OPERATING SYSTEMS-2 (Course Project)

REPORT
Linux Slab Allocator in C++

V Sri Charan Reddy(CS16BTECH11044)

A Santhosh Reddy(CS16BTECH11003)

The assignment and release of objects(memory) are among the most common operations in the kernel. A fast kernel memory allocator is therefore essential. However, in many cases, the cost of initialisation or destroying the object exceeds the allocation cost and freeing memory for this.

The slab is the primary unit of currency in the slab allocator. When the allocator needs to grow a cache, for example, it acquires an entire chunk of objects at once. Similarly, the allocator reclaims unused memory (shrinks a piece) by relinquishing a complete slab.

The slab allocator consists of a variable number of caches that are linked together on a doubly linked circular list called a *cache chain*

To help eliminate internal fragmentation normally, many sets of buckets of small memory buffers ranging from 2^2 (4) bytes to 2^{13} (8192) bytes are maintained

Struct slab:

Each slab contains contents such as total objects that can be stored within the slab, free objects which is the total number of objects that can still be allocated.

Also a bitmap which represents the occupancy position within a slab.

In each slab there is nextslab pointer which points to the next slab.

Also there is a pointer to the bucket to which it belongs to.

Struct bucket:

Each bucket stores the size of the objects which can be stored in its bucket and also a pointer to the first slab to which it corresponds to.

Initially if the firstslab points to NULL, we create a new slab of 64kB memory and join it to the bucket. Now each time the slab gets filled with objects create another slab and point the previous slab's next pointer to the newly created slab's pointer.

Struct object:

Users data is stored here within the struct object. It also stores a slab pointer which denotes the slab to which it belongs to. This will be useful in order to find to which bucket it belongs to.

Mymalloc:

This was the function which was used inorder to allocate the memory for each new slab. Every time we need a slab a memory of 64kB is allocated using the mmap system call. It returns a void pointer to the memory location to which an object is sent into.

Myfree:

This function takes a void pointer as input and deletes the object which is being pointed out by that pointer. If the free number of objects in slab in which the pointer is present is one then we delete the slab and we point the previous slabs next pointer to the current slab's nextslab pointer.

If the number of free objects in slab is greater than 1 then we simply delete an object, i.e we set the bitmap of that particular element to be as zero.