운영체제(가) 설계과제 5번

숭실대학교 컴퓨터학부

20182612 박민호

**<개요>**

>> 1번

- alloc.c를 직접 구현

- 주어진 alloc.h에서 선언된 함수들을 구현해야 함.

- 메모리 관리자는 OS에서 mmap을 통해서 4kb를 할당. 할당된 메모리를 요청에 따라 관리를 해 주어야함.

- 할당된 메모리를 관리하는 방법은 배열을 이용한 연결리스트를 사용했습니다.

- 구현해야 하는 기본 함수

1. init\_alloc() : mmap을 통해 os에서 4kb 페이지를 할당하고, 메모리 관리자 초기화. 사용자가 메모리 관리자에게 메모리를 요청하기 전에 호출, 성공 시 0, 에러 시 0이 아닌 수 리턴.
2. cleanup() : 메모리 관리자의 상태를 정리, mmap으로 할당받은 페이지를 os 커널에게 다시 반환. 성공 시 0, 에러 시 0이 아닌 수 리턴
3. alloc(int) : int 만큼의 크기를 가진 버퍼에 대한 포인터를 리턴, 에러 시 null 리턴
4. dealloc(char \*) : 해당 주소에 할당된 메모리 chunk를 해제

- 메모리 관리자는 5가지의 기능을 제공

1. 헤더 또는 메타 데이터 정보를 할당한 페이지에 저장하면 안됨 - 할당 크기를 추적하는데 필요한 모든 메타데이터는 코드에 정의된 데이터 구조 내에 있어야 하며, 매핑 된 4kb 페이지 자체에 포함되지 않아야 함.
2. 이중 할당 금지.
3. chunk의 재사용이 가능해야 함, 메모리 관리를 위해 chunk를 더 작은 chunk로 분할 할 수 있어야 함.
4. free한 chunk가 인접할 경우 두 chunk를 합쳐 메모리 할당에 사용할 수 있어야 함.
5. free된 chunk가 여기저기 흩어져 있을 경우, 빈 chunk를 잘 리턴할 수 있는 메모리 할당방식 고려(최적 적합, 최초 적합, 최악 적합 등등)을 선택하여 할당

- test\_alloc.c를 사용하여 테스트

>> 2번

- 프로세스가 요청 시에만 memory mapped 페이지 할당

- 메모리 관리자에 대한 기본적인 구현 방식은 1과 같이 했습니다.

- 기본 함수

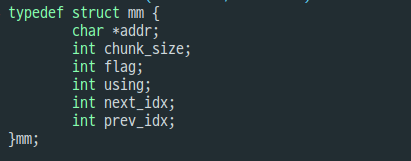
1. init\_alloc() : 메모리 관리자를 초기화, 필요한 데이터 구조를 초기화 할 수 있음. 그러나 요청 시에만 메모리를 할당 해야 하기 때문에 여기서는 메모리 매핑하면 안됨.
2. cleanup() : OS커널로 매핑된 메모리를 반환하지 않는다고 가정하기 때문에 페이지를 해제할 필요 없음.
3. alloc(int) : int만큼의 크기를 가진 버퍼에 대한 포인터를 리턴  
   - 만약 해당 페이지에 적당한 chunk가 없을 경우 새로운 4kb 페이지를 요청  
   - 4페이지 즉 총 16kb를 초과하여 할당하지 않음.  
   - 에러 시 NULL 리턴
4. dealloc(char \*) : 해당 주소에 위치한 chunk를 해제.

- 메모리 관리자의 기능은 과제 1과 같음.

- test\_ealloc.c를 사용하여 테스트

**<구현 내용>**

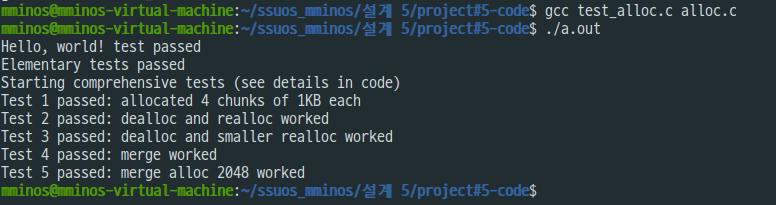
- 1, 2 공통적으로 chunk들을 관리하는 구조체 배열을 생성한 후, 연결리스트를 관리하는 방식과 유사한 형태로 구현했습니다.



- addr은 해당 chunk의 주소, chunk는 할당한 chunk의 크기, flag는 해당 chunk의 사용여부, using은 mm구조체(chunk정보 구조체)의 사용여부, next\_idx와 prev\_idx는 연결리스트와 유사한 동작을 위한 mm구조체 배열의 인덱스 값입니다.

**<실행 화면>**

>> 1번



>> 2번

