Chris Kasper
CS 370
Programming Assignment 3

<u>Design</u>

For memory management, Inferno sorts its memory into three different pools: the main pool, image pool, and the heap pool. When there is an allocation request from one of the pools, it removes that memory from the pool, and returns the block of data that has been allocated. When a block is going to get deallocated, it gets organized into a binary tree that is known as a free list. This free list is utilized to provide Inferno with a best fit memory management system when the free list is not empty.

In this assignment, I implemented slab allocation for requests ≤ 4K bytes, and maintained lists of free blocks in a pool of size 32, 64, 128, 256, 512, 1024, 2048, and 4096 bytes. This requires changes in *alloc.c*. One of the first changes is to check the size of the requests for allocation, and if they are ≤ 4K bytes, round them up to nearest power of 2. Anything bigger will be handled by the existing code.

When a block is going to be unallocated, the size is checked. If it can go into the slab list of the pool, you put it in the size-specific list. Each list is a linked-list, where newly added blocks are added to the head of the list. What is getting stored in these lists are the pointers to the blocks. The magic number of the block is also changed to "0x31ab", which a special number to denote a slab block. This lead to a small change in *poolaudit()* so the function wouldn't treat this block as "bad magic" or corrupted.

When a request comes in for a size ≤ 4K bytes, we adjust the size as noted above, and check if the size-specific slab list is empty or not. If it is empty, then we allocate normally. However, if it is not empty, then we take out the block at the head of the size-specific list, adjust that list, and return the block.

The main idea behind slab allocation is that we are reusing blocks of fixed size, and avoiding having to traverse through a binary tree to look for a block of memory that it fits best with the request coming in.

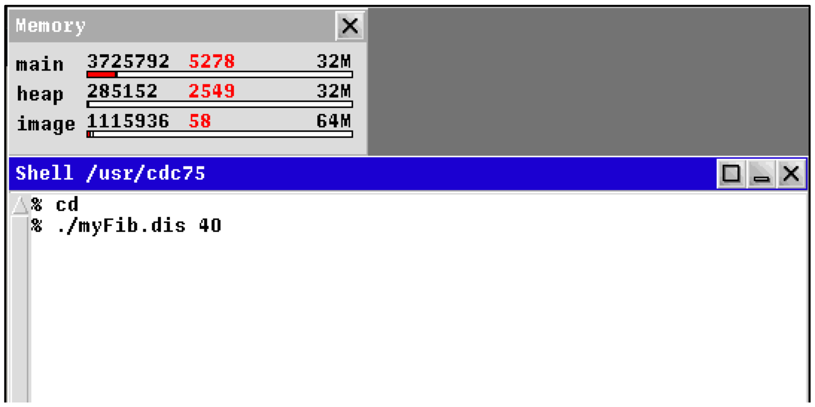
Testing

There were two types of testing I performed in order to see that my slab allocation modifications were working properly. The first one was to make sure that the slab allocation lists in each pool were being maintained properly. To verify this, I created a function called *printSlabList(Pool* p)*, which printed out the lists of the specified pool. I called this function before and after adding to a slab list, and before allocating blocks from the slab list to make sure the lists were being updated and used in the correct fashion. These checkpoints have been commented out in my code.

The second type of testing that was to make sure that the slab allocation actually worked. Besides starting Inferno and making sure it didn't crash, I utilized Inferno's memory monitor program to view how much memory is being allocated in each pool. I then ran one of my Limbo programs that computes the Fibonacci Sequence and see how the memory monitor is adjusted.

Test Inferno without slab allocation:

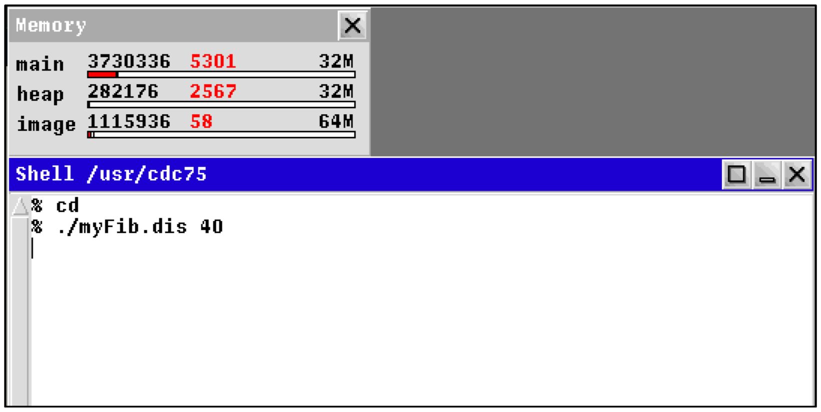
Before running ./myFib.dis 40:



```
main   3725792  5278     32M
heap   285152   2549     32M
image  1115936  58       64M
```

```
% cd
% ./myFib.dis 40
```

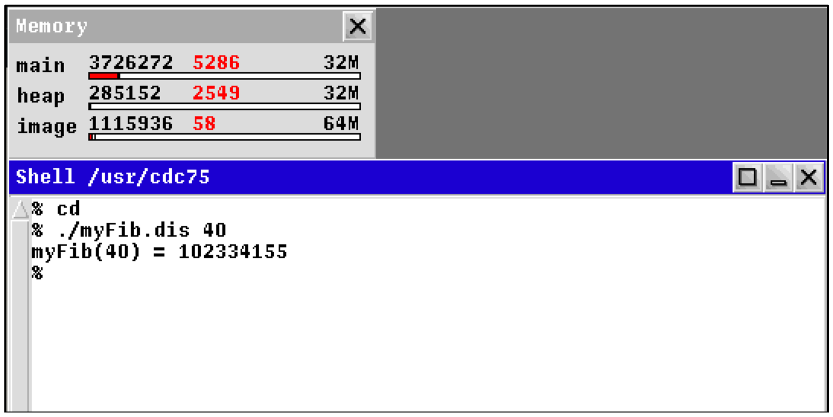
During:



```
main   3730336  5301     32M
heap   282176   2567     32M
image  1115936  58       64M
```

```
% cd
% ./myFib.dis 40
```

After:



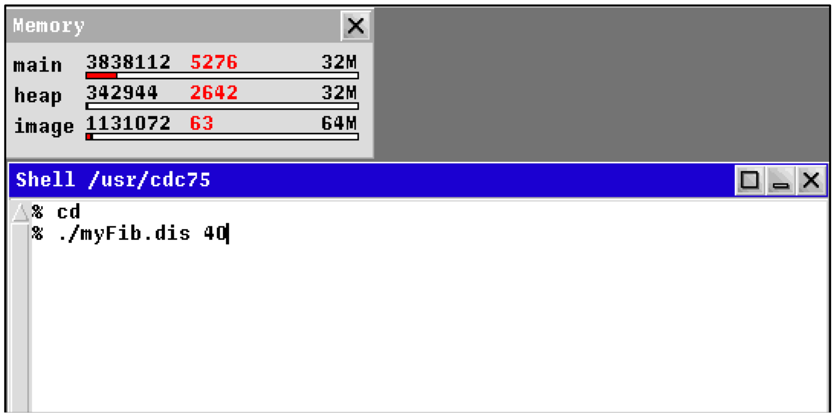
```
main   3726272  5286        32M
heap   285152   2549        32M
image  1115936  58          64M
```

```
% cd
% ./myFib.dis 40
myFib(40) = 102334155
%
```

In this test, we don’t see any changes to the image pool memory, but we do notice that memory had be allocated from the heap, and then was deallocated once the program was finished. The main memory pool also had memory allocated from it (around 5,000 bytes), and then when the program finished, deallocated only some of it back.

<u>Test Inferno with Slab Allocation:</u>

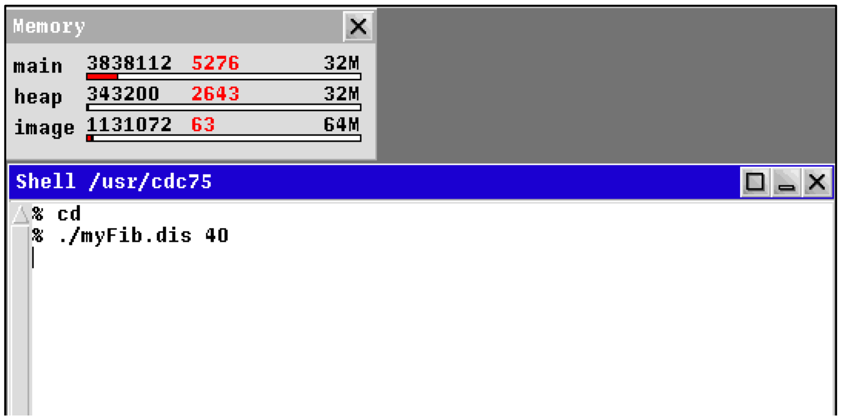
Before running ./myFib.dis 40:



```
main   3838112  5276        32M
heap   342944   2642        32M
image  1131072  63          64M
```

```
% cd
% ./myFib.dis 40
```

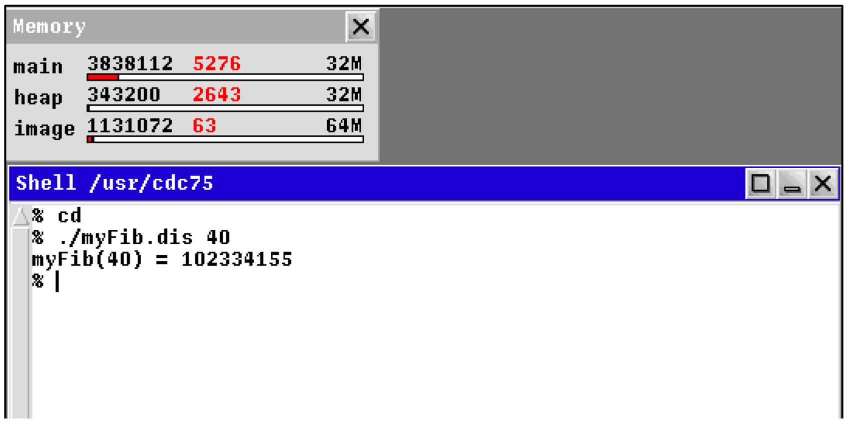
During:



```
main   3838112  5276        32M
heap   343200   2643        32M
image  1131072  63          64M
```

```
% cd
% ./myFib.dis 40
```

After:



```
main   3838112  5276        32M
heap   343200   2643        32M
image  1131072  63          64M
```

```
% cd
% ./myFib.dis 40
myFib(40) = 102334155
% |
```

Based on the memory monitor, we can see that no memory from the main memory had to be allocated to run "*./myFib.dis 40*". We can assume it used some of the slab memory. The same could be said from the image pool. Now, the heap pool did have memory allocated from it. We can see it went from 342,944 bytes to 343,200 bytes. However, doing some simple math tells us it had 256 bytes allocated. This would mean prior to running the program, the heap pool did not have any slab blocks of 256 bytes.