System Programming Project 4

담당 교수 : 김영재

이름 : 김효림

학번 : 20221549

1. 개발 목표

explicit free list 방식을 이용해 dynamic memory allocator를 구현했다.

2. 개발 범위 및 내용

A. 개요

heap_listp를 전역변수로 선언하여 전체적인 free list를 관리하였다.

교과서에 있는 매크로를 사용하였는데, 이중 중요한 매크로는 ALIGNMENT(블록의 정렬 기준, 8 BYTE), CHUNKSIZE(heap 확장 크기, 4096 BYTE), WSIZE(word size, 4 BYTE), DSIZE(double word size, 8 BYTE)가 있다.

B. 변수 및 함수 설명

- mm_init : 힙을 초기화한다. 블록을 설정하여 Free list를 생성한다.
- mm_malloc : 적절한 free list의 공간을 찾아 메모리를 할당하거나, 적절한 공간이 없을 경우 힙을 확장하여 메모리를 할당한다. ALIGN과 SIZE_T_SIZE macro를 이용하여 사이즈를 조정하였다.
- mm_free: 메모리 블록을 해제한다. Header와 Footer tag에 0을 할당하여, 메모리 공간이 비어있음을 표시한 후 coalesce를 진행한다.
- mm_realloc : 원하는 크기로 메모리 블록의 크기를 조정한다. 명세서에 따라 oldptr이 NULL인 경우 mm_malloc(size)를 실행하며, size가 0인 경우 mm_free(oldptr)을 실행한다. 마지막으로 oldptr이 NULL이 아닌 경우 세 가지 케이스로 나누어 함수가 실행된다.
  - ⇨ In-place Expansion : 새로운 크기가 현재 블록 크기 이하인 경우 블록을 그대로 사용한다.
  - ⇨ Coalescing with Next Block : 다음 블록이 비어 있고, 병합한 크기가 필요한 크기 보다 크다면 병합을 진행한다.
  - ⇨ Allocate New Block : 그 외의 경우, 새로운 블록을 할당하여 데이터를 복사한다. 그 후 기존 블록은 mm_free를 통해 해제한다.
- extend_heap : mm_sbkr 함수를 통해 힙을 확장한다. 첫 번째 블락의 Header

와 Footer에 크기를 저장하고, epilogue header를 할당해 준다. free list를 병합하기도 한다.

- Coalesce : free list의 인접한 블록을 병합한다. 이전 공간이 free된 경우, 다음 공간이 free된 경우, 앞뒤 공간이 모두 free된 경우에 따라서 다른 방식으로 진행된다.
- find_fit : 주어진 크기에 적합한 free list의 block을 찾는다. First fit 방식으로 구현되었으며, free list를 순차적으로 검색하여 요청한 크기 이상인 첫 번째 블록을 할당한다.
- Place : free list block을 allocated block으로 변환한다. 현재 사이즈와 비교하여 분할이 필요하다면 블락은 분할하여 메모리 공간을 새롭게 할당하기도 한다.
- insert_Block : 새로운 블록을 free list에 넣는다.
- remove_Block : free list에서 block을 제거한다.

3. 구현 결과

```
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   89%    5694  0.002901  1963
 1       yes   92%    5848  0.000493 11864
 2       yes   94%    6648  0.003381  1966
 3       yes   96%    5380  0.002154  2498
 4       yes   66%   14400  0.000272 52961
 5       yes   88%    4800  0.006457   743
 6       yes   85%    4800  0.003679  1305
 7       yes   55%   12000  0.022604   531
 8       yes   51%   24000  0.032350   742
 9       yes   97%   14401  0.000232 62073
10       yes   38%   14401  0.000254 56719
Total          77%  112372  0.074776  1503

Perf index = 46 (util) + 40 (thru) = 86/100
```