รายงาน

จัดทำโดย

นายกร	อิสระนิมิตร	รหัสนักศึกษา 57010013
นายกฤตชัย	ชัยสุนทรโยธิน	รหัสนักศึกษา 57010024
นางสาวฐิติรัตน์	โชคสวัสดิ์	รหัสนักศึกษา 57010354
นายติณณ์	จิตต์จนะ	รหัสนักศึกษา 57010494
นางสาวทิพากร	ธนวรรณรัชต์	รหัสนักศึกษา 57010524
นายนลธวัช	หนูมอ	รหัสนักศึกษา 57010669
นางสาวแพรพลอย	รัตนากร	รหัสนักศึกษา 57010939
นายศรายุธ	ต่อธิติธรรม	รหัสนักศึกษา 57011224
นาย สรวิศ	เดชเสน	รหัสนักศึกษา 57011314

เสนอ

ดร.อักฤทธิ์ สังข์เพ็ชร

ดร.อรทัย สังข์เพ็ชร

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชา 01076259 Operating Systems

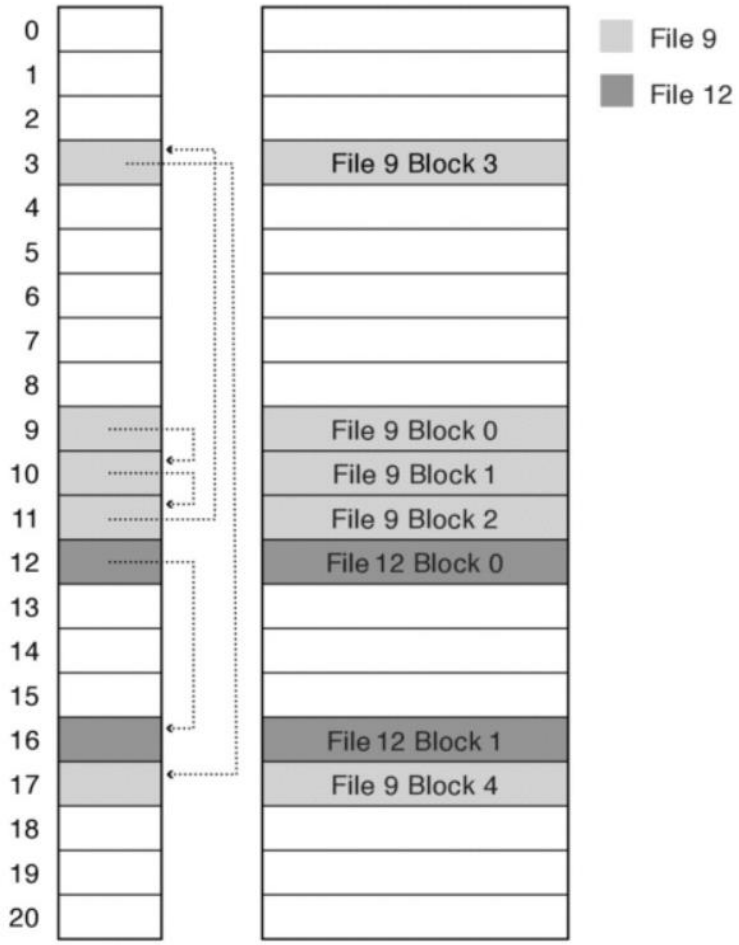
ภาคเรียนที่ 2 ปีการศึกษา 2559

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การออกแบบ

- แนวทางการออกแบบ File System
    - Index structure : ใช้ Linked list ในการ map ข้อมูลโดยจะมี map table ที่เชื่อมโยงกับ data blocks ในการใช้งานคือ เมื่อเรียกไฟล์หนึ่งขึ้นมา จะได้ index ของส่วนแรกของไฟล์ index นั้นจะบอกว่าส่วน data จริง ๆนั้นอยู่ที่ไหน และบอกว่าส่วนต่อไปของไฟล์นั้นอยู่ที่ไหนโดยการชี้ไปที่ index ส่วนต่อไป
    - Index structure granularity : block :
    - Free space allocation : ใช้ array int ในการระบุพื้นที่ที่ยังว่างอยู่ใน data block ถ้า data block ที่เชื่อมโยงนั้นยังว่างอยู่ array ช่องนั้นจะเป็น 0 แต่ถ้า data block ที่เชื่อมโยงอยู่นั้นมีข้อมูลอยู่ (ข้อมูลที่ไม่เป็นขยะ) ก็จะไม่เป็นค่า 0 และยังสามารถระบุได้ว่าเป็น end of file หรือไม่เพื่อใช้หยุดการเข้าถึงไฟล์เมื่อมาพบส่วนของไฟล์ส่วนสุดท้ายแล้ว
    - การหาพื้นที่ว่างอยู่เพื่อจะเขียนไฟล์ใหม่เข้ามานั้น จะเริ่มจากการตรวจดู index ที่ชี้ data block อยู่ที่ละ index แบบ sequence เมื่อพบ index แรกที่ระบุว่าว่างอยู่ ก็ใช้ index และ data block ชุดนั้นเลย หากเป็นไฟล์ข้อมูลที่ใหญ่เกิน data block ที่รองรับ ก็ split ไฟล์ออกแล้วไปตรวจหา data block ต่อไปที่ยังว่างอยู่ โดย index แรกนั้นจะระบุว่า index ชุดต่อไปอยู่ที่ address ใด ทำต่อไปอย่างนี้จนกว่าจะสามารถเก็บไฟล์ได้สมบูรณ์
- รูปแบบและโครงสร้างข้อมูลที่เกี่ยวข้องกับการจัดเก็บข้อมูลบน Disk / Image File ทั้งส่วนที่เป็นข้อมูลและ Metadata



เช่น echo “abc” > test.txt ทำงานดังนี้

หากไฟล์มีขนาดเล็กกว่าหรือเท่ากับ data block จะใส่ข้อมูลลงไปเลย หากข้อมูลมีขนาดใหญ่กว่าจะแบ่งข้อมูลออกเป็นชิ้นเล็ก ๆแล้วใส่ลงไปใน data block แต่ละช่อง ซึ่งมี pointer ชี้บอกตำแหน่งที่เก็บข้อมูลถัดไป

- กระบวนการสร้างเวอร์ชั่นใหม่ของแต่ละไฟล์

ใน directory หนึ่งจะมีไฟล์ archive ทุก ๆ 1 directory เพื่อเก็บ version ของไฟล์ที่แก้ไข โดยเอา file ก่อนจะ modify ไปบันทึกไว้ เช่น test-1.txt 20.00 หมายความว่าเป็นไฟล์ version ที่ 1 โดยจะบันทึกเลข version และเวลาที่บันทึกไฟล์ไว้ ซึ่งหากมี user มาแก้ไขเปลี่ยนแปลงเกินเวลาที่กำหนด ไฟล์ archive จะบันทึก version ใหม่เป็น text-2.txt 20.15 (สมมติให้เวลาที่กำหนดไว้ไม่เกิน 10 นาที และเข้าไปแก้ไขตอน 20.15) แต่หากแก้ไขในเวลาที่กำหนด จะเซฟทับไฟล์ล่าสุด

หลักการทำงานของ vCowFS

คือ การสร้าง file เปล่าขึ้นมาหนึ่ง file แล้ว mount ไปที่ ๆต้องการ เปรียบเสมือนใส่ซีดีเปล่าซึ่งสามารถเขียนข้อมูลลงไปได้

-- ใส่โค้ด อาจจะอธิบายการทำงานของโค้ดด้วย --

อธิบายฟังก์ชัน

```
static int myfs_getattr(const char *path, struct stat *stbuf, struct fuse_file_info
*fi)
```

เรียกดูสถานะของ File

```
static int myfs_mknod(const char *path, mode_t mode, dev_t rdev)
```

สร้าง File ขึ้นมาใหม่

```
static int myfs_mkdir(const char *path, mode_t mode)
```

สร้าง directory ขึ้นมาใหม่

```
static int myfs_unlink(const char *path)
```

ลบ File

```
static int myfs_rmdir(const char *path)
```

ลบ Directory

```
static int myfs_rename(const char *from, const char *to, unsigned int flags)
```

เปลี่ยนชื่อ File หรือ Directory

```
static int myfs_chmod(const char *path, mode_t mode, struct fuse_file_info
*fi)
```

เปลี่ยน access control ของ File

```
static int myfs_chown(const char *path, uid_t uid, gid_t gid, struct
fuse_file_info *fi)
```

เปลี่ยน id ของเจ้าของ File

```
static int myfs_truncate(const char *path, off_t size, struct fuse_file_info *fi)
```

เปลี่ยนขนาดของ File

```
static int myfs_open(const char *path, struct fuse_file_info *fi)
```

เปิด File

```
static int myfs_read(const char *path, char *buf, size_t size, off_t offset, struct
fuse_file_info *fi)
```

อ่าน File

```
static int myfs_write(const char *path, const char *buf, size_t size, off_t offset,
struct fuse_file_info *fi)
```

เขียน File

```
static int myfs_release(const char *path, struct fuse_file_info *fi)
```

ปิด File

```
static int myfs_readdir(const char *path, void *buf, fuse_fill_dir_t filler, off_t
offset, struct fuse_file_info *fi, enum fuse_readdir_flags flags)
```

อ่าน Directory

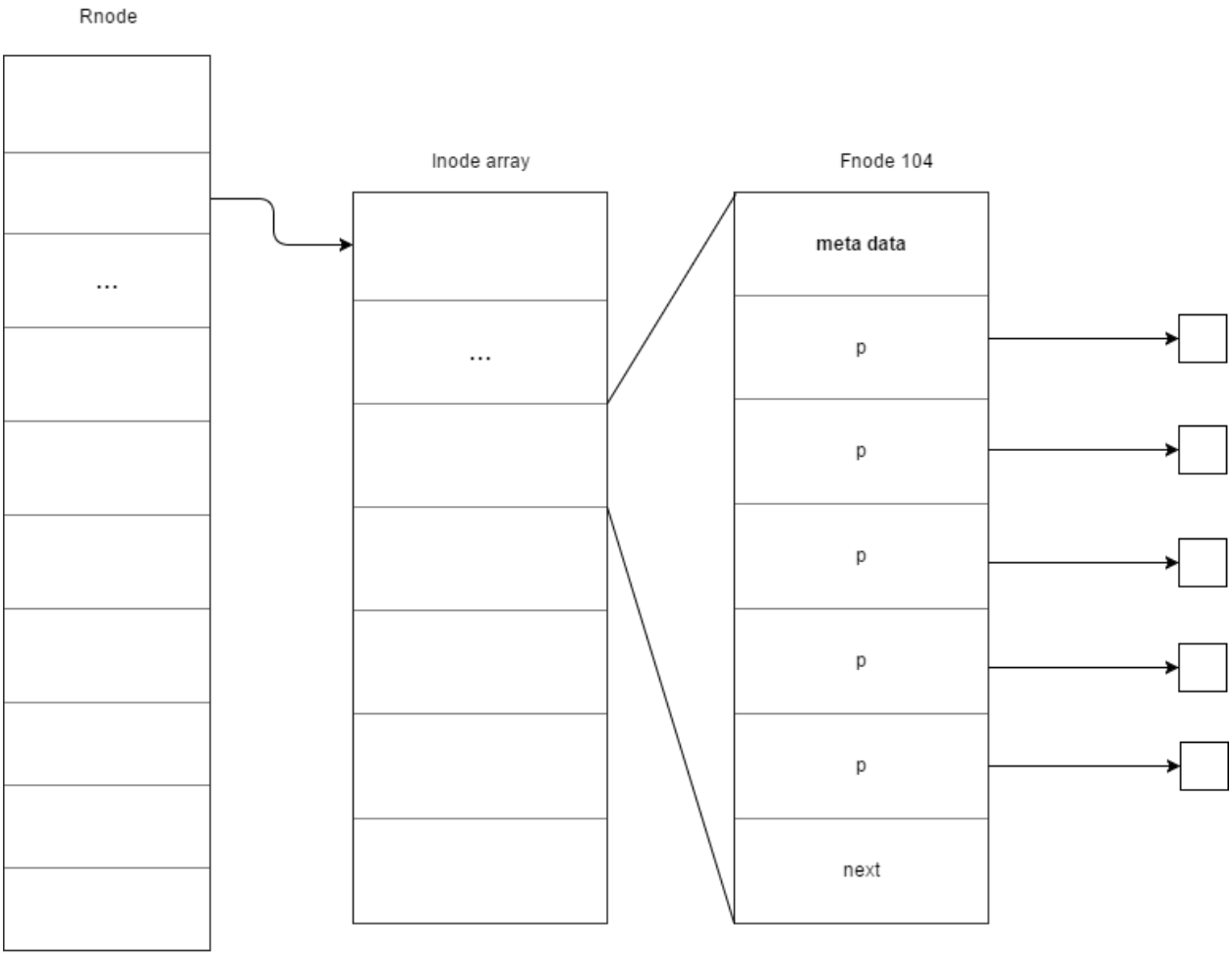
```
static int myfs_release(const char *path, struct fuse_file_info *fi)
```

ปิด Directory

```
static int myfs_fsync(const char *path, int isdatasync, struct fuse_file_info *fi)
```

บังคับเขียนข้อมูลในหน่วยความจำลง File

Design



meta data

- File ID/Name	32
- Last Edited	4
- Permission	1+1
- File Owner	32
- Versioning Number	4
- Valid Flag	1
- Type Flag	1