5free-space-mgmt-lab

CST 334 (Operating Systems)
Dr. Glenn Bruns

Lab: Free Space Management

1. Copy these files to your home directory on mlc104: (or get them from the OSTEP page)

   /home/CLASSES/brunsglenn/OSTEP/HW-Freespace/malloc.py

   /home/CLASSES/brunsglenn/OSTEP/HW-Freespace/README-malloc

2. Run the program as follows from your home directory:

   ○ ./malloc.py -s 2 -S 100 -b 1000

The options say that the memory allocator has 100 bytes to allocate, and the address of the memory to be allocated starts at 1000.

The program output shows a sequence of requests to allocate and free memory. Your job is to figure out the state of the free memory list after each request. For example:

- initially the list has a single entry of [addr: 1000, sz: 100], meaning a free block of 100 bytes starting at address 1000. I write this more compactly as (1000, 100).

- after a request for 1 byte, the list is (1001, 99), because 1 byte was taken off the front of the single entry in the free list. Now the remaining free memory is a single block of 99 bytes starting at address 1001. The name of the pointer to the 1 byte of allocated memory is Ptr[0].

- Ptr[0] is now freed, so the list becomes (1000,1) -> (1001,99). By default, malloc.py wants the freed memory list to be ordered, but it is not automatically coalesced.

- Now a request for 7 bytes is made, and the list becomes (1000,1) -> (1008, 92), and a pointer to the allocated memory is in Ptr[1].

- Ptr[1] is now freed, so the list becomes (1000,1) -> (1001,7) -> (1008,92).

- etc.

Run the program again, this time with -c, and you'll see how the list is updated every time memory is allocated or freed.

3. Run the program again, just as in 2, but with -s 3 (don't use -c), giving new random output. Now look at the output and, at each step, show whether memory could be allocated, and the state of the free list. When you're done, run the program again with the -c option to check your work.

4. Do question 1 at the end of OSTEP chapter 17. It is further work with malloc.py. Quoting the text:

First run with the flags -n 10 -H 0 -p BEST -s 0 to generate a few random allocations and frees. Can you predict what alloc()/free() will return? Can you guess the state of the free list after each request? What do you notice about the free list over time?

Run ./malloc.py -help to see what -n and the other options mean.

5. If you still have time, try other problems at the end of chapter 17.
