Mybuddy Report

201720720 조민재

이번 과제는 Buddy memory allocator simulator를 구현하는 것이었다.

먼저 init_buddy를 구현하기 위해 chunks의 struct에 각 order마다 free list를 TAILQ를 사용해서 만들어주었다. Init_buddy에서 TAILQ를 초기화해주었고, nr_page_order를 MAX_ORDER만큼의 chunk로 잘라 초기화해 주어야 하므로 그 차이의 개수만큼 chunk를 만들어 할당해주고, order가 MAX_ORDER인 free list에 넣어주었다. 넣어줄 때 시작 주소도 각각 초기화하였다.

다음으로 fini 함수에 할당했던 chunk를 해제해주는 코드를 작성했다.

다음으로 free list의 chunk를 할당해주는 alloc함수를 작성하였다. 일단 할당을 요청했을 때 해당 order의 free list에 chunk가 있으면 그대로 할당해주고 끝내면 된다. 하지만 만약 그게 아니라면, 상위 order의 chunk를 쪼개서 할당해주어야 하므로, chunk를 가진 최소 상위 order를 order를 증가하면서 찾고, 찾았으면 해당 order의 free list의 가장 앞에 있는 chunk를 꺼내 반으로 쪼개 왼쪽 chunk를 할당시켜 주고, 나머지 오른쪽은 free list에 넣어 주었다. 내가 원하는 크기가 될 때까지 잘라서 할당해 주어야 하므로, 반복문을 돌려 할당해줄 chunk와 free list에 넣어줄 chunk를 일단 둘 다 계속 하위 order의 free list에 넣어 준 다음, 원하는 크기가 되었으면 반복문을 끝내고 free list에서 가장 앞의 chunk를 할당해준다. 만약 상위 order를 다 찾아도 chunk가 없으면 에러 메시지를 출력하고 끝나게 했다.

다음으로 free함수는 더 복잡했는데, 먼저 free 해주려는 페이지의 order의 free list에 chunk가 없으면 그냥 반환해준다. 만약 그렇지 않으면, 페이지가 left인지 right인지 확인한다. 시작 주소와 chunk사이즈를 통해 left인 것을 확인했으면, 해당 free list의 chunk를 foreach로 반복문을 돌려서 오른쪽 버디가 있는지 찾는다. 찾았으면 버디를 free list에서 제거한 후에 order를 하나 올려 주고, 올렸을 때도 버디가 있으면 찾아서 merge해야 하므로 상위 order에 buddy가 없을 때까지 buddy를 찾고, 더 이상 없으면 merge를 하고 끝낸다. Free 해주려는 페이지가 right인 경우에도 비슷하게 해 주는데, 여기서 다른 점은 merge 한 chunk의 시작 주소는 왼쪽 chunk의 주소와 같으므로 buddy로 설정해야 한다는 것이었다. Right인 경우 또한 flag를 세워서 버디가 있는지 판단하고, 버디를 더 이상 찾지 못할 때까지 상위 order로 올려 보낸 후에 마지막에 merge하여 해당 order의 free list에 넣어주면 된다.

그리고 free page를 프린트 하는 함수는 그냥 foreach로 반복문을 돌려 모든 order의 free list에 있는 chunk의 시작 주소를 프린트해주었다.

Get unusable index는 전체 free page 중에 해당 order보다 작은 order의 page의 비율을 구하면 되는 것이었다. 이것 또한 FOREACH를 사용해 각각 모두 더한 후에 나누어 주었다.

이번 과제에서 힘들었던 점은 메모리 릭을 체크하는 것이었는데, 아무래도 할당 문제가 항상 있

다보니까 어디서 문제가 일어난 것이고 어떻게 고쳐줘야 할지 생각하는 것이 어려웠던 것 같다. 그리고 merge한다는 개념이 처음에는 정말로 내가 공간을 합쳐야 한다고 생각을 해서, 이것을 어떻게 코드로 구현할지 막막했다. 실제로는 시작 주소와 order만 잘 설정해 주면 된다는 것을 깨닫고는 한결 더 쉽게 생각할 수 있었다. 그래서 실제로 buddy system allocator가 어떻게 동작하는지를 simulation으로나마 이해하고 배울 수 있게 되었다.

지금까지의 과제보다는 생각하고 구현하기 쉬웠던 것 같고, 딱히 운영체제의 시스템콜과 같은 것이 필요한 게 아니라 알고리즘 문제에 가까운 것 같아서 생각만 잘 하면 그렇게 어렵지 않게 짤 수 있었던 것 같다. 테스트하는 코드도 잘 되어 있어서 테스트하기도 편했다. 지금까지 과제가 너무 어려웠어서 오히려 쉽게 느껴졌던 것 같다.