System Programming
Lab #5

2020-04-22

sp-tas

Lab Assignment #3 : Malloc Lab

- Download skeleton code & pdf from eTL
  - malloclab-handout.tar, malloclab-handout.pdf
- Hand In
  - Upload your files eTL
    - 압축파일 양식 : [학번]_[이름]_malloclab.tar (or .zip, etc)
    - Ex) 2020-12345_홍길동_malloclab.tar
  - A zip file should include
  - (1)mm.c (2)report
    - mm.c 양식 : mm-[학번].c eg) mm-2020-12345.c (제출할때만 바꿔서)
    - Report 양식 : [학번]_[이름]_malloclab_report.pdf (or .hwp, .txt etc)
- Please, READ the Hand-out thoroughly!
- Assigned : April 22
- Deadline : May 6, 23:59:59
- Delay policy : Same as before
- Next week's Lab
  - Malloclab Q&A session

Before we start

- We need to know
  - 1. Macros
  - 2. Pointer Arithmetic

Macros

- Why macros?
  - 1. Faster than function calls
  - 2. Encapsulate pointer arithmetic code lines
    - pointer arithmetic is error-prone & confusing
- Differences (with inline functions)
  - Macros are done with preprocessor (before compile time)
  - No call, return overheads
- Drawbacks
  - Less expressive than functions
  - Arguments are not type-checked
  - Unintended side effects
    - Ex) #define xsquared(x) (x*x)
    - What happens when xsquared(x++) is called?
  - Use () frequently!!

In the textbook

code/vm/malloc/mm.c

```
1  /* Basic constants and macros */
2  #define WSIZE       4       /* Word and header/footer size (bytes) */
3  #define DSIZE       8       /* Double word size (bytes) */
4  #define CHUNKSIZE  (1<<12)  /* Extend heap by this amount (bytes) */
5
6  #define MAX(x, y) ((x) > (y)? (x) : (y))
7
8  /* Pack a size and allocated bit into a word */
9  #define PACK(size, alloc)  ((size) | (alloc))
10
11 /* Read and write a word at address p */
12 #define GET(p)       (*(unsigned int *)(p))
13 #define PUT(p, val)  (*(unsigned int *)(p) = (val))
14
15 /* Read the size and allocated fields from address p */
16 #define GET_SIZE(p)  (GET(p) & ~0x7)
17 #define GET_ALLOC(p) (GET(p) & 0x1)
18
19 /* Given block ptr bp, compute address of its header and footer */
20 #define HDRP(bp)       ((char *)(bp) - WSIZE)
21 #define FTRP(bp)       ((char *)(bp) + GET_SIZE(HDRP(bp)) - DSIZE)
22
23 /* Given block ptr bp, compute address of next and previous blocks */
24 #define NEXT_BLKP(bp)  ((char *)(bp) + GET_SIZE(((char *)(bp) - WSIZE)))
25 #define PREV_BLKP(bp)  ((char *)(bp) - GET_SIZE(((char *)(bp) - DSIZE)))
```

code/vm/malloc/mm.c

Figure 9.43 Basic constants and macros for manipulating the free list.

Pointer Arithmetic

- Pointers are 4-byte numbers
  - Use unsigned int / int to get 4-byte numbers

- Read, write address as below

```
/* Read and write a word at address p */
#define GET(p)        (*(unsigned int *)(p))
#define PUT(p, val)   (*(unsigned int *)(p) = (val))
```

Pointer Arithmetic

- What does ptr + a mean?
  - (type_1)* ptr1 = some value;
  - (type_1)* ptr2 = ptr + a;
- This is really computing :
  - **ptr2 = ptr1 + (a * sizeof(type_a));**
  - **lea (ptr1, a, sizeof(type_a)) , ptr2;**
- Practice
  - int * ptr = (int *) 0x12341230;
  - int * ptr2 = ptr + 1;

  - char *ptr = (char *) 0x12341230;
  - char * ptr2 = ptr + 1;

Pointer Arithmetic

- What does ptr + a mean?
  - (type_1)* ptr1 = some value;
  - (type_1)* ptr2 = ptr + a;
- This is really computing :
  - **ptr2 = ptr1 + (a * sizeof(type_a));**
  - **lea (ptr1, a, sizeof(type_a)) , ptr2;**
- Practice
  - int * ptr = (int *) 0x12341230;
  - int * ptr2 = ptr + 1; //ptr2 = 0x12341234
  - char *ptr = (char *) 0x12341230;
  - char * ptr2 = ptr + 1; //ptr2 = 0x12341231

Pointer Arithmetic

- What does below mean?

```
23   /* Given block ptr bp, compute address of next and previous blocks */
24   #define NEXT_BLKP(bp)  ((char *)(bp) + GET_SIZE(((char *)(bp) - WSIZE)))
25   #define PREV_BLKP(bp)  ((char *)(bp) - GET_SIZE(((char *)(bp) - DSIZE)))
```

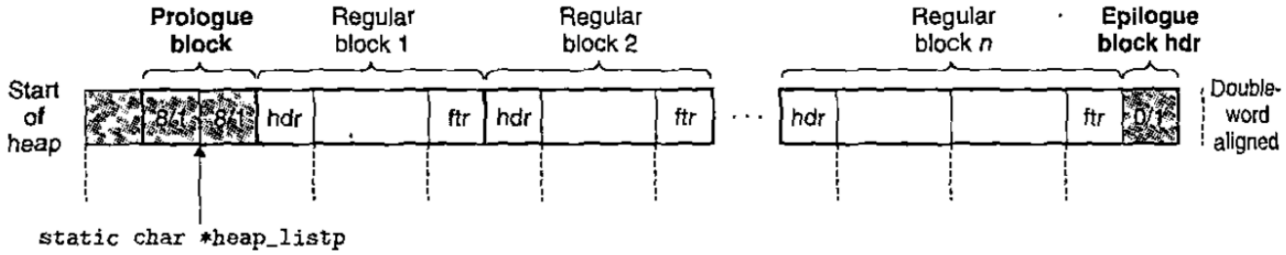
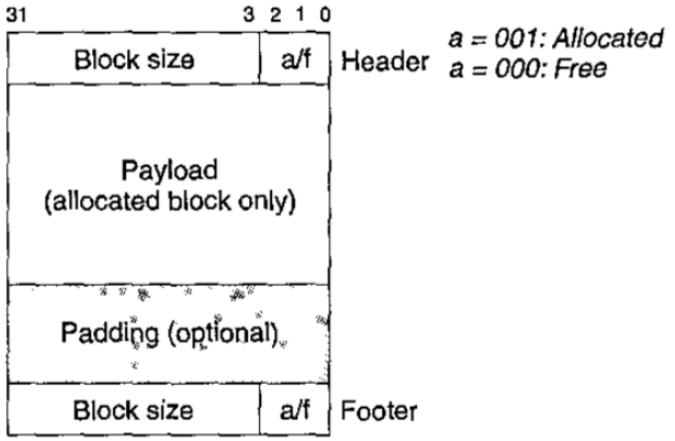


Figure 9.42 Invariant form of the implicit free list.

Figure 9.39
Format of heap block that uses a boundary tag.



Lab Assignment #3 : Malloc Lab

- Download skeleton code & pdf from eTL
  - malloclab-handout.tar, malloclab-handout.pdf
- Hand In
  - Upload your files eTL
    - 압축파일 양식 : [학번]_[이름]_malloclab.tar (or .zip, etc)
    - Ex) 2020-12345_홍길동_malloclab.tar
  - A zip file should include
  - (1)mm.c (2)report
    - mm.c 양식 : mm-[학번].c eg) mm-2020-12345.c (제출할때만 바꿔서)
    - Report 양식 : [학번]_[이름]_malloclab_report.pdf (or .hwp, .txt etc)
- Please, READ the Hand-out thoroughly!
- Assigned : April 22
- Deadline : May 6, 23:59:59
- Delay policy : Same as before
- Next week's Lab
  - Malloclab Q&A session

Malloc Lab Preview

- Implementing your own dynamic storage allocator
  - mm_init, mm_malloc, mm_free, mm_realloc
- Ways to keep track of free, allocated blocks of memory
  - Implicit linked list of blocks
  - Explicit linked list of free blocks
  - Segregated lists of different size free blocks
- Other design decisions :
  - How to look for free blocks? (First fit, next fit, best fit, …)
  - Should the linked lists be doubly linked?
  - When to coalesce blocks?

Support Routines

- Functions you can use (implemented in memlib.c)

- `void *mem_sbrk(int incr)`: Expands the heap by `incr` bytes, where `incr` is a positive non-zero integer and returns a generic pointer to the first byte of the newly allocated heap area. The semantics are identical to the Unix `sbrk` function, except that `mem_sbrk` accepts only a positive non-zero integer argument.
- `void *mem_heap_lo(void)`: Returns a generic pointer to the first byte in the heap.
- `void *mem_heap_hi(void)`: Returns a generic pointer to the last byte in the heap.
- `size_t mem_heapsize(void)`: Returns the current size of the heap in bytes.
- `size_t mem_pagesize(void)`: Returns the system's page size in bytes (4K on Linux systems).

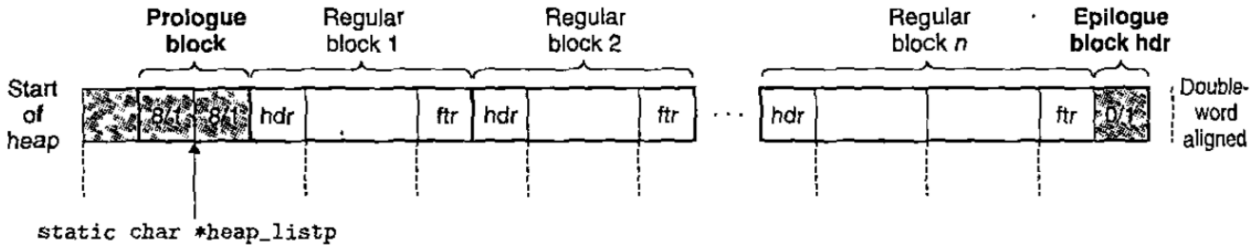


Figure 9.42 Invariant form of the implicit free list.

Watch Out!

7 Programming Rules

- You should not change any of the interfaces in `mm.c`.
- You should not invoke any memory-management related library calls or system calls. This excludes the use of `malloc`, `calloc`, `free`, `realloc`, `sbrk`, `brk` or any variants of these calls in your code.
- You are not allowed to define any global or `static` compound data structures such as arrays, structs, trees, or lists in your `mm.c` program. However, you *are* allowed to declare global scalar variables such as integers, floats, and pointers in `mm.c`.
- For consistency with the `libc malloc` package, which returns blocks aligned on 8-byte boundaries, your allocator must always return pointers that are aligned to 8-byte boundaries. The driver will enforce this requirement for you.

mm.c 이외의 파일들은 수정하지 마세요.

Testing with trace files

- `-t <tracedir>`: Look for the default trace files in directory `tracedir` instead of the default directory defined in `config.h`.
- `-f <tracefile>`: Use one particular `tracefile` for testing instead of the default set of trace-files.
- `-h`: Print a summary of the command line arguments.
- `-l`: Run and measure `libc` malloc in addition to the student's malloc package.
- `-v`: Verbose output. Print a performance breakdown for each tracefile in a compact table.
- `-V`: More verbose output. Prints additional diagnostic information as each trace file is processed. Useful during debugging for determining which trace file is causing your malloc package to fail.

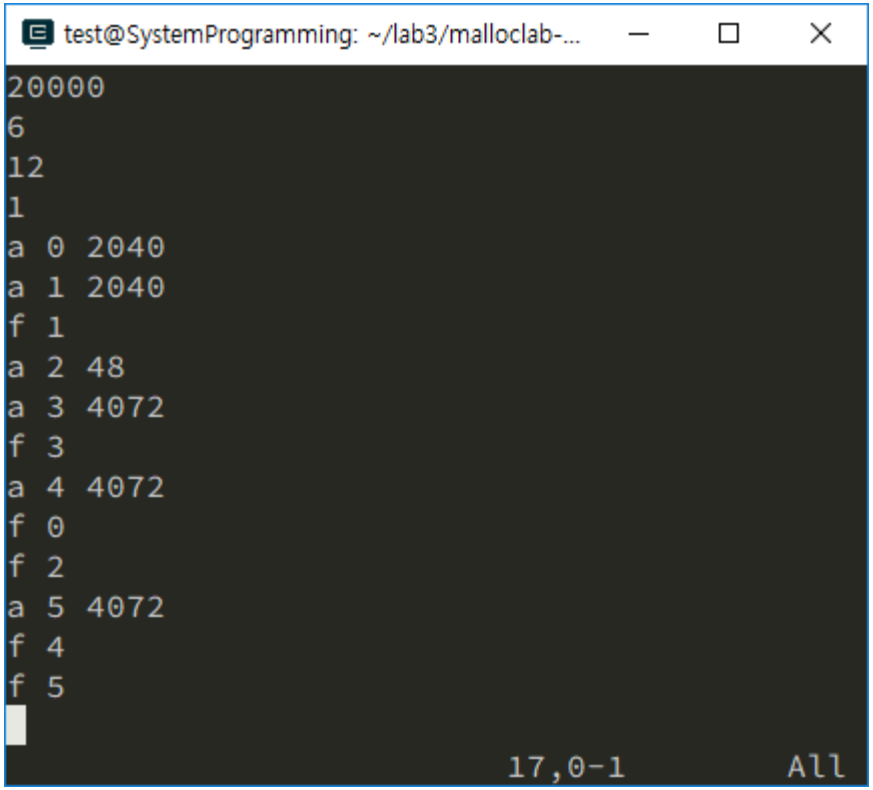
```
ta@sp3:~/yschoi/testing/malloclab-handout/src$ ./mdriver -f ./traces/short1.rep  -V
Team Name:implicit first fit
Member 1 :Dave OHallaron:droh
Measuring performance with gettimeofday().

Testing mm malloc
Reading tracefile: ./traces/short1.rep
Checking mm_malloc for correctness, efficiency, and performance.

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   66%      12  0.000001 20000
Total          66%      12  0.000001 20000

Perf index = 40 (util) + 40 (thru) = 80/100
```

Testing with trace files

- Heap size(unused)
- Ids
- Ops
- Weight(unused)

- Op id size
- …

- A – alloc
- F – free
- R – realloc



Short1.rep

Testing with trace files

```
test@SystemProgramming: ~/lab3/malloclab-handout/src
test@SystemProgramming:~/lab3/malloclab-handout/src$ ./mdriver -V
Using default tracefiles in ./traces/
Measuring performance with gettimeofday().

Testing mm malloc
Reading tracefile: amptjp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cccp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cp-decl-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: expr-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: coalescing-bal.rep
ERROR: mem_sbrk failed. Ran out of memory...
Checking mm_malloc for correctness, ERROR [trace 4, line 7673]: mm_malloc failed.
Reading tracefile: random-bal.rep
ERROR: mem_sbrk failed. Ran out of memory...
Checking mm_malloc for correctness, ERROR [trace 5, line 1662]: mm_malloc failed.
Reading tracefile: random2-bal.rep
ERROR: mem_sbrk failed. Ran out of memory...
Checking mm_malloc for correctness, ERROR [trace 6, line 1780]: mm_malloc failed.
Reading tracefile: binary-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: binary2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc-bal.rep
ERROR: mem_sbrk failed. Ran out of memory...
Checking mm_malloc for correctness, ERROR [trace 9, line 1705]: mm_realloc failed.
Reading tracefile: realloc2-bal.rep
ERROR: mem_sbrk failed. Ran out of memory...
Checking mm_malloc for correctness, ERROR [trace 10, line 6562]: mm_realloc failed.

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   23%    5694  0.000068 83246
 1       yes   19%    5848  0.000070 83782
 2       yes   30%    6648  0.000083 79712
 3       yes   40%    5380  0.000063 85942
 4        no     -       -         -     -
 5        no     -       -         -     -
 6        no     -       -         -     -
 7       yes   55%   12000  0.000152 79051
 8       yes   51%   24000  0.000154155945
 9        no     -       -         -     -
10        no     -       -         -     -
Total           -       -         -     -

Terminated with 5 errors
test@SystemProgramming:~/lab3/malloclab-handout/src$
```

./mdriver -V

Evaluation

- Our evaluation
  - Correctness(20 points)
  - Performance(100 points)
    - Space utilization + Throughput
  - Style(10 points)
  - Report(10 points)

Last year's Questions

- Trace file의 경로변경은 어떻게 하나요?
  - ->src/config.h
- mm_malloc의 input이 0인 경우의 return값?
  - -> NULL.
- Global/Static으로 array 선언하면 안되나요?
  - -> 네 안됩니다.
- 고친 부분이 없는데 Trace file을 돌릴 때 가끔 점수가 다르게 나옵니다.
  - -> 서버의 CPU 사용량에 따라 점수가 다르게 나올 수 있습니다.

How to begin

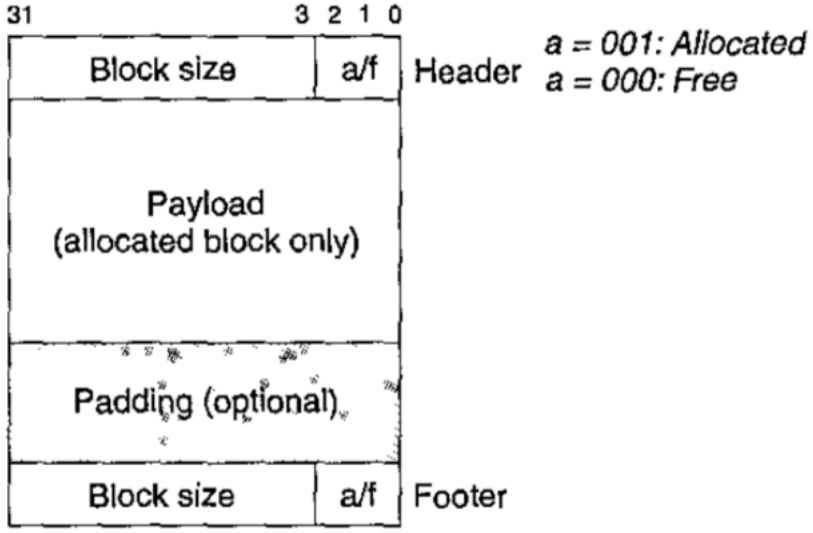
- From handout.pdf, 10.hint

- *Understand every line of the malloc implementation in the textbook.* The textbook has a detailed example of a simple allocator based on an implicit free list. Use this is a point of departure. Don't start working on your allocator until you understand everything about the simple implicit list allocator.

- **Workflow**
- 1. Understand textbook
- 2. Implement mm.c with implicit list
- 3. modify your implementation (to explicit, segregated)
- 4. use diff policies (Next-fit, First-fit, … / When to coalesce / etc)
- 5. use diff data structures

Understanding Textbook

Figure 9.39
Format of heap block that uses a boundary tag.



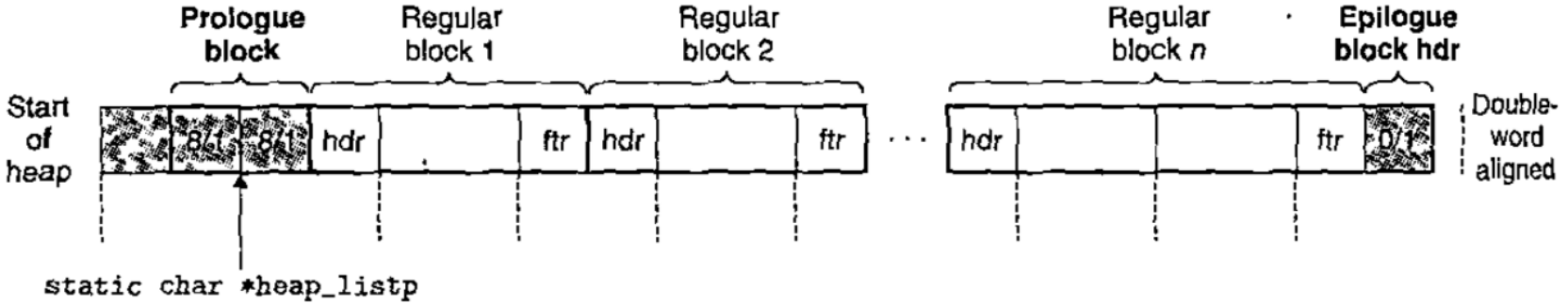


Figure 9.42 Invariant form of the implicit free list.

Understanding Textbook

code/vm/malloc/mm.c

```
1   int mm_init(void)
2   {
3       /* Create the initial empty heap */
4       if ((heap_listp = mem_sbrk(4*WSIZE)) == (void *)-1)
5           return -1;
6       PUT(heap_listp, 0);                          /* Alignment padding */
7       PUT(heap_listp + (1*WSIZE), PACK(DSIZE, 1)); /* Prologue header */
8       PUT(heap_listp + (2*WSIZE), PACK(DSIZE, 1)); /* Prologue footer */
9       PUT(heap_listp + (3*WSIZE), PACK(0, 1));     /* Epilogue header */
10      heap_listp += (2*WSIZE);
11
12      /* Extend the empty heap with a free block of CHUNKSIZE bytes */
13      if (extend_heap(CHUNKSIZE/WSIZE) == NULL)
14          return -1;
15      return 0;
16  }
```

code/vm/malloc/mm.c

Figure 9.44 `mm_init` creates a heap with an initial free block.

끝

- 과제 기한 5월 6일까지
- 질문
  - etl Q&A 게시판
  - ta_sp20@dcslab.snu.ac.kr
- 다음 시간에
  - Q&A