MP4 Test Plan Casey Hird

Our test plan is divided into several test cases which analyze different border cases or performances of our memory allocation with or without coalescing.

Unit Driver 0:

Example Output:

----- Begin unit driver 0 -----

string length=15

:hello world 15c:

Free list after first allocation

MP4 Heap Memory Statistics

Number of blocks in free list: 2

Min: 4048

Max: 4048

Average: 2032

Total bytes in free list: 4064

Number of sbrk calls: 1

Number of requested pages: 1

Heap status: heap is in-use -- leaks are possible

p=0x2160020, size=253, end=0x2160ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x2160020 <-- dummy

Free list after first free

unit driver 0 has returned all memory to free list

MP4 Heap Memory Statistics

Number of blocks in free list: 3

Min: 16

Max: 4048

Average: 1365

Total bytes in free list: 4096

Number of sbrk calls: 1

Number of requested pages: 1

Heap status: all memory is in the heap -- no leaks are possible

p=0x6040b0, size=0, end=0x6040b0, next=0x2160020 <-- dummy

p=0x2160020, size=253, end=0x2160ff0, next=0x2160000

p=0x2160000, size=1, end=0x2160010, next=0x6040b0

----- End unit driver 0 -----

Test driver 0 is a simple test of whether the allocate and free functions work correctly at all. This function very simply allocates 16 bytes of memory to store a string, prints the stats of the heap, then frees that memory and prints those stats again. We see in the output in the test log that we get the output we would expect from this. Note especially that after all memory has been freed, the total number of bytes in the free list is 4096, the size of one page, which makes sense since we need to request a single page of memory to hold 16 bytes. Also, we see that in the stats when the memory has been allocated and not yet freed, the total number of bytes is 4064, which is equal to 4096 – 32, which makes sense since we needed to allocate space for the 16-byte word plus an additional 16-byte header, requiring us to remove 32 bytes from the free list. Finally, we see that this result is independent of the search pattern used, since the memory being allocated is too small for the method by which we find it to be significant.

Unit Driver 1:

Example Output:

----- Begin unit driver 1 -----

There are 256 units per page, and the size of chunk_t is 16 bytes

first: 496 bytes (31 units) p=0xd60010

p=0xd60200, size=223, end=0xd60ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0xd60200 <-- dummy

second: 2032 bytes (127 units) p=0xd60210

p=0xd60a00, size=95, end=0xd60ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0xd60a00 <-- dummy

third: 1520 bytes (95 units) p=0xd60a10

p=0x6040b0, size=0, end=0x6040b0, next=0x6040b0 <-- dummy

unit driver 1: above Mem_print shows empty free list

fourth: 1008 bytes (63 units) p=0xd61010

p=0xd61400, size=191, end=0xd61ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0xd61400 <-- dummy

first free of 1/8 a page p=0xd60010

p=0x6040b0, size=0, end=0x6040b0, next=0xd61400 <-- dummy

p=0xd61400, size=191, end=0xd61ff0, next=0xd60000

p=0xd60000, size=31, end=0xd601f0, next=0x6040b0

second free of 3/8 a page p=0xd60a10

p=0xd61400, size=191, end=0xd61ff0, next=0xd60000

p=0xd60000, size=31, end=0xd601f0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0xd60a00 <-- dummy

p=0xd60a00, size=95, end=0xd60ff0, next=0xd61400

third free of 1/2 a page p=0xd60210

p=0xd60000, size=31, end=0xd601f0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0xd60a00 <-- dummy

p=0xd60a00, size=95, end=0xd60ff0, next=0xd61400

p=0xd61400, size=191, end=0xd61ff0, next=0xd60200

p=0xd60200, size=127, end=0xd609f0, next=0xd60000

fourth free of 1/4 a page p=0xd61010

unit driver 1 has returned all memory to free list

p=0x6040b0, size=0, end=0x6040b0, next=0xd60a00 <-- dummy

p=0xd60a00, size=95, end=0xd60ff0, next=0xd61400

p=0xd61400, size=191, end=0xd61ff0, next=0xd60200

p=0xd60200, size=127, end=0xd609f0, next=0xd60000

p=0xd60000, size=31, end=0xd601f0, next=0xd61000

p=0xd61000, size=63, end=0xd613f0, next=0x6040b0

MP4 Heap Memory Statistics

Number of blocks in free list: 6

Min: 496

Max: 3056

Average: 1365

Total bytes in free list: 8192

Number of sbrk calls: 2

Number of requested pages: 2

Heap status: all memory is in the heap -- no leaks are possible

----- End unit test driver 1 -----

Test driver 1 uses 4 memory allocations, the first intended to partition the initial free list, and the fourth to test that we can still allocate memory when the current free list is empty. We see that this does work correctly, since our memory allocation function uses the morecore() function to request a new page of memory when necessary. We also see that this changes the value of the total bytes in the free list after all memory has been freed—since we had to request 2 pages of memory this value is now 2*4096 = 8192. Also, we see that the minimum number of bytes in a chunk is 496, which makes sense since the smallest chunk we allocated was 1/8 of the size of a page. We then find that 4096/8 = 512 and 512-16=496 (since one chunk of 16 bytes is allocated for the header). Similarly, the maximum number of bytes in a chunk is allocated for the fourth allocation, which requires a new page. Since this allocation

was for ¼ of the page, we find that 4096/4 = 1024, then 4096-1024-16 = 3056 (minus 16 for the header) so we have 3056 bytes remaining as the size of the largest chunk.

Unit Driver 2:

Example Output:

----- Begin unit driver 2 -----

There are 256 units per page, and the size of chunk_t is 16 bytes

first: 4096 bytes (256 units) p=0x14df010

p=0x6040b0, size=0, end=0x6040b0, next=0x6040b0 <-- dummy

second: 4080 bytes (255 units) p=0x14e0010

p=0x6040b0, size=0, end=0x6040b0, next=0x6040b0 <-- dummy

second: 4112 bytes (257 units) p=0x14e1010

p=0x14e2020, size=253, end=0x14e2ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x14e2020 <-- dummy

first free of entire page p=0x14df010

p=0x6040b0, size=0, end=0x6040b0, next=0x14e2020 <-- dummy

p=0x14e2020, size=253, end=0x14e2ff0, next=0x14df000

p=0x14df000, size=255, end=0x14dfff0, next=0x6040b0

second free of 1 less than a page p=0x14e0010

p=0x14e2020, size=253, end=0x14e2ff0, next=0x14df000

p=0x14df000, size=255, end=0x14dfff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x14e0000 <-- dummy

p=0x14e0000, size=255, end=0x14e0ff0, next=0x14e2020

third free of 1 more than a page p=0x14e1010

p=0x14df000, size=255, end=0x14dfff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x14e0000 <-- dummy

p=0x14e0000, size=255, end=0x14e0ff0, next=0x14e2020

p=0x14e2020, size=253, end=0x14e2ff0, next=0x14e1000

p=0x14e1000, size=257, end=0x14e2010, next=0x14df000

unit driver 2 has returned all memory to free list

p=0x14df000, size=255, end=0x14dfff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x14e0000 <-- dummy

p=0x14e0000, size=255, end=0x14e0ff0, next=0x14e2020

p=0x14e2020, size=253, end=0x14e2ff0, next=0x14e1000

p=0x14e1000, size=257, end=0x14e2010, next=0x14df000

MP4 Heap Memory Statistics

Number of blocks in free list: 5

Min: 4048

Max: 4112

Average: 3276

Total bytes in free list: 16384

Number of sbrk calls: 3

Number of requested pages: 4

Heap status: all memory is in the heap -- no leaks are possible

----- End unit test driver 2 -----

Test driver 2 evaluates the ability of the memory allocation function to satisfy requests for certain border conditions—specifically, we request exactly one page, 1 less than a page of memory, and one more than a page, in that order. We get a result that may be surprising which is that this leaves us with 5 blocks having been created in the free list. This does make sense though, since we have the dummy block, then 1 block for the whole page that is allocated, then when we request 1 less than a page of memory we have one block for the request and one for the remainder of the page, then when we request 1 more than the page size we need to first request 2 pages, then allocate one block to fulfill the request and one block for the remainder of the second page we requested. From this we see that we have the correct result of 5 blocks in the free list and 4 pages being requested from the morecore() function. We also see that the minimum block size is the size of a page minus 3 blocks (4096 – 3*16 = 4048) subtracting one block size each for the header of the chunk, the header of the remainder, and the one chunk less than the page size. Similarly, we see that the maximum block size is the size of a page

plus one block (4096+16 = 4112), as supported by the fact that the largest request is for 1 chunk greater than the page size.

Unit Driver 3:

Example Output:

----- Begin unit driver 3 -----

There are 256 units per page, and the size of chunk_t is 16 bytes

first: 4416 bytes (276 units) p=0x24f0010

p=0x24f1150, size=234, end=0x24f1ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x24f1150 <-- dummy

second: 8192 bytes (512 units) p=0x24f2010

p=0x6040b0, size=0, end=0x6040b0, next=0x24f1150 <-- dummy

p=0x24f1150, size=234, end=0x24f1ff0, next=0x6040b0

second: 20496 bytes (1281 units) p=0x24f4010

p=0x24f9020, size=253, end=0x24f9ff0, next=0x24f1150

p=0x24f1150, size=234, end=0x24f1ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x24f9020 <-- dummy

first free of over a page p=0x24f0010

p=0x24f1150, size=234, end=0x24f1ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x24f9020 <-- dummy

p=0x24f9020, size=253, end=0x24f9ff0, next=0x24f0000

p=0x24f0000, size=276, end=0x24f1140, next=0x24f1150

second free of exactly 2 pages p=0x24f2010

p=0x6040b0, size=0, end=0x6040b0, next=0x24f9020 <-- dummy

p=0x24f9020, size=253, end=0x24f9ff0, next=0x24f0000

p=0x24f0000, size=276, end=0x24f1140, next=0x24f1150

p=0x24f1150, size=234, end=0x24f1ff0, next=0x24f2000

p=0x24f2000, size=511, end=0x24f3ff0, next=0x6040b0

third free of more than 5 pages p=0x24f4010

p=0x24f9020, size=253, end=0x24f9ff0, next=0x24f0000

p=0x24f0000, size=276, end=0x24f1140, next=0x24f1150

p=0x24f1150, size=234, end=0x24f1ff0, next=0x24f2000

p=0x24f2000, size=511, end=0x24f3ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x24f4000 <-- dummy

p=0x24f4000, size=1281, end=0x24f9010, next=0x24f9020

unit driver 3 has returned all memory to free list

p=0x24f9020, size=253, end=0x24f9ff0, next=0x24f0000

p=0x24f0000, size=276, end=0x24f1140, next=0x24f1150

p=0x24f1150, size=234, end=0x24f1ff0, next=0x24f2000

p=0x24f2000, size=511, end=0x24f3ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x24f4000 <-- dummy

p=0x24f4000, size=1281, end=0x24f9010, next=0x24f9020

MP4 Heap Memory Statistics

Number of blocks in free list: 6

Min: 3744

Max: 20496

Average: 6826

Total bytes in free list: 40960

Number of sbrk calls: 3

Number of requested pages: 10

Heap status: all memory is in the heap -- no leaks are possible

----- End unit test driver 3 -----

Test driver 3 analyzes the response of our memory allocation function to requests of sizes that require several pages of memory to fill. We have three allocations in this, the first over a page, the second exactly two pages, and the third over 5 pages. We see that when we are requesting chunks of memory this large, our allocation must request 10 total pages of memory, regardless of the search being used, and that these requests will result in 10*4096 = 40960 bytes being in the free list after all memory has been freed.

Unit Driver 4:

Example Output:

----- Begin unit driver 4 -----

There are 256 units per page, and the size of chunk_t is 16 bytes

first: 1024 bytes (64 units) p=0x1131010

p=0x1131410, size=190, end=0x1131ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1131410 <-- dummy

second: 512 bytes (32 units) p=0x1131420

p=0x1131620, size=157, end=0x1131ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1131620 <-- dummy

first free of 1/4 of a page p=0x1131010

p=0x6040b0, size=0, end=0x6040b0, next=0x1131620 <-- dummy

p=0x1131620, size=157, end=0x1131ff0, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

second: 512 bytes (32 units) p=0x1131630

p=0x1131830, size=124, end=0x1131ff0, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1131830 <-- dummy

first: 2048 bytes (128 units) p=0x1132010

p=0x1132810, size=126, end=0x1132ff0, next=0x1131830

p=0x1131830, size=124, end=0x1131ff0, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1132810 <-- dummy

third free of 1/8 of a page p=0x1131630

p=0x1131830, size=124, end=0x1131ff0, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1132810 <-- dummy

p=0x1132810, size=126, end=0x1132ff0, next=0x1131620

p=0x1131620, size=32, end=0x1131820, next=0x1131830

second free of 1/8 of a page p=0x1131420

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1132810 <-- dummy

p=0x1132810, size=126, end=0x1132ff0, next=0x1131620

p=0x1131620, size=32, end=0x1131820, next=0x1131830

p=0x1131830, size=124, end=0x1131ff0, next=0x1131410

p=0x1131410, size=32, end=0x1131610, next=0x1131000

second: 8208 bytes (513 units) p=0x1133010

p=0x1135020, size=253, end=0x1135ff0, next=0x1132810

p=0x1132810, size=126, end=0x1132ff0, next=0x1131620

p=0x1131620, size=32, end=0x1131820, next=0x1131830

p=0x1131830, size=124, end=0x1131ff0, next=0x1131410

p=0x1131410, size=32, end=0x1131610, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1135020 <-- dummy

fifth free of over 2 pages p=0x1133010

p=0x1132810, size=126, end=0x1132ff0, next=0x1131620

p=0x1131620, size=32, end=0x1131820, next=0x1131830

p=0x1131830, size=124, end=0x1131ff0, next=0x1131410

p=0x1131410, size=32, end=0x1131610, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1135020 <-- dummy

p=0x1135020, size=253, end=0x1135ff0, next=0x1133000

p=0x1133000, size=513, end=0x1135010, next=0x1132810

fourth free of 1/2 of a page p=0x1132010

p=0x1131620, size=32, end=0x1131820, next=0x1131830

p=0x1131830, size=124, end=0x1131ff0, next=0x1131410

p=0x1131410, size=32, end=0x1131610, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1135020 <-- dummy

p=0x1135020, size=253, end=0x1135ff0, next=0x1133000

p=0x1133000, size=513, end=0x1135010, next=0x1132810

p=0x1132810, size=126, end=0x1132ff0, next=0x1132000

p=0x1132000, size=128, end=0x1132800, next=0x1131620

unit driver 4 has returned all memory to free list

p=0x1131620, size=32, end=0x1131820, next=0x1131830

p=0x1131830, size=124, end=0x1131ff0, next=0x1131410

p=0x1131410, size=32, end=0x1131610, next=0x1131000

p=0x1131000, size=64, end=0x1131400, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x1135020 <-- dummy

p=0x1135020, size=253, end=0x1135ff0, next=0x1133000

p=0x1133000, size=513, end=0x1135010, next=0x1132810

p=0x1132810, size=126, end=0x1132ff0, next=0x1132000

p=0x1132000, size=128, end=0x1132800, next=0x1131620

MP4 Heap Memory Statistics

Number of blocks in free list: 9

Min: 512

Max: 8208

Average: 2275

Total bytes in free list: 20480

Number of sbrk calls: 3

Number of requested pages: 5

Heap status: all memory is in the heap -- no leaks are possible

----- End unit test driver 4 -----

Test driver 4 examines how the free list is affected when we have several allocations and frees and do not allocate all memory before freeing all memory. So, we allocate 5 blocks of memory of varying sizes, and free all these blocks of memory in a different order, so that we have some memory being freed before other memory is allocated, leading to a spacing of free memory. It is of course important here that all memory is eventually returned to the free list but is also worth noting where new blocks of memory are placed. We see that when we use the best fit search policy, we have slightly better spacing amongst the blocks in our free list. This is an important consequence of the Rover pointer, which begins allocation at the location following the most recently accessed node in the list, rather than beginning every action at the dummy pointer, so that actions do not become so concentrated near the dummy pointer. We should also note that in this example, the effect of best fit is even more pronounced, in that it leads to less memory being requested by the free list. Since blocks are allocated more effectively with the best fit search policy, we conserve memory as much as possible which in this example this leads to the need to request only 4 pages of memory when using the best fit policy, while 5 pages of memory are required to fulfill all requests when the first fit or worst fit policies are used. This driver demonstrates primarily the benefits of the best fit search principle.

Unit Driver 5:

Example Output:

----- Begin unit driver 5 -----

There are 256 units per page, and the size of chunk_t is 16 bytes

p=0x15916d0, size=146, end=0x1591ff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x15916d0 <-- dummy

Here is the free list with all memeory allocated

first free of 1/8 of a page p=0x1591010

third free of 1/6 of a page p=0x15914d0

p=0x15916d0, size=146, end=0x1591ff0, next=0x1591000

p=0x1591000, size=32, end=0x1591200, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x15914c0 <-- dummy

p=0x15914c0, size=32, end=0x15916c0, next=0x15916d0

Notice the hole in free list where p2 has been allocated

second free of 1/8 of a page p=0x1591220

unit driver 5 has returned all memory to free list

p=0x1591000, size=32, end=0x1591200, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x15914c0 <-- dummy

p=0x15914c0, size=32, end=0x15916c0, next=0x15916d0

p=0x15916d0, size=146, end=0x1591ff0, next=0x1591210

p=0x1591210, size=42, end=0x15914b0, next=0x1591000

MP4 Heap Memory Statistics

Number of blocks in free list: 5

Min: 512

Max: 2336

Average: 819

Total bytes in free list: 4096

Number of sbrk calls: 1

Number of requested pages: 1

Heap status: all memory is in the heap -- no leaks are possible

----- End unit test driver 5 -----

Test driver 5 examines the shape and function of the free list when we free memory blocks that are not contiguous and are instead separated by a chunk of allocated memory. We allocate 3 blocks of memory, all in a single page, and then free the first and third blocks. We can then print out the free list, then free the remaining allocated block and print the list again. Comparing these two lists we see that the final printed list has contiguous blocks of memory. These may be ordered differently, but there are no gaps in the memory contained in the free list (except for the jump to and from the dummy block). In the previous list, though, we can see that there is a hole where the memory contained in the free list is not contiguous, and we can see that this is the memory that was given in the second allocation. Thus, test driver 5 demonstrates to us the structure and "appearance" of the free list.

Coalescing:

We implement the function of coalescing in the latter half of the test log. So, each test driver function is run again with coalescing enabled. We can see in these that, in general, we have fewer memory chunks in the free list after all memory is freed, since we will have coalesced memory together. We do still see though that we have the same amount of allocated memory and the same amount of page requests. These differences and similarities can be seen between each combination of test driver and search policy in the test log, but here we will examine coalescing through test driver 6, which was designed specifically to analyze coalescing.

Unit Driver 6:

Example Output:

----- Begin unit driver 6 -----

There are 256 units per page, and the size of chunk_t is 16 bytes

p=0x194d6d0, size=146, end=0x194dff0, next=0x6040b0

p=0x6040b0, size=0, end=0x6040b0, next=0x194d6d0 <-- dummy

Free list before any memory is freed.

MP4 Heap Memory Statistics

Number of blocks in free list: 2

Min: 2336

Max: 2336

Average: 1176

Total bytes in free list: 2352

Number of sbrk calls: 1

Number of requested pages: 1

Heap status: heap is in-use -- leaks are possible

second free of middle of page p=0x194d220

Free list after middle block is freed.

MP4 Heap Memory Statistics

Number of blocks in free list: 3

Min: 672

Max: 2336

Average: 1013

Total bytes in free list: 3040

Number of sbrk calls: 1

Number of requested pages: 1

Heap status: heap is in-use -- leaks are possible

third free of end of page p=0x194d4d0

Free list after end block is freed and coalesced.

MP4 Heap Memory Statistics

Number of blocks in free list: 2

Min: 3552

Max: 3552

Average: 1784

Total bytes in free list: 3568

Number of sbrk calls: 1

Number of requested pages: 1

Heap status: heap is in-use -- leaks are possible

first free of beginning of page p=0x194d010

unit driver 6 has returned all memory to free list

p=0x6040b0, size=0, end=0x6040b0, next=0x194d000 <-- dummy

p=0x194d000, size=255, end=0x194dff0, next=0x6040b0

MP4 Heap Memory Statistics

Number of blocks in free list: 2

Min: 4080

Max: 4080

Average: 2048

Total bytes in free list: 4096

Number of sbrk calls: 1

Number of requested pages: 1

Heap status: all memory is in the heap -- no leaks are possible

----- End unit test driver 6 -----

Test driver 6 was designed to demonstrate the function of coalescing in the free list. To do this, we allocate 3 contiguous blocks of memory, block 1, block 2, and block 3. We then free block 2, which is between blocks 1 and 3, and see that there is no coalescing since there are no contiguous free memory blocks. We then free block 3 and see that it coalesces with the free memory that was left as a remainder form the allocation of the 3 blocks, and that it coalesces with the contiguous and now free memory from block 2. Finally, we free block 1 and demonstrate that it will coalesce with the memory from blocks 2 and 3 and with the remainder memory and see that we avoid an error that might arise by attempting to coalesce with the dummy node when it is adjacent in the free list. We see this effect in the example given above and note that result of this coalescing is that the number of blocks in the free list after all memory has been freed is 2—the dummy block and one additional block which holds all of the memory that has been requested from the operating system. This is then incredibly efficient in terms of fulfilling new requests for memory.