시스템프로그래밍 project 4

20210428 정현정

1. 개발 목표

C 프로그래밍에서 사용되는 dynamic memory allocator를 구현한다. 즉, 기존의 동적할당과 관련된 함수를 사용하지 않고 malloc, free, realloc 등의 함수들을 직접 구현한다. 이때 explicit list를 활용한다.

2. 개발 범위 및 내용

A. 개발 범위

아래 4개의 함수를 구현한다.

```
int mm_init (void);
void *mm_malloc (size_t size);
void mm_free (void *ptr);
void *mm_realloc(void *ptr, size_t size);
```

B. 개발 내용

개발 범위에 명시되어 있는 4개의 함수를 mm.c에 구현한다.

- mm_init

mm_init 함수는 초기 힙 영역을 설정한다. 프롤로그 및 에필로그 블록을 설정하고, 힙을 일정 크기만큼 확장한다. 모든 과정이 성공하면 0을 반환, 실패하면 -1을 반환한다.

- mm_malloc

mm_malloc 함수는 요청된 크기의 메모리를 할당한다. 적절한 크기의 블록을 찾고, 찾을 블록을 할당하여 그 주소값을 반환한다. 만약 적절한 크기의 블록을 찾지 못하면 힙 영역을 확장하여 새로운 블록을 할당하고 그 주소값을 반환한다. 힙 영역 확장에 실패하면 NULL을 반환한다.

- mm_free

mm_free 함수는 인자로 받은 주소의 블록을 해제하고 free block을 모아놓은 형태인 free list에 추가한다. 이때 전후로 free block이 있으면 병합하여 크기가 큰 free block을 생성한다.

- mm_realloc

mm_realloc 함수는 인자로 받은 포인터가 가르키는 블록의 크기를 변경한다. 요청된 크기가 0이면 메모리 해제, ptr이 NULL이면 기존의 malloc과 동일한 기능을 수행한다.

C. 개발 방법

a. subroutines, global variable에 대한 설명

- static char *free_listp;

free block 의 linked list 를 가리키는 포인터이다. mm_init 함수에서 NULL 로 초기화를 수행한다.

- static void *extend_heap(size_t words);

힙 영역을 확장한다. 주어진 words 만큼 힙을 늘리고 생성된 새로운 블록을 free list 에 추가하고 병합한다.

- static void *find_fit(size_t asize);

asize 에 맞는 크기의 free block 을 찾아 제공한다. 이때 asize 는 사전에 alignment 를 수행한 값이다. free list 는 block 의 사이즈가 작은 것부터 오름차순으로 구성되어 있어, 순차적으로 free list 를 탐색하며 사이즈가 asize 보다 크거나 갖은 것 중 최초로 매칭되는 block 을 찾는다.

- static void *place(void *bp, size_t asize);

인자로 받은 bp 에 asize 바이트만큼 block 을 할당한다. 만약 해당 block 에 할당 후 남은 바이트수가 16 바이트보다 작으면 split 할 수 없으므로 그 전부를 할당하고, split 이 가능하다면 split 수행 후 새롭게 free list 에 추가한다. 만약 24

바이트를 넘어간다면 탐색의 효율성을 위해 뒤로 뺀다. 이때 24 바이트는 여러 번 mdriver 를 수행하면서 결과가 최적인 값을 선택한다.

- static void *coalesce(void *bp);

bp 전후로 free block 들을 찾아 병합한다. 전후가 모두 비어있는 경우, 뒤만 비어있는 경우, 앞만 비어있는 경우, 모두 비어있지 않은 경우로 나누어 병합을 수행한다.

- static void delete_free_list(void *bp);

bp 가 가리키는 block 을 free list 에서 삭제한다. free list 의 첫 block 인 경우와 아닌 경우로 나누어서 linked list 를 수정한다.

- static void add_free_list(size_t size, void *bp);

explicit list 로 구현된 free list 를 탐색하며 size 만큼의 공간을 할당할 수 있는 free block 을 찾는다. 이때 free block 들의 사이즈가 오름차순으로 정렬되도록 삽입하여 탐색의 효율성을 높였다. free list 의 첫 block 인 경우와 그렇지 않은 경우로 나누어서 linked list 를 수정한다.

- static char *extend_heap_2(char *bp, size_t asize, int *left);

bp 와 alignment 를 수행한 size 를 입력 받아 필요한 경우에만 힙 영역을 확장한다. left 는 남은 영역의 크기를 의미한다.

- void mm_free(void *bp)

전달받은 bp 의 block 을 해제한다. 이때 기존 강의자료의 코드에 free list 에 add 하는 부분을 추가적으로 작성하였다.

- void *mm_realloc(void *ptr, size_t size)

인자로 받은 ptr 이 포인터가 NULL 이면 mm_malloc 과 동일한 작업을 수행한다. 만약, size 가 0 이면 mm_free 와 동일한 작업을 수행한다. 따라서 mm_realloc 함수에서 내부적으로 mm_free 함수를 호출한다. 그 외의 경우 realloc 을 진행한다.

현재 위치한 block 의 size 와 변경하고자 하는 size 를 비교하여 새로운 size 가 더 큰 경우에만 realloc 를 수행한다. 만약 현재 위치한 block 바로 다음 block 이 비어있는 상태이고 병합하였을 때 새로운 size 를 할당할 수 있을만큼 크기가 충분하다면 새로운 메모리 할당을 수행하지 않고 병합하여 realloc 을 수행한다. 이외의 경우에는 mm_malloc 함수를 호출하여 새로운 메모리를 할당하고 기존 메모리 내용을 복사하며 기존 메모리를 할당 해제한다.

- #define NEXT_FREE(bp) (*(char **)(bp))
  free list 에서 다음 free block 의 포인터를 반환한다.

- #define PREV_FREE(bp) (*(char **)(bp + WSIZE))
  free list 에서 이전 free block 의 포인터를 반환한다.

D. 개발 결과 및 검증

mdriver 프로그램으로 성능을 검증한다. mdriver -v를 실행했을 때의 결과는 다음과 같다.

- local에서 수행한 경우

```
[20210428]::NAME: Hyeonjung Jeong, Email Address: cathy2750@sogang.ac.kr
Using default tracefiles in ./tracefiles/
Measuring performance with gettimeofday().

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   99%    5694  0.000754  7550
 1       yes   99%    5848  0.000501 11680
 2       yes   99%    6648  0.000914  7270
 3       yes   99%    5380  0.000687  7837
 4       yes   97%   14400  0.000288 50000
 5       yes   96%    4800  0.014099   340
 6       yes   95%    4800  0.013369   359
 7       yes   61%   12000  0.079474   151
 8       yes   88%   24000  0.015910  1508
 9       yes   83%   14401  0.000258 55731
10       yes   85%   14401  0.000190 75835
Total          91%  112372  0.126443   889

Perf index = 55 (util) + 40 (thru) = 95/100
```

- cspro 서버에서 수행한 경우

```
Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   99%    5694  0.003144  1811
 1       yes   99%    5848  0.001077  5428
 2       yes   99%    6648  0.002368  2807
 3       yes   99%    5380  0.002241  2400
 4       yes   97%   14400  0.001978  7279
 5       yes   96%    4800  0.050095    96
 6       yes   95%    4800  0.020303   236
 7       yes   61%   12000  0.287378    42
 8       yes   88%   24000  0.111076   216
 9       yes   83%   14401  0.000434 33205
10       yes   85%   14401  0.000271 53121
Total          91%  112372  0.480365   234

Perf index = 55 (util) + 16 (thru) = 70/100
cse20210428@cspro:~/prj4-malloc$
```

- flow chart