будет хранить свободные блоки размером 2^i страниц.

Начальное состояние

- ▶ Возьмем дескриптор 0-ой страницы
 - ▶ отметим дескриптор как свободный;
 - ▶ зададим в дескрипторе уровень $N - 1$;
 - ▶ добавим дескриптор в список $N - 1$.

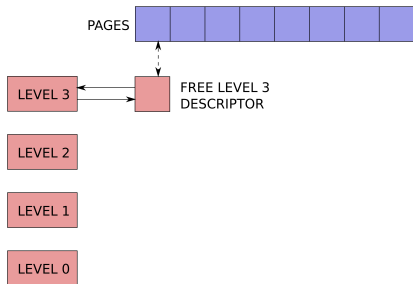
Инициализация



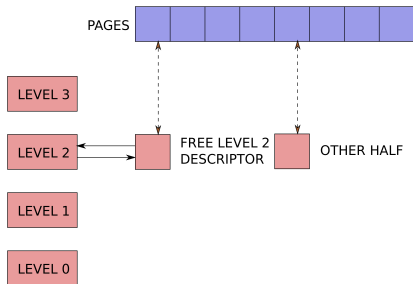
Аллокация

- ▶ Мы хотим аллоцировать 2^i страниц
 - ▶ найдем непустой список с блоками $\geq 2^i$;
 - ▶ берем один из блоков и делим его пополам, пока не останется 2^i .

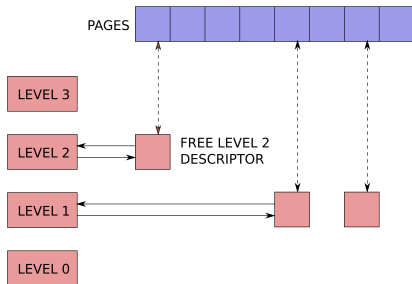
Аллокация



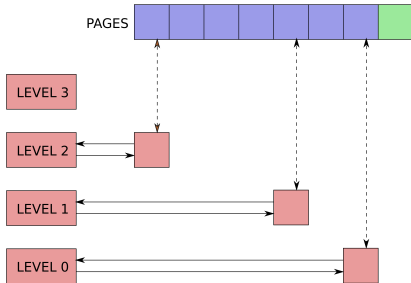
Аллокация



Аллокация



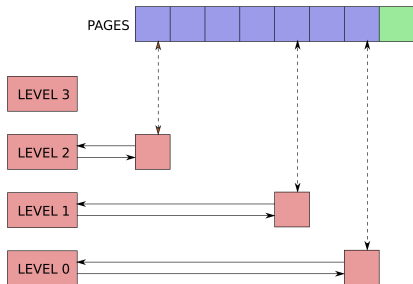
Аллокация



Освобождение

- ▶ Теперь мы хотим освободить блок размера 2^i
 - ▶ мы могли бы просто добавить дескриптор в список i , но это приведет к фрагментации;
 - ▶ мы должны попытаться объединить смежные блоки.

Buddies



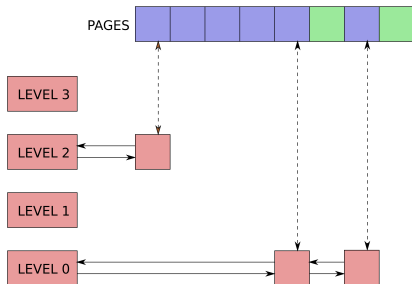
Buddies

- ▶ Как найти парный блок?
 - ▶ если мы освобождаем блок размера 2^i с номером j ;
 - ▶ парный блок имеет номер $j \oplus 2^i$;
 - ▶ \oplus - исключающее побитовое ИЛИ.

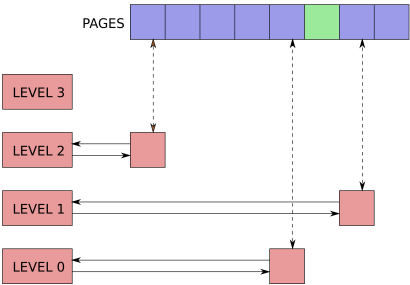
Освобождение

- ▶ Когда можно объединять парные блоки?
 - ▶ если оба блока свободны;
 - ▶ если оба блока имеют один размер (уровень в дескрипторе).

Освобождение



Buddies



Кеширующий аллокатор

- ▶ Создадим кеш блоков фиксированного размера
 - ▶ кеш будет аллоцировать/освобождать большие регионы памяти, используя другой аллокатор;
 - ▶ кеш "нарезает" большие регионы на блоки фиксированного размера.

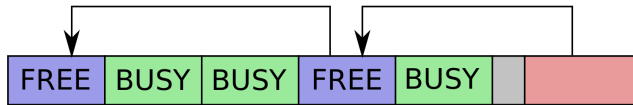
Кеширующий аллокатор

- ▶ Кеширующий аллокатор имеет ряд достоинств:
 - ▶ фиксированный размер блоков позволяет бороться с фрагментацией;
 - ▶ аллокация/освобождение могут работать за $O(1)$;
 - ▶ можно скомбинировать кеши разных размеров и построить универсальный аллокатор.

SLAB

- ▶ Кеширующий аллокатор, предложенный Джеффом Бонвиком и использованный в SunOS (Solaris):
 - ▶ slab - описывает большой регион памяти, который разбивается на маленькие блоки фиксированного размера;
 - ▶ все свободные блоки связываются в список;
 - ▶ количество элементов списка и указатель на первый элемент сохраняются в заголовке.

SLAB



Управление SLAB-ами

- ▶ SLAB аллокатор управляет slab-ами:
 - ▶ если нет slab-а со свободными объектами - аллоцируем новый;
 - ▶ если все объекты в slab-е свободны - можно освободить slab.

Управление SLAB-ами

- ▶ SLAB аллокатор поддерживает три списка slab-ов:
 - ▶ полностью свободные slab-ы;
 - ▶ частично занятые slab-ы;
 - ▶ полностью занятые slab-ы.

Освобождение

- ▶ Чтобы освободить элемент, нужно найти slab, которому он принадлежит
 - ▶ мы можем сохранить указатель на slab рядом с аллоцированной памятью;
 - ▶ мы можем потребовать, чтобы размер и выравнивание slab-а были равны 2^i .

Освобождение

