Report

Sz232

1. Implementation

1.1 Data structure

The basic structure is a doubly linked list that connects all freed datagrams. Every time when a block is freed, it's added to the free list. The link node includes the size of assigned block, the pointer to previous freed block and the pointer to the next freed block.

```
typedef  struct  block
    {size_t size;
    struct block *next;
    struct block *prev;
} block;
```

1.2 Implementation of Best Fit

Because it's using best-fit policy, the program will start from the head node and go through the link list to find the block that matches best. When the block is large enough, it will split into two blocks.

When merging free blocks. It will go through both directions until there's no blocks available. Some optimizations are introduced, such as the program will stop immediately when it finds the block that exactly matches the require.

2. Result and analysis

2.1 Result

The Lock version is slightly slower than the no lock version.

	Execution Time	Data Segment Size
No Lock	0.10	48089
Lock	0.11	46082

2.2 Analysis

The execution time and data segment size are averaged over tests.

The reason why no lock version is slightly faster than the lock version is that while the system is calling sbrk(), other tasks like inserting, merging and removing blocks can still go parallelly.

And the no lock version also takes more space. This is because each thread occupies more data in thread's local memory.

3. Potential improvements

3.1 Improvement of performance

Using different locks may improve the performance in lock version. Given a long array, it’s pretty expensive to lock the whole array for one single operation. The read and write of array could be separated, so that inserting blocks won’t interfere with the thread that’s looking for the best fit block.