Multicore Programming Project 3

담당 교수 : 최재승

이름 : 김태곤

학번 : 20191583

1. 개발 목표

C 프로그램을 위한 dynamic memory allocator인 malloc, free, realloc 함수를 작성한다. CSAPP에서 제공하는 implicit 방법이 아닌, explicit list 방식과 여러 policy를 적용하여 효율을 높인다.

2. 개발 범위 및 내용

A. 개발 범위

Malloc, free, realloc 함수를 구현한다.

B. 개발 내용

Explicit free list 방식을 이용하여 속도 측면에서 효율을 높였다. Free list를 구현하기 위하여 free list를 관리하는 함수를 구현하여 추가 및 삭제하였다.

C. 개발 방법

C.1 변수

Global 변수로 heap_listp와 free_listp를 설정하였다.

C.2 Macro

Free_list를 구현하기 위한 Macro 함수를 추가하였다.

GET_PREV(bp) : 이전 free list의 block pointer를 나타낸다.

GET_NEXT(bp) : 다음 free list의 block pointer를 나타낸다.

SET_PREV(bp,val) : 이전 free list block의 값을 변경한다.

SET_NEXT(bp,val) : 다음 free list block의 값을 변경한다.

이 외의 Macro는 기존 제공된 code와 동일하다.

C.3 추가된 Function

remove_free_list : 이전 free list와 다음 free list를 연결해 제거한다.

add_free_list : block의 이전 포인터를 현재 free list의 헤드로 설정하고, block point를 NULL로 설정한다. 이후 free list의 헤드를 block으로 설정하

여 free list의 가장 앞부분에 block을 추가한다.

C.4 변경된 Function

Coalesce(bp) : 101인 경우(prev_alloc && next_alloc) add_free_list 함수를 이용하여 free list에 추가한다. 100인 경우 remove_free_list(NEXT_BLKP)를 사용하여 다음 block을 제거하고, GET_SIZE를 통해 크기를 알아낸다. 이후 HD와 FT을 설정하고, size에 그 값을 더해 add_free_list로 free list에 추가한다. 001인 경우 이전 block의 크기만큼 size를 더하고, 새로운 free block의 HD아 FT을 설정하고, bp를 이전 block 가리키게 한다. 000인 경우 100과 001을 합친 과정을 거친다.

Realloc(ptr,size) : 먼저 기존 malloc함수에서 구한 방법대로 asize의 크기를 구한다. 이후 현재 block의 size와 비교하여 asize가 작거나 같은 경우 현재 메모리 block을 그대로 반환한다. 그렇지 않을 경우 다음 block의 할당 여부와 크기를 확인한다. 다음 block이 할당되지 않고, 현재 block의 크기와 다음 block의 크기를 합친 크기가 assize 이상이면, 다음 block을 free list에서 제거하고, 현재 block의 크기를 확장한다. 그 외의 경우, 새로운 메모리 block을 mm_malloc으로 할당한 후, 이전 메모리 block에서 새로운 메모리 block으로 데이터를 복사하고, 이전 메모리 block을 해제한다. 이후 새로 할당된 메모리 block을 반환한다. 이 과정을 통해 메모리 block의 크기를 조정하고, 필요에 따라 새로운 메모리를 할당하거나 기존 메모리를 해제하는 작업을 수행한다.

Place(bp,asize) : csize-asize가 2*DISZE보다 크거나 같으면 현재 bp의 HD와 FT을 0으로 설정하고, 다음 bp의 HD와 FT를 1로 설정한다. 그렇지 않을 경우 remove_free_list(bp)를 하고 HD와 FT를 1로 설정한다.

C.5 제공된 code에서 수정이 거의 없는 Function

Mm_init, mm_malloc, mm_free, extend_heap, find_fit 함수들은 기존에 제공된 코드와 거의 동일하다. Mm_init 함수에는 free_listp를 초기화해주는 과정만 추가되었다. Find_fit 함수는 Best Fit으로 작성하고자 하였으나, 평가 점수가 더 떨어지는 결과가 발생해 그냥 code에서 제공한 first fit policy를 적용하였다. 그 외의 mm_malloc, mm_free, extend_heap 함수는 제공된 code와 동일하다.

3. 구현 결과

```
[cse20191583@cspro:~/multicore/prj3-malloc$ ./mdriver -V
[20191583]::NAME: Taegon Kim, Email Address: tg85911@naver.com
Using default tracefiles in ./tracefiles/
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
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
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
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   95%    5694  0.000387 14721
 1       yes   95%    5848  0.000421 13891
 2       yes   96%    6648  0.000408 16286
 3       yes   98%    5380  0.000300 17945
 4       yes   97%   14400  0.000221 65306
 5       yes   91%    4800  0.000694  6917
 6       yes   87%    4800  0.000670  7162
 7       yes   81%   12000  0.015312   784
 8       yes   56%   24000  0.169175   142
 9       yes  100%   14401  0.000141102207
10       yes   85%   14401  0.000117123296
Total          89%  112372  0.187845   598

Perf index = 53 (util) + 40 (thru) = 93/100
```

총 평가점수 93점으로 이번 프로젝트의 목표 점수였던 90점을 성공적으로 달성하였다.