프로그래밍 수행

결과 보고서

-Assignment 4-

과목명	**|**	운영체제
담당 교수	**|**	차호정 교수님
학과	**|**	교육학과
학번	**|**	2018182058
이름	**|**	박주언
제출일	**|**	2020 년 6 월 29 일

개발 환경

✔ 메모리 정보

```
mingbee@mingbee-VirtualBox:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8G        910M        2.3G         28M        651M        2.7G
Swap:          1.4G          0B        1.4G
```

$free -h 명령어를 사용해 메모리 사용 현황을 확인했다. -h 옵션으로 가독성을 높였다.

```
mingbee@mingbee-VirtualBox:~$ cat /proc/meminfo
MemTotal:        4029820 kB
MemFree:         2339316 kB
MemAvailable:    2821128 kB
Buffers:           97380 kB
Cached:           573548 kB
SwapCached:            0 kB
Active:          1119648 kB
Inactive:         355268 kB
Active(anon):     804904 kB
Inactive(anon):    28384 kB
Active(file):     314744 kB
Inactive(file):   326884 kB
Unevictable:          32 kB
Mlocked:              32 kB
SwapTotal:       1459804 kB
SwapFree:        1459804 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:        804112 kB
Mapped:           241640 kB
Shmem:             29300 kB
KReclaimable:      86488 kB
Slab:             121616 kB
SReclaimable:      86488 kB
SUnreclaim:        35128 kB
KernelStack:        7808 kB
```

```
PageTables:         30004 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:      3474712 kB
Committed_AS:     3792756 kB
VmallocTotal:   34359738367 kB
VmallocUsed:        32920 kB
VmallocChunk:           0 kB
Percpu:              2592 kB
HardwareCorrupted:      0 kB
AnonHugePages:          0 kB
ShmemHugePages:         0 kB
ShmemPmdMapped:         0 kB
CmaTotal:               0 kB
CmaFree:                0 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
Hugetlb:                0 kB
DirectMap4k:       137152 kB
DirectMap2M:      4057088 kB
```

$cat /proc/meminfo 명령어를 사용해 전체 메모리 정보 또한 확인할 수 있었다.

✔ CPU 정보

```
mingbee@mingbee-VirtualBox:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 142
model name      : Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
stepping        : 10
cpu MHz         : 1992.001
cache size      : 8192 KB
physical id     : 0
siblings        : 6
core id         : 0
cpu cores       : 6
apicid          : 0
```

$cat /proc/cpuinfo 명령어를 사용해 CPU 정보를 확인했다. CPU 코어는 6 개다.

✔ 컴파일러 버전

```
mingbee@mingbee-VirtualBox:~$ gcc --version
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

$ gcc --version 명령어를 사용하여 gcc 컴파일러의 버전을 확인하였다.

```
mingbee@mingbee-VirtualBox:~$ g++ --version
g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

$ g++ --version 명령어를 사용하여 g++ 컴파일러의 버전을 확인하였다.

채점환경과 동일한 컴파일 환경을 만들기 위해, gcc 5, g++ 5 패키지도 설치하여 테스트해보았다.

```
mingbee@mingbee-VirtualBox:~$ sudo update-alternatives --config gcc
There are 2 choices for the alternative gcc (providing /usr/bin/gcc).

  Selection    Path            Priority   Status
------------------------------------------------------------
  0            /usr/bin/gcc-7   80        auto mode
* 1            /usr/bin/gcc-5   60        manual mode
  2            /usr/bin/gcc-7   80        manual mode
```

```
mingbee@mingbee-VirtualBox:~$ sudo update-alternatives --config g++
There are 2 choices for the alternative g++ (providing /usr/bin/g++).

  Selection    Path            Priority   Status
------------------------------------------------------------
  0            /usr/bin/g++-7   80        auto mode
* 1            /usr/bin/g++-5   60        manual mode
  2            /usr/bin/g++-7   80        manual mode
```

각각 **$sudo update-alternatives --config gcc** 그리고 **$sudo update-alternatives --config g++** 커맨드로 우분투에 깔려 있는 여러가지 컴파일러 버전을 확인할 수 있었다. 버전을 확인하는 명령어를 입력하는 경우에는 priority 가 높은 7.5 버전이 출력되게 설정하였다.

✔ 시스템 정보

```
mingbee@mingbee-VirtualBox:~$ uname -a
Linux mingbee-VirtualBox 5.3.0-46-generic #38~18.04.1-Ubuntu SMP Tue Mar 31 04:1
7:56 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

$uname -a 명령어를 사용하여 과제 수행을 진행한 시스템의 정보를 확인하였다

Makefile

✔ Makefile 작성

```
As4 > M Makefile
1   project4 :
2       g++ -o project4 project4.cpp
3   clean :
4       rm -f project4
```

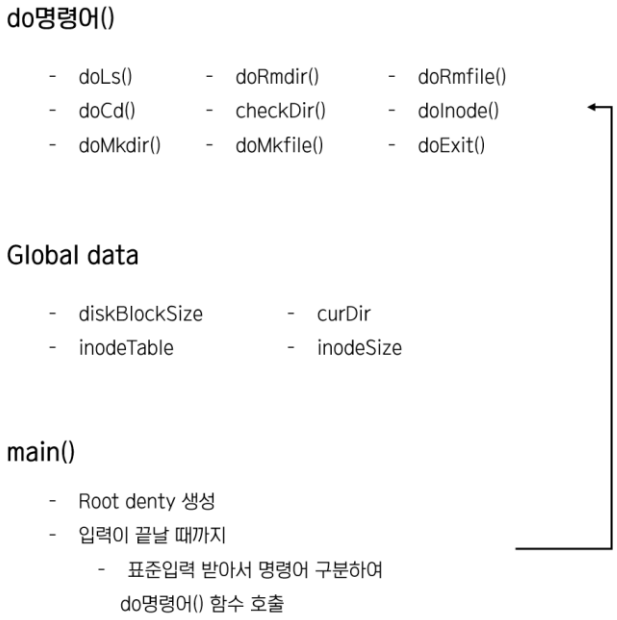
/

저번 과제에서 makefile 작성법을 조금 익혔기 때문에 조금 수월하게 작성할 수 있었다.

```
mingbee@mingbee-VirtualBox:~/Desktop/OS/As4$ make
g++ -o project4 project4.cpp
mingbee@mingbee-VirtualBox:~/Desktop/OS/As4$ make clean
rm -f project4
```

프로그램 구조 설명

✔ 프로그램 전체 구조



먼저 글로벌 데이터로는 disk block 의 남은 사이즈를 저장하는 diskBlockSize int 형 변수가 있다. inodeTable 은 bool 배열로써, 해당 id 의 inode 가 사용가능한지 안한지 저장한다. curDir 은 지금 현재 위치하고 있는 덴트리를 가르키고 있고, 초기값은 루트 덴트리로 설정했다. inodeSize 는 inode id 를 부여하기 위해 현재 어디까지 id 가 부여됐는지 저장하는 변수이다. 또한 글로벌 변수로 super block 오브젝트 또한 생성해준다. inode, dentry, superblock 은 모두 클래스로 구현하였다.

main() 함수로 들어가게 되면 먼저 root dentry 를 생성하여 sb 의 s_root 에 연결해준다. 그리고 이 덴트리를 curDir 로 설정해준다. 그런 다음 while 문을 도는데, exit 명령어가 입력될 때까지 계속해서 명령어를 받아 실행시킨다. 표준입력을 라인별로 읽어와 제일 앞부분 토큰이 명령어에 해당하면 각 명령어로 처리하고, 그렇지 않으면(엔터 혹은 오탈자, 없는 내용들) 모두 에러처리 하고 다음으로 넘겼다. 각 명령어는 명령어에 필요한 인자들을 가지고 do 명령어() 함수를 호출한다. 인자가 여러 개인 명령어(mkdir, rmdir, rmfile 등)는 do 명령어()를 호출하기 전에 먼저 입력된 인자들이 validate 한지부터 체크한 뒤 함수를 호출하였다.

do 명령어() 함수는 각각 명령어가 필요로 하는 작업을 수행한다. 특히 doRmdir() 같은 경우에는, 해당 dir 에 속해있는 모든 파일과 덴트리를 삭제해야하므로 checkDir()이라는 함수를 새로 만들어서 해당 디렉토리의 하위 디렉토리를 모두 돌면서 파일을 삭제해주고 디스크 블록을 반환해주었다.

✔ checkDir()

checkDir(dentry dir)

if (dir의 d_dentry 가 비어있으면)

- dir의 d_inode 의 파일 디스크할당을 해제시켜준다.

return

else (dir의 d_dentry가 비어있지 않으면)

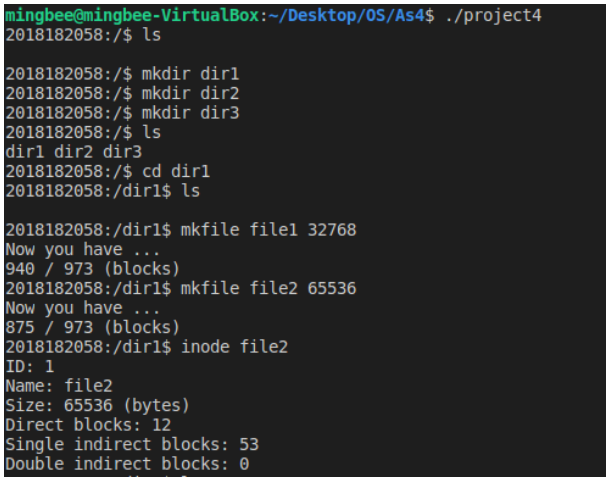
- dir의 d_inode 의 파일 디스크할당을 해제시켜준다.
- dir의 d_dentry 속 각 dentry에 대해

checkDir(each dir in d_dentry)

특히 doRmdir() 를 호출하였을 때 해당 덴트리에 대해 checkDir() 함수를 호출한다. 이 디렉토리의 모든 하위 디렉토리에 대해 checkDir()를 재귀함수로 사용하여 계속해서 체크하고, 디스크 블럭 할당을 해제시켜 준다. 이때 베이스 케이스는 해당 디렉토리의 d_dentry가 비어있는 경우이다.

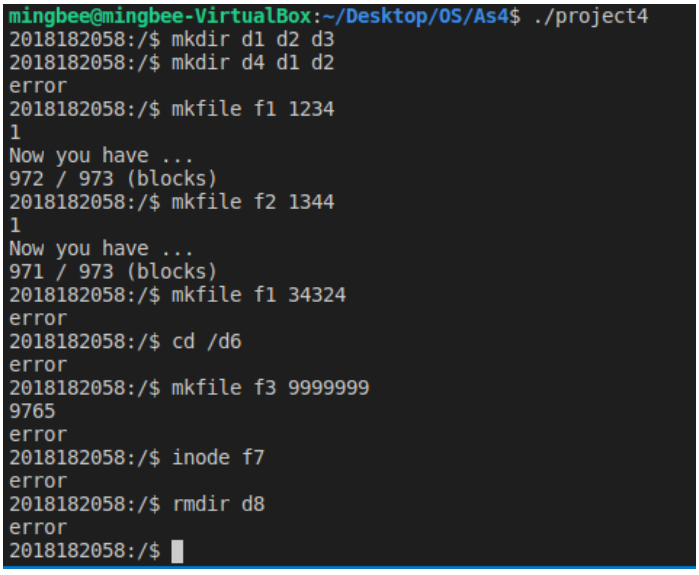
프로그래밍 결과

✔ test input



위 사진과 같이 주어진 input 에 대해서 과제 스펙 속 output 과 동일하게 작동이 되는 것을 확인할 수 있었다.

✔ 예외 처리 테스트



위 사진처럼 여러 가지 경우에서 예외처리가 제대로 동작하고 있는지 확인해 보았다.

1. mkdir 로 여러가지 인수가 들어올 때 특정 디렉토리 이름이 이미 존재하고 있는 경우
2. mkfile 로 파일을 만들 때 이미 있는 파일의 이름을 참조하는 경우

3. cd 명령어로 없는 디렉토리로 이동하려는 경우
4. mkfile 로 허용가능한 범위 이상의 사이즈의 파일을 할당하는 경우
5. inode 명령어로 없는 파일을 참조하는 경우
6. rmdir 로 없는 디렉토리를 참조하는 경우

(결과 화면에 디버깅 과정에서 확인 차 삽입해둔 테스트 출력이 포함되어 있습니다. 양해 부탁드립니다.)

💻 과제 수행 중 애로 사항

가장 어려웠던 것은 c++ 언어 사용의 미숙이었다. 저번 과제와 마찬가지로 포인터 사용이 미숙해서 애를 조금 먹었다. 저번 과제를 통해 포인터 사용에 조금 익숙해지기는 했지만 평소에 c++을 많이 사용하지 않다 보니 조금 잊은 내용들이 있어서 다시 상기시키느라 시간이 걸렸다.

특히 for문 등에서 반복적으로 사용하는 특정 클래스 오브젝트의 포인터 같은 경우에, 계속해서 같은 것을 가리키는 문제가 있었다. 포인터와 주소 참조 등의 개념이 생소하다 보니 같은 실수를 반복했는데, new 커맨드를 사용하여 문제를 해결하는 방법을 알아냈다.

또한 클래스의 생성 객체 관리 등의 문제 또한 자바와 차이점이 있어 다루기 힘든 점들이 있었다. 특히 생성자를 생성하지 않아서 벡터가 자동생성이 안된다거나, 벡터를 생성하려면 꼭 reserve를 써주어야한다는 점 등 아직도 잘 모르겠지만 오류가 많이 떠서 디버깅 할 때 애를 먹었다. 차근차근 c++의 동적 할당 등에 대해 알아가고 익힐 수 있는 시간이었다.

그리고 리눅스 cd 커맨드에 익숙하지 않아서 그 명령어들의 의미와 조합을 생각하는데에 시간이 조금 걸렸다. 그래도 이번 과제를 통해, 파일 시스템의 구조와 구현에 대해 생각해볼 수 있어서 좋았다. 그동안 터미널을 키면 cd 등의 명령어를 처리하기 어렵고 헷갈렸는데 이번 과제를 통해 보다 명확하게 이해하고 숙지할 수 있었던 것 같다.

참고 사이트

강의 슬라이드

강의 교재