CS3523

OPERATING SYSTEMS

Programming Assignment 5

Introduction

The source code files attached provide a library to implement a slab allocator in user space in c++.

The source files are libmymem.cpp and libmymem.hpp. memutil.cpp is used to test the libraries.

The library provides 2 API's - void *mymalloc (unsigned int size) and void myfree(void *ptr)

How it works -

In the library, we declare a global array of struct buckets. Each bucket is used to keep track of a object size. We keep certain number of slabs per bucket. When we get a request, we go to the corresponding bucket, search for a slab with a free object and return the address of that object to the user.

Every time we allocate, we add a pointer back to the slab header just before the object. This pointer is used to free the object. When a request to free the object comes in, we take the address of the object, decrement it to find the location of the slab pointer and make the deletions there.

While deleting, if all the objects in the slab are free, we delete the slab.

The value of the return address is found by pointer arithmetic.

The bitmap

To keep track of which objects are free in a slab, we use a bitmap. It is an array of ints, where each int represents a binary number. If, in the binary representation, the digit is 0, then there is a free object at that index, else its an allocated object.

For each slab, we traverse the bitmap till we find a 0. If we don't, then we move on to the next slab or allocate a new slab for the current object.

Thread Safety

We ensure thread safety by maintaining a mutex lock per bucket. SImultaneous access to buckets of different size is permitted, but simultaneous access to the same bucket is not allowed. By maintaining a separate mutex per bucket instead of one mutex for the entire thing, we improve performance.

When a thread is interrupted before its finishes execution, since it still holds the lock, the values in the bucket will be accessible by other threads only after this thread finishes execution. In this way the data is protected, hence the library is thread safe and re entrant.

Slab allocation

The memory for the slab is allocated using a private, anonymous memory map. The slab size in 64KB. The mmap call maps a memory chunk of size 64KB to the user space. The pointer that mmap returns is the pointer to the beginning of the slab. We type cast it and store the slab header and other metadata.

The instructions to use the library are given in the readme file attached.