운영체제 과제 #2. Load time Function Interposition, ku\_malloc

컴퓨터공학과 201511269 송상연

**Basic Design**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9 | typedef struct \_\_hnode\_t { //size 16      size\_t size;      struct \_\_hnode\_t \*next;  } hnode\_t;    typedef struct \_\_hheader\_t { //size 16      size\_t size;      unsigned long magic;  } hheader\_t; | [cs](http://colorscripter.com/info#e) |

우선 ku\_malloc.h에는 다음과 같이 free한 부분에 대한 헤더 정보를 가지고 있는 구조체인 hnode\_t와 malloc한 부분에 대한 헤더 정보를 가지고 있는 구조체인 hheader\_t가 선언되어 있다.

전역변수로는 hnode\_t \*형인 free\_list와, next-fit을 위해 마지막으로 할당된 free\_list의 노드를 기억할 hnode\_t \*형인 alloc\_last\_node가 있다.

테스트코드에서 malloc을 호출하면, ku\_malloc.c에 있는 malloc함수를 이용하여, size만큼 할당한다.(사실상은 hheader\_t의 크기 + size가 할당된다.) 여기서 아직 free list가 만들어지지 않은 경우에는 sbrk(할당할 크기)로 heap의 크기를 늘려준다. 만약 sbrk로 할당할 수 없는 크기라면, sbrk함수가 (void \*)-1을 반환하는데, 그걸 이용하여 malloc함수에서는 NULL을 반환한다. 그리고 free list가 만들어진 경우에는, free list의 노드의 size와 할당해야 할 크기를 비교하여 할당해야 할 크기가 노드의 size보다 작은 경우, 그리고 next-fit algorithm을 고려하여 우선적으로 alloc\_last\_node가 아닌 경우에 그 노드에 할당한다. 그리고 할당될free list 노드의 size와 할당할 size를 비교했을 때, 그 차이가 sizeof(hnode\_t)보다 작거나 같을 경우, 그 free list 노드 전부를 할당하고, 아닐 경우 splitting을 한다. Splitting을 할 경우, hheader\_t \*형인 hptr이 그 할당될 free list노드의 주소를 가리키고, ptr은 hptr+sizeof(hheader\_t)를 가리킨다. Free list의 해당 노드의 주소는 ptr+size로 변경되며, 그 노드의 전 노드인 prevNode->next는 그 주소로 변경된다. Bool형 변수인 nextfit을 선언하였는데, 초기화를 false로 하고, next-fit을 성공한 경우, true로 바꿔주고, false인 경우, 먼저 alloc\_last\_node이후에 할당해야 할 크기보다 큰 free list node가 없다고 여겨 free list의 처음부터 다시 체크하여 할당해야 할 크기보다 큰 free list node에 할당한다, 그 경우가 아니라면, free list의 모든 노드가 할당해야 할 크기보다 작다고 여겨 sbrk를 이용해 할당한다.

테스트코드에서 free를 호출하면, ku\_malloc.c에 있는 free함수로부터 넘어온 ptr의 hheader\_t부분의 magic number가 ptr ^ KU\_MAGIC임을 확인하고, hnode\_t형인 newNode를 추가하여 hptr->size를 newNode.size로, 전역변수인 free\_list를 newNode.next로 선언함으로써 free list의 head부분에 newNode를 넣어주고, hheader\_t를 hnode\_t로 변환한다. 그리고 free list에서 노드끼리 연결되는 부분을 coalescing한다.(coalescing은 구현하지 못했습니다..)

**Description for important functions**

Function Name : malloc

Functionality : 여러가지 경우의 수에 맞게 hheader\_t \*hptr의 size와 magic에 알맞는 값을 넣어준다, free list가 없을 시에는 sbrk로 heap의 크기를 늘려서 할당하고, free list가 있을 시에는 next-fit algorithm을 이용하여 free list에 인자로 넘어온 size만큼 할당하고, splitting한다. Free list의 모든 노드가 인자로 넘어온 size보다 작다면, sbrk로 heap의 크기를 늘려준다. 만약 sbrk로도 할당할 수 없는 크기라면, NULL을 반환한다.

Parameters : size\_t size

Return Value : void \*

Function Name : free

Functionality : 매개변수인 ptr의 hheader\_t \*hptr의 정보를 읽어 magic number가 일치할 경우, hheader\_t \*자료형을 hnode\_t \*자료형으로 변환하여 free list node의 size를 설정한 후, free list의 Head부분에 넣는다, 만약 free list node끼리 연결되는 노드가 있다면, coalescing한다.

Parameters : void \*ptr

Return Value : void