

Operating System Assignment 4 Report

201620898 소프트웨어 강병수

1. What I did

이번 과제는 메모리 관리 함수를 직접 구현해봄으로써 heap 영역에 대한 이해를 높이는 데 그 목적이 있었다. 구현은 heap 영역의 break를 직접 설정하는 sbrk 함수와 이중 연결리스트를 사용하였다. Allocation, deallocation, reallocation, 이렇게 세 가지를 구현하는 것이 과제의 목표였는데, 좀더 효율적이고 가독성 높은 코드 작성을 위해 다음과 같은 8개의 function을 작성하였다.

```
meta *m_merge(meta *cur1, meta *cur2);

meta *m_split(meta *cur, size_t size);

void *m_find(size_t size);

void *m_malloc(size_t size);

void *m_realloc(void *ptr, size_t size);

void m_free(void *ptr);

void *m_travel(int idx);

void print_mem(void);
```

*m_malloc, *m_realloc, m_free의 함수 원형은 과제에서 주어진 그대로 따랐으며, 각각 메타데이터 블록 다음의 실제 메모리 공간의 시작 주소를 반환한다. *m_merge와 *m_split은 각각 free와 realloc 과정을 수행할 때 사용되는 함수들이며, 주어진 블록을 합치거나 나누는 역할을 한다. *m_find는 메모리를 할당할 블록을 탐색하는 함수이다. 주어지는 fit method에 따라 다른 방식으로 탐색하여 메모리 블록의 시작 주소값을 반환한다. *m_travel은 블록 검색을 위한 용도이며 지정한 idx번째 블록에 위치하는 블록을 반환한다. 이는 realloc 함수에서 필요한 *ptr 포인터 탐색에 쓰인다. 마지막으로 print_mem 함수는 최종 결과를

출력하기 위한 함수이다.

입력 부분은 main에서의 argv 인자를 통해서 실행 시에 바로 input 파일을 입력받도록 하였다. Command를 fscanf함수로 입력받은 후에, 새로운 string 입력은 미리 선언된 buf 문자열 배열에 임시로 저장된다. 이후 이 string은 memcpy함수를 통해 *m_malloc 함수로 할당받은 포인터 주소로 복사된다. 탐색 방식(Fit)을 지정하는 변수는 전역변수 extern int형을 사용하였다.

Allocation에 관련된 함수에서는 매크로 함수($\text{align}(n)=(((n+3)/4)*4)$)를 통해서 4byte 단위로 입력값을 align하도록 하였으며, base가 존재하지 않는 경우나 새 블록이 필요할 경우에 sbrk 함수를 사용하여 heap 영역을 추가로 할당하도록 구현하였다.

Merge 함수는 재귀 방식을 사용하여 주어진 블록의 앞과 뒤에 더 이상 merge할 free block이 존재하지 않을 때까지 연쇄적으로 실행되도록 구현하였다. Split 함수는 주어진 size 값만큼 split할 수 없을 경우에는 아무 처리 없이 입력받은 포인터를 그대로 반환하도록 구현하였으며, 가능할 경우에는 주어진 size가 나뉜 블록의 앞 부분이 되도록 구현하였다.

여러 메세지 출력에 관한 부분도 대부분 구현하였으나 예제와 같은 출력 결과를 위해 모두 주석 처리하였다. 주어진 예제에 대한 결과는 모두 잘 나오는 것을 확인하였다.

2. What I learned

먼저, 약했던 포인터 개념을 보강할 수 있는 좋은 기회가 되었다. 메모리 시작 주소를 넘기기 위해 계속해서 meta형인 변수에 1 대신 META_SIZE를 더한다던가, free함수가 영동한 블록을 해제시킨다던가 하는 버그 때문에 처음 코드를 짜는 시간만큼 디버깅에 시간을 소요했다. 과제를 해결하면서 이렇게 구멍난 개념들이 무엇이었는지 알 수 있었고 부족한 개념들을 채울 수 있었다.

다음으로, 메모리 구조에 대한 이해가 확실해졌다. 실제 free함수를 사용했을 때에도 여전히 내용이 메모리에 남아있는 이유나, malloc과 realloc함수에 대한 구분점에 대해서 직관적으로 이해할 수 있는 기회가 되었다.

부수적으로 얻은 점으로 IDE에서 디버깅 기능을 훨씬 더 잘 활용할 수 있게 되었다. 개발환경은 ubuntu에서의 vim과 macOS의 Xcode를 사용하였는데, Xcode에서 디버깅 시에 breakpoint가 걸리는 조건을 설정하거나 직접 메모리 내부를 뜯어볼 수 있다는 것도 이번에 처음 알게 되었다.

3. Feedback

Test case가 조금 더 많았으면 좋았을 것 같다. 예제에서의 test case는 first fit에 대한 길지 않은 예제 위주라서 테스트를 위해 예제를 생각하는 데에도 시간이 많이 소요되었다.