**Pintos Project 5: Filesystem**

담당 교수 :

조 / 조원 :20191613 윤상현

개발 기간 :

1. **개발 목표**

* **해당 프로젝트에서 구현할 내용을 간략히 서술.**
* Extensible file & file growth: 외부 단편화로 인해, 공간이 충분함에도, 파일을 할당하지 못하는 경우를 방지, 파일 크기늘리기 구현
* Subdirectory: 단일 디렉토리가 아니라, 다중 계층의 디렉토리를 가능하게 함

Buffer cache: 자주 쓰는 데이터를 버퍼 캐시에 저장해놓아서 속도 증가

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Extensible file & file growth

기존의 linear하게 할당할 때에는 외부단편화로 인하여, 실제 공간이 충분하여도 할당이 불가능한 경우가 있었음. 그러한 문제 해결

그리고 파일의 고정된 크기를 가져서, 파일의 크기를 늘릴 수 없었던 문제 해결

1. Subdirectory

단일 루트 디렉터리로 구성되어 있으나, 구현할시 파일 뿐아니라 디렉토리들 또한 생성할 수 있음.

1. Buffer cache  
   자주 쓰는 데이터를 디스크에서 항상 읽어올 필요없이, 작은 공간의 메모리에 저장하여 속도를 증가 시킬 수 있음.

**구현한 경우만 작성**

* 1. **개발 내용**
* **아래 항목의 내용만 서술 (기타 내용은 서술하지 않아도 됨.)**
* Extensible file & file growth
  + Index structure와 management에 대해서 기술

Ppt에 나온대로, direct, 1lv indirect, 2lv indirect 각각으로 구현함.

Direct 123 block, lv1 1 block, lv2 1block으로, 총 123+128+128\*128

Block을 다룰 수 있음. 이렇게 할 경우 직선적으로 할당할 필요가 없기때문에, 외부 단편화와 같이 file growth도 해결 됨.

* Subdirectory
  + Directory entry 관리 방법

기존의 디렉토리가 블록에 디렉토리 엔트리를 통해 관리 하던 것에

그냥 파일과 inode에 디렉토리 인지 여부를 저장하는 변수를 추가하여

관리함. 파일의 이름은 15글자 미만 그대로 이나, 따로 전체 경로를 parsing하는 함수를 만들어서 전체 경로가 15글자 이상인 경우를 관리해줌.

Mkdir, isdir, chdir, readdir등의 시스템콜과 기존의 remove open read write create등의 시스템콜을 수정하여 세부적인 기능을 구현합니다.

* Buffer cache
  + Buffer cache eviction 방식

Second chance 알고리즘을 사용하여, evict될 때 각각의 후보들에게 second chance를 주고, 그 후보들이 다시 참조되면 또 기회를 줌.

* + Buffer cache flush 방식

해당 버퍼 캐시가 evicte될 떄나, 파일 시스템이 unmount될떄나 핀토스가 종종료될 때 dirty한 경우, 항상 flush해주는 방식을 사용하였습니다. 후자의 두 경우에는 전체 버퍼에 대해 전부 검사해서 flush했습니다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* **II. A.의 개발 범위를 포함하여 구현 내용에 대한 일정 작성**

먼저 inode의 구조를 변경하여 extensible file & file growth를 구현합니다.

그 후, buffer cache를 구현하여 먼저 구현한 기능들이 똑같이 작동 되는지를 보면서 구현. 마지막으로 subdirectory를 구현합니다.

* 1. **개발 방법**
* **II. B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**

Userprog/syscall.c process.c t

Isdir, mkdir, chdir, readdir 의 시스템 콜 추가

remove create open close등의 시스템콜이 디렉토리를 다룰 수 있게 변경

Threads/thread.h

상대경로의 루트 디렉토리를 뜻하는 nowd를 각 스레드마다 추가.

Inode.c

Strct inode\_disk 자료구조에 isdir 변수 추가, 기존의 start 블록 1개에서, 123개 직접, 1개 lv1, 1개 lv2의 간접 블록으로 변경.

위의 구조 변경에 따라서, alloc\_inode, free\_inode 함수 추가, 기존의 free\_map\_allocate, release함수를 대체

원래는 start에서 시작하는 직선 할당이었지만, 이제 링크드 리스트와 같은 형태로 할당하므로, n번째의 블록을 추적해줄 byte\_to\_sector 수정, file의 길이를 넘어서는 작성을 시도할 경우, alloc\_inode로 길이를 늘리도록 inode\_write\_At 수정.

Filesys.c

filesys\_creat가 디렉토리 여부를 확인할 수 있도록 입력변수에 bool isdir 추가 하고, 디렉토리를 다룰 수 있게 수정, 이름 대신 경로를 받아도 다룰 수 있게 수정.

filesys\_open이 디렉토리를 다룰 수 있게 수정, 이름 대신 경로를 다룰 수 있게 수정

filesys\_remove가 디렉토리를 다룰 수 잇게 수정, 경로를 다룰 수 있게 수정1

Directory.c

dir\_lookup에서, “.”과 “..”또한 다룰 수 있게 수정

dir\_add에서 위에서 추가한 “..”에 대한 부모 디렉터리에 대한 정보를 추가하게 수정

dir\_remove : 빈디렉터리만 삭제 가능하게 수정

dir\_readdir: .과 ..을 제외하고 읽을 수 있게 수정

dir\_empty: 삭제할 때 디렉터리가 비어있는 지 확인

dir\_chdir: 시스템 콜. 현재 상대 루트 디렉터리를 변경

dir\_split: 경로를 받아서, 디렉토리와 파일 두개로 나눠줌.

dir\_open\_by\_path: 위에서 받은 디렉토리경로를 따라가면서 가장 바깥쪽의 디렉토리를 열어줌.

Cache.c

void buffer\_init: 버퍼를 초기화하고, lock도 초기화.

void buffer\_close: 파일 시스템을 종료할때에, 버퍼의 데이터를 다시 디스크로 전부 다시써주고 초기화.

void buffer\_read : 버퍼에 디스크의 내용을 저장.

void buffer\_write 버퍼의 내용을 디스크에 씀.

Buffer\_evict: second chance 알고리즘에 따라서 버퍼를 evict.

Buffer\_lookup: buffer에 해당 섹터의 내용이 있는지를 확인.

Buffer\_flush: 해당 엔트리가 dirty인 경우, 디스크에 rewrite

1. **연구 결과**
   1. **Flow Chart**

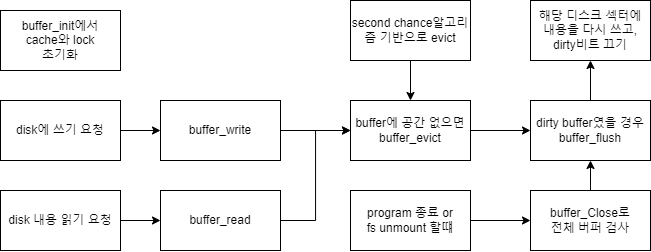
* **II. B. 개발 내용에 대한 Flow Chart를 작성**
* Extensible file & file growth



* Subdirectory



* Buffer cache

  
**구현한 경우만 작성**

* 1. **제작 내용**
* **II. B. 개발 내용의 실질적인 구현에 대해 코드 관점에서 작성.**
* **구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명.**
* **개발상 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결책에 대해 설명.**
* Extensible file & file growth

Inode.c를 수정합니다.

Inode\_disk 에 isdir 변수 1개, 123개의 lv0섹터, 1개의 lv1섹터, 1개의 lv2섹터를 추가하고, unused 섹터를 삭제햇습니다. Lv1은 블록의 주소 128개를 가지고 잇고, lv2는 128개의 주소를 가진 블록을 128개 가리키고 있습니다.

Alloc\_inode로 할당할 블록의 수만큼 lv0,lv1,lv2를 거치면서 할당하고,

Free\_inode도 똑같이 거치면서 해제해줍니다.

똑같이, byte\_to\_sector도 차례로 거치면서, n번째 바이트를 가리키는 블록이 어디인지를 반환해줍니다.

이것으로 externeal fragment에서 자유로운 extensible file이 구현이 되었고, file\_growth 는 inode\_write\_at에서, 파일의 기존 길이보다 긴 위치에 작성을 하려고 할 경우, alloc\_inode로 새로 할당을 해줌으로써 파일의 길이를 늘리도록 구현하였습니다.

* Subdirectory

서브 디렉토리는 기존의 파일에 dir, inode\_disk에 isdir라는 변수를 추가하여 디렉토리인지를 보이는 방식으로 구현하였습니다, thread에 현재 상대 디렉토리의 루 트를 뜻하는 nowd를 추가하였습니다.

Isdir: inode\_disk 의 isdir 를 리턴

Mkdir: filesys\_create(~, isdir) (추가함)를 호출

Chdir: thread의 now를 변경.

Readdir: 이미 잇던 dir\_readdir호출

Inumber: fd를 file찾고, file->inode->sector를 리턴

Dir\_split: 경로에서 맨 뒤의 원소를 파일, 그 앞의 것을 경로로 분리해줍니다.

Open\_by\_path: 받은 경로가 뜻하는 디렉토리를 열어서 반환해줍니다.

🡺둘이 합쳐서 경로가 뜻하는 디렉터리를 열고, 거기서 맨 뒤의 디렉토리나 파일을 찾아서 아래의 행동들을 할 수 있게 해줍니다.

시스템 콜 Create/open/remove가 각각 filesys\_create/open/remove를 각각 호출합니다.

Filesys~ 함수들은 기존에 name 변수를 받아서 기능하였는데, 위에서 추가한 splitpath, open\_by\_path를 통해서 디렉터리와 파일을 나누어서 열어야 할 파일이 디렉터리인 경우와 아닌 경우를 각각 처리하도록 하였습니다.

나누어서 file == null(~/~/~/인 경우에만, 경로가 null인 경우는 원래대로 false)이거나 file->isdir 는 참인 경우에는 디렉터리를 처리해야 합니다. (분리한 결과 파일과 디렉토리가 모두 존재하거나, 전체 경로를 처리한 결과 전부 디렉터리인 경우)

Create: dir\_add호출: “..”에 대한 정보를 추가합니다. (부모 디렉터리 가리킴)

Open: file\_open 호출: 파일을 열고, file->dir에 디렉터리 저장

Remove: dir\_remove호출: dir이 비어있으면 삭제(root, nowd인 경우 미삭제)

* Buffer cache

Filesys\_init에서 초기화할 때 같이 buffer\_init으로 버퍼 초기화하고 lock 초기화합니다. 그 후, 기존의 block\_read를 buffer\_read가 block\_write를 buffer\_write가 대체합니다.

버퍼가 가득 찬 시점에서는 second chance 알고리즘을 기반으로 evict를 진행합니다. (evict시 dirty버퍼는 디스크에 다시 써줘야 함) 각각의 후보들에게 한번의 기회를 주고, 다시 참조될 경우 다시 회복시켜 주는 방식입니다.

buffer에서 buffer\_llookup으로 찾고, 버퍼에 내용이 없으면 block\_read로 읽어옵니다. 이때 공간이 없으면 buffer\_evict합니다. 이떄, buffer\_flush(디스크로 다시쓰는 함수)가 호출됩니다. Buffer\_close함수가 filesys\_done함수에서 파일 시스템을 종료할 때에 같이 호출됩니다. 버퍼의 모든 내용을 디스크에 다시 쓰는 기능을 합니다.

**구현한 경우만 작성**

* 1. **시험 및 평가 내용**
* **Src/filesys make grade 수행결과를 캡처 하여 첨부.**