PROJECT3 DOCUMENT

20200901 이효주

아래 변수는 코드 전체에서 메모리 할당 및 해제 과정을 관리하고 크기를 계산하며 힙의 상태를 추적하는데 사용하였다.

WSIZE: 바이트 단위의 워드(word) 크기를 나타내고, 4 바이트로 정의하였다.

DSIZE: 바이트 단위의 더블 워드(double word) 크기를 나타내고, 8 바이트로 정의하였다.

CHUNCKSIZE: 힙(heap)을 확장할 때 할당되는 덩어리(chunk)의 크기를 결정하고, 이 코드에서는 (1<<12)로 설정되어 있으며, 4096 바이트이다.

MAX(x, y): 주어진 두 값 x 와 y 중에서 최댓값을 반환하는 매크로이다.

ALIGN(size): 주어진 size 를 ALIGNMENT 의 배수로 올림하는 매크로이며 ALIGNMENT 은 8 바이트로 정의하였다.

SIZE_T_SIZE: size_t 의 크기를 바이트 단위로 나타내며 sizeof(size_t)의 정렬된 크기로 계산된다.

team: 이 코드에 참여한 팀 구성원의 정보를 저장하는 구조체로 학번, 전체 이름 및 이메일 주소를 포함한다.

start_bp: 힙의 시작점을 가리키는 포인터로 mm_init 함수에서 초기화되어 힙의 시작 위치를 추적한다.

last_bp: 힙에서 마지막으로 할당된 블록을 가리키는 포인터로 mm_malloc 함수에서 업데이트되어 마지막으로 할당된 블록을 추적한다.

Approach:

이 함수는 할당할 메모리 블록의 크기를 나타내는 매개변수인 size 를 받는다.

먼저 size 가 0 인지 확인합니다. 0 이라면 메모리를 할당하지 않아야 하므로 함수는 NULL 을 반환한다.

다음으로 함수는 요청된 크기에 기반하여 adjusted_size 를 계산하고, ALIGNMENT(8 로 정의된 값)의 배수로 크기를 올림하여 정렬을 보장한. 또한, 헤더와 푸터의 오버헤드를 고려하기 위해 조정된 크기에 2 * DSIZE(더블 워드 크기)를 더한다.

그런 다음 함수는 **next_fit 함수**를 호출한다. 이 함수는 **next-fit 할당 전략**을 사용하여 적합한 빈 블록을 찾는다. 이 전략은 last_bp(마지막으로 할당된 블록)부터 빈 블록을 순환적으로 탐색합니다. 적합한 빈 블록이 발견되면 next_fit 함수에 의해 반환된다.

적합한 빈 블록을 찾을 수 없는 경우 함수는 adjusted size 와 CHUNCKSIZE 중 더 큰 값을 extended_size 로 결정한다. 이는 힙이 적어도 요청된 크기나 최소 크기로 확장되도록 보장한다.

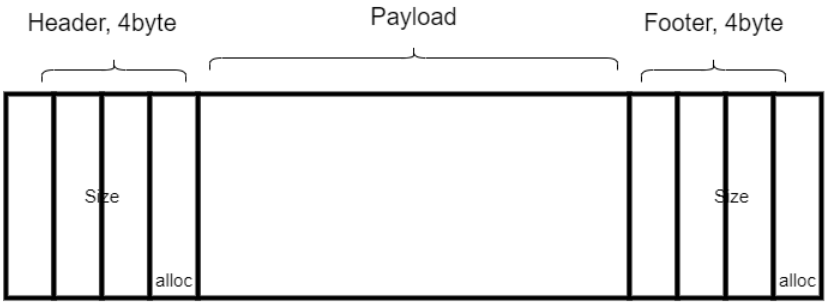
그런 다음 함수는 extend_heap 함수를 호출하여 힙을 주어진 단어 수만큼 확장한다. 이 함수는 mem_sbrk 시스템 호출을 사용하여 운영 체제로부터 추가 메모리를 요청합니다. 요청이 실패한 경우 에러를 나타내는 NULL 이 반환된다.

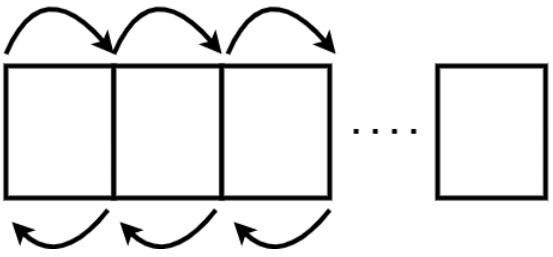
힙 확장이 성공하는 경우 extend_heap 함수는 새로운 블록의 헤더와 푸터를 초기화하고 다음 블록의 헤더를 종료 마커(크기 0, 할당됨)로 설정한다.

mm_malloc 함수는 그런 다음 mark 함수를 호출하여 할당된 블록을 표시한다. mark 함수는 블록 포인터(bp)와 조정된 크기를 매개변수로 받는다. 할당 후 남은 공간이 새로운 빈 블록을 수용할 수 있는 크기(2 * DSIZE 이상)인 경우, 블록은 할당된 블록과 빈 블록으로 분할된다. 할당된 블록은 조정된 크기를 유지하고, 빈 블록은 남은 크기를 가진다.

마지막으로, last_bp 는 새로 할당된 블록으로 업데이트되고, 함수는 할당된 메모리로서의 블록 포인터(bp)를 반환한다.

Approach with Next fit strategy:





Memory allocation Pseudo code:

```
cur_pointer = last_alloc_pointer.next_pointer
while(cur_pointer < end pointer){
  if( size<block_size) return cur_pointer
   cur_pointer=cur_pointer.next_pointer
}
cur_pointer=start_pointer
while (cur_pointer < last_alloc_pointer){
  if(size<block_size)
cur_pointer=cur_pointer.next_pointer
}
```

Memory Free Pseudo code:

```
If (next_block_free and prev_block_free){
 merge(cur,next)
 merge(prev,cur)
change_HEADER_FOOTER(prev)
}
else if (next_block_free and prev_block_alloc){
 merge(next,cur)
  change_HEADER_FOOTER(cur)
}
else if (next_block_alloc and prev_block_free){
merge(prev,cur)
change_HEADER_FOOTER(prev)
```

```
  }
  else
  change_HEADER_FOOTER(cur)
```

Memory Reallocation Pseudo code:

```
lf (new_size<=cur_size)
 do nothing
else{
 change_HEADER_FOOTER(cur)
 mm_malloc(new_size)
}
```