System Programming Project 4

담당 교수 : 박성용 교수님

이름 : 이민석

학번 : 20201614

1. 개발 목표

- 이번 프로젝트는 C언어를 이용하여 직접 'malloc()', 'free()', 'realloc()' 함수를 구현하는 것이다. 기존의 'stdlib.h'의 라이브러리의 'malloc()', 'free()', 'realloc()' 함수들과 기능을 같게 하며, 동적 메모리 할당을 효율적으로 할 수 있도록 만드는 것이 목표이다. 이 때 효율적인 메모리 관리 성능 지표로 Space Utilization Optimization과 Throughput Optimization을 고려하게 된다. 즉, 메모리 할당 시 내부 및 외부 단편화를 최소화하여, 사용 가능한 메모리를 최대한 활용하고, 메모리 할당 및 해제 요청을 신속하게 처리해야 한다.

2. 개발 범위 및 내용

A. 개발 범위 및 내용

Free Block들을 관리하는 방법 중 Explicit List 방법을 채택하였다. Free block만을 연결하여, 탐색을 보다 효율적으로 수행하고자 하였다. 또한 'mm_init()' 함수를 이용하여 초기 Heap 설정과 메모리를 초기화 하며, 'mm_malloc()'함수를 통해 메모리 block을 할당하고, 'mm_free()'로 메모리 block을 해제하며, 'mm_realloc()'함수를 통해 메모리 block의 크기를 재조정한다. 만약 Heap 영역이 부족할 경우, 'sbrk()'함수를 통해 Heap 크기를 늘리게 되는데, 'extend_heap()'이 이를 해결한다. 또한 block을 적당히 splitting과 coalescing하게 되며, 이는 'place()', 'coalescing()'이 하게 된다.

B. 개발 방법(새로 구현한 함수 및 매크로, flowchart)

■ **매크로**

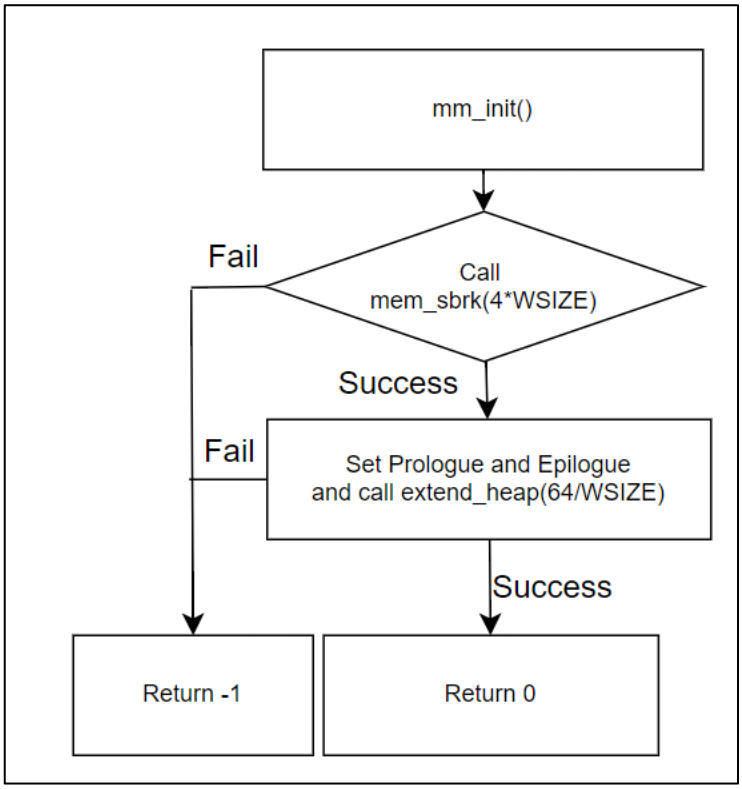
- ◆ ALIGN(size): 주어진 'size'를 8byte 단위로 정렬
- ◆ SIZE_T_SIZE: 'size_t' 타입 크기를 8byte 단위로 정렬한 값
- ◆ SET_PTR(p,ptr): 포인터 'p'에 'ptr' 값을 설정
- ◆ NEXT_PTR(ptr): free block list의 다음 포인터를 반환
- ◆ PREV_PTR(ptr): 자유 블록 리스트의 이전 포인터 반환

■ **함수 구현**

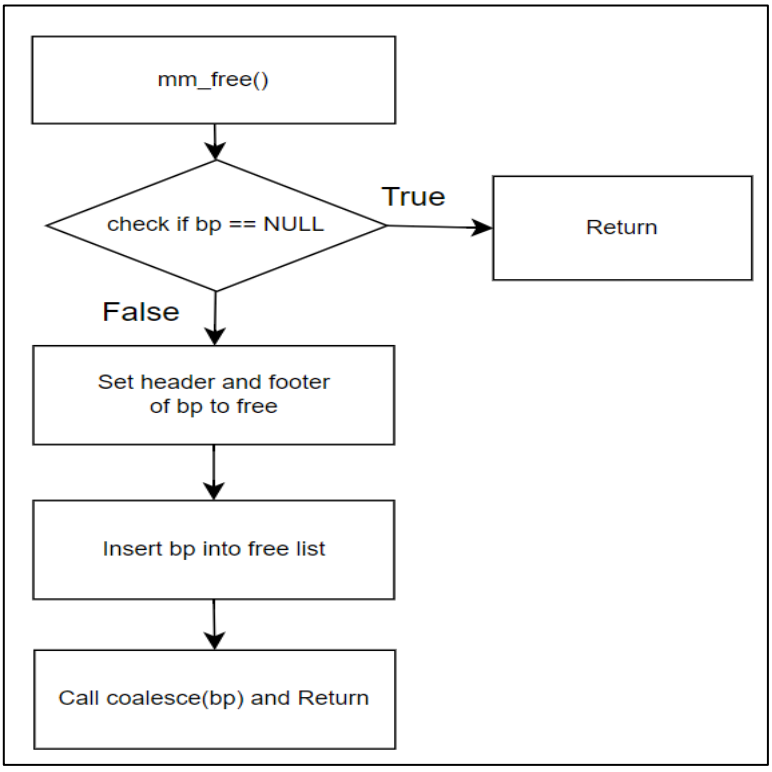
- int mm_init(void): Heap을 초기화하는 함수로, prologue block과 epilogue block을 설정하며, 초기 Heap 확장을 수행한다. 성공 시 0을, 실패 시 -1을 반환한다.
- void *mm_malloc(size_t size): 인자로 넘겨받은 'size' 크기의 memory block을 할당하는 함수이다. Block size를 조정하고 free block list에서 적절한 block을 찾는다. 적절한 block이 없으면 Heap을 확장하여 새로운 block을 할당한다. 필요한 경우 블록을 splitting한다.
- void mm_free(void *bp): 주어진 포인터가 가리키는 memory block을 free하는 함수이다. free된 블록을 free block list에 추가하고 인접한 free block들과 coalescing한다.
- void *mm_realloc(void *bp, size_t size): 기존 memory block의 크기를 변경하는 함수이다. 필요한 경우 새로운 memory block을 할당하고 그 block에 데이터를 복사한다. 기존 block을 free하고 인접 block과 병합하여 utilization을 최대화한다.
- static void *extend_heap(size_t words): Heap 영역을 확장하는 함수이다. 새로운 free block을 초기화하고 free block list에 추가한다. 인접한 free block과 coalescing하여 external fragmentation을 줄인다
- static void insert_node(void* ptr, size_t size): free block을 free block list에 추가하는 함수이다. 이 때 LIFO(Last-In-First-Out) 방법을 채택하여 새로운 free block은 free block list의 맨 앞에 삽입하게끔 하였다.
- static void delete_node(void* ptr): free block을 free block list에 제거하는 함수이다. 이 때 LIFO(Last-In-First-Out) 방법을 채택하여 특정된 free block은 free block list에서 제거한 후, 이전 block 또는 다음 block과 연결한다.
- static void *coalesce(void *bp): 인접한 free block을 병합하는 함수이다. 이전 free block과 다음 free block의 할당 상태를 확인하여 필요시 병합한다.
- static void *place(void *bp, size_t asize): Allocated block을 free block list에서 제거하고, 필요한 경우 블록을 splitting하여 남은 부분을 free block list에 추가하는 함수이다.

- FlowChart (Allocator 주요 함수에 대한 Flowchart)

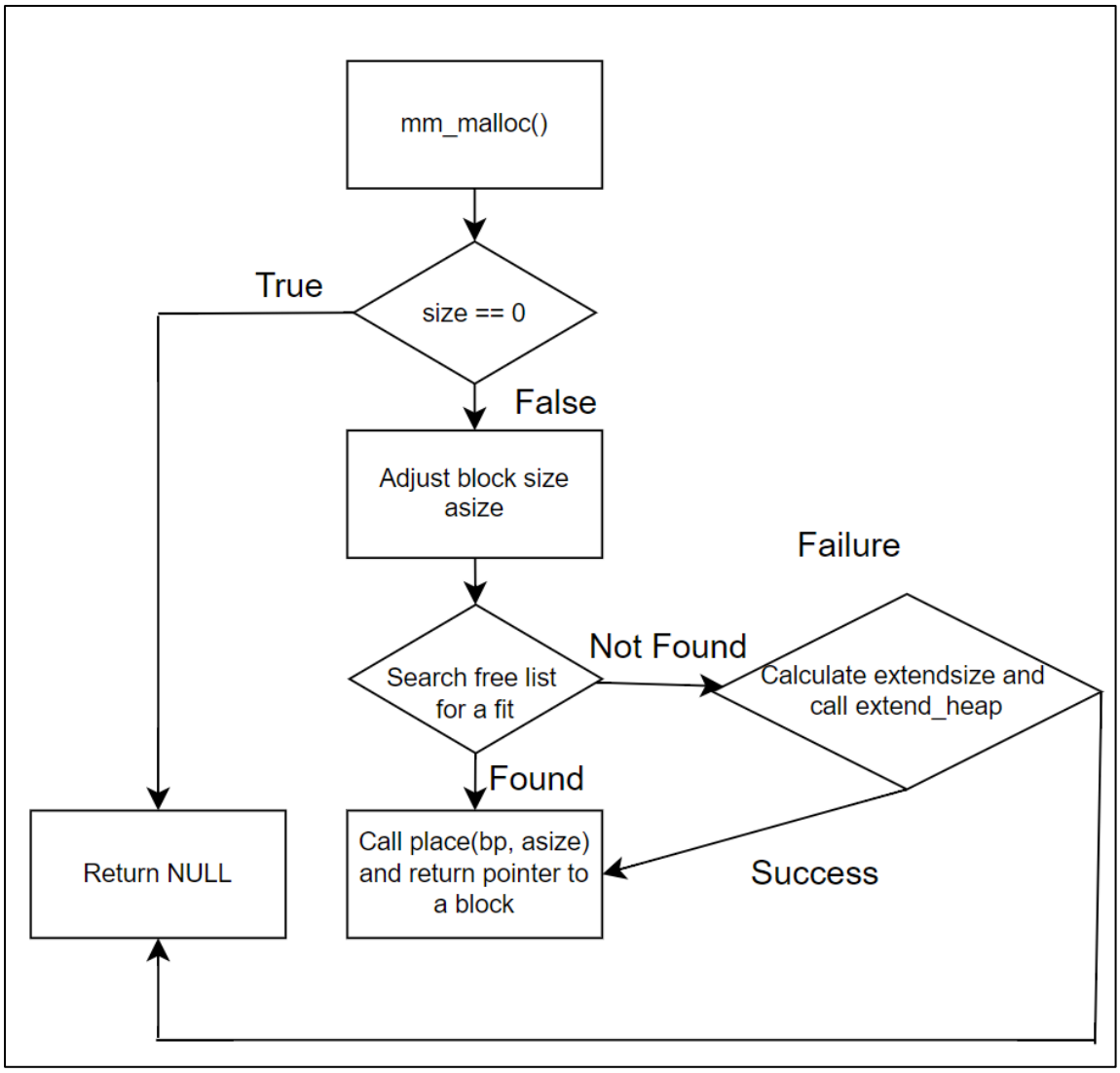
- int mm_init(void)



- void mm_free(void *bp)



- void *mm_malloc(size_t size)



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