과제 #4 : SSU Memory Allocation and Multi-level File System

○ 과제 목표

- xv6 메모리 및 파일시스템 구조 이해 및 응용

○ 기본 지식

- xv6의 메모리 관리 이해
  - ✓ vm.c, trap.c, proc.c, defs.h, mmu.h 등에서 동작하는 메모리 관리 방법(가상 메모리, 동적 메모리, PTE 등) 분석
- xv6의 파일시스템 이해
  - ✓ fs.c, fs.h, file.c, file.h, defs.h, params.h 등에서 동작하는 파일시스템 구조 분석(read, write, close, link 등)

○ 과제 내용

1. 가상 메모리 할당을 위한 ssualloc() 시스템 콜 구현 및 메모리 정보를 확인하기 위한 getvp(), getpp() 시스템 콜 구현

✓ ssualloc()

- ssualloc() 시스템 콜은 하나의 매개변수를 입력받음
  - ☞ 매개변수는 할당할 가상 메모리의 크기
  - ☞ 매개변수는 양수이며, 페이지 크기의 배수임
  - ☞ 유효하지 않은 값을 입력받았을 때, -1을 반환함
  - ☞ 유효한 값을 입력받고, 가상 메모리를 성공적으로 할당했을 때 할당한 메모리의 주소를 반환함
- ssualloc() 시스템 콜은 가상 메모리를 할당하지만 물리 메모리를 할당하지 않음
  - ☞ 할당된 가상 메모리에 사용자가 접근할 때 해당 가상 메모리에 상응하는 물리 메모리가 존재하지 않을 경우 특정 trap이 발생하며 이때 물리 메모리를 페이지 단위로 할당하도록 xv6를 수정
  - ☞ 또한 ssualloc()을 통해 하나 이상의 가상 메모리 페이지가 할당된 경우 접근된 페이지에 대해서만 물리 메모리 페이지를 할당해야 함 (가상 메모리 페이지 0, 1이 있고, 1에 접근 했을 때, 1에 대한 물리 메모리 페이지만을 생성)
  - ☞ Hint. xv6는 현재 위에 대한 trap을 처리하지 않으며, 이를 처리하도록 수정하여야 함
  - ☞ 해당 trap에 대한 질문은 받지 않으며, trap의 종류와 처리한 방법에 대한 설명을 보고서에 포함
- 원활한 과제 진행을 위해 ssualloc_test 프로그램 제공 예정 (수정 불가)
  - ☞ 제공하는 ssualloc_test를 실행하기 위해 Makefile을 수정할 것

✓ getvp(), getpp()

- ssualloc_test 프로그램의 실행을 위해 getvp(), getpp() 시스템 콜을 구현해야 함
- getvp(), getpp() 시스템 콜은 매개변수를 입력받지 않음
- getvp(), getpp()는 각각 호출한 프로세스의 가상 메모리 페이지, 물리 메모리 페이지 개수를 반환함
- getvp(), getpp()를 사용한 ssualloc_test의 실행 결과는 (예시 1)과 같음

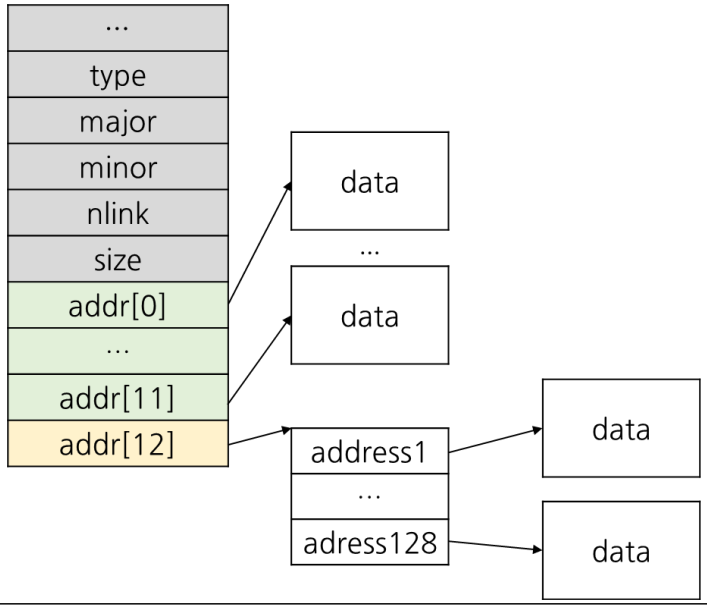
(예시 1). ssualloc_test 실행 결과

```
$ ssualloc_test
Start: memory usages: virtual pages: 3, physical pages: 3
ssualloc() usage: argument wrong...
ssualloc() usage: argument wrong...
After allocate one virtual page: virtual pages: 4, physical pages: 3
After access one virtual page: virtual pages: 4, physical pages: 4
After allocate three virtual pages: virtual pages: 7, physical pages: 4
After access of first virtual page: virtual pages: 7, physical pages: 5
After access of third virtual page: virtual pages: 7, physical pages: 6
After access of second virtual page: virtual pages: 7, physical pages: 7
```

○ multi-level 파일시스템 구현

- [그림 1]은 xv6 파일시스템이 사용하는 inode 구조체의 일부를 나타냄.

[그림 1] xv6 파일시스템



☞ addrs 배열이 가리키는 블록 번호는 디스크에 파일이 실제로 저장된 데이터 블록의 번호를 의미함

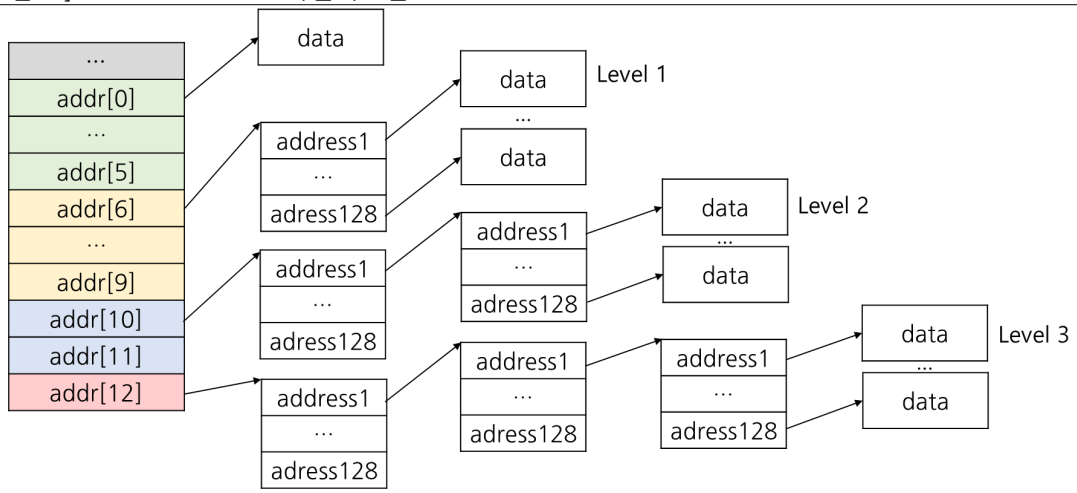
☞ 기존 xv6의 파일시스템 구조는 12개의 direct 블록, 1개의 indirect 블록으로 구성되어있음.

☞ direct 블록인 addrs[0] ~ addrs[11]에는 파일의 내용이 저장된 디스크의 데이터 블록을 가리키는 포인터 변수가 저장됨

☞ indirect 블록인 addrs[12]는 데이터 블록을 직접 가리키는 것이 아닌, 또 다른 인덱스 블록(데이터 블록 테이블)을 가리키며 인덱스 블록 내부에는 실제 데이터 블록을 가리키는 포인터 변수들이 저장됨

- xv6는 하나의 파일에 140개(12+128)의 데이터 블록을 저장 공간으로 사용할 수 있으며, 이에 따라 한 파일의 크기는 140 * 512B(Bytes) = 71,680B, 약 70KB임
- 본 과제에서는 해당 구조를 수정하여 6개(addrs[0] ~ addrs[5])의 direct 블록, 4개(addrs[6] ~ addrs[9])의 indirect 블록, 2개(addrs[10] ~ addrs[11])의 2-level indirect 블록, 1개(addrs[12])의 3-level indirect 블록을 가지는 파일시스템을 구현하며 대략적인 구조는 [그림 2]과 같음

[그림 2] xv6 multi-level 파일시스템



☞ 따라서 본 과제에서 구현하는 파일시스템은 2,130,438(6+128*4+128*128*2+128*128*128) 개의 데이터 블록을 저장 공간으로 사용할 수 있으며 약 1GB의 파일 크기를 지원함

- 원활한 과제 진행을 위해 ssufs_test 프로그램 제공 예정 (수정 불가)

☞ 제공하는 ssufs_test를 실행하기 위해 Makefile을 수정할 것

- 대용량 파일 생성을 위해 params.h의 FSSIZE를 변경해야함 (1000 -> 2500000)
- multi-level 파일시스템을 구현한 후, ssufs_test의 실행 결과는 (예시 2)와 같음

(예시 2). ssufs_test 실행 결과

```
$ ssufs_test
### test1 start
create and write 5 blocks...    ok
close file descriptor...        ok
open and read file...           ok
unlink file1...                 ok
open file1 again...             failed
### test1 passed...

### test2 start
create and write 500 blocks...  ok
close file descriptor...        ok
open and read file...           ok
unlink file2...                 ok
open file2 again...             failed
### test2 passed...

### test3 start
create and write 5000 blocks... ok
close file descriptor...        ok
open and read file...           ok
unlink file3...                 ok
open file3 again...             failed
### test3 passed...

### test4 start
```

```
create and write 50000 blocks...        ok
close file descriptor...        ok
open and read file...           ok
unlink file4...                 ok
open file4 again...             failed
### test4 passed...
```

○ 과제 제출 마감

- 2023년 12월 4일 (월) 23시 59분 59초까지 구글클래스룸으로 제출
- 보고서 (hwp, doc, docx 등으로 작성
  - ✓ 총 2개의 테스트 프로그램이 수행된 결과
  - ✓ 1. 의 trap의 종류와 처리 과정
- xv6에서 변경한 소스코드(Makefile 포함 필수)
  - ✓ 변경의 기준은 xv6-public 원본이며, 첨부한 소스코드를 복사하여 채점함.
  - ✓ 기존 설계 1~3에서 수정한 Makefile, 소스코드 등으로 인한 오류 발생 시 예외 없이 0점 처리
- 마감시간 이후 24시간까지 지연 제출 가능. 그 이후 제출은 0점 처리. 설계과제 마감시간 이후 지연 제출은 30% 감점.

○ 필수 구현

- 1, 2

○ 배점 기준 (부분점수 없음)

- 1 : 50점
- 2 : 50점