운영체제 과제 4
보고서

컴퓨터인공지능학부 202323007 이진선

1. 코드 및 코드 설명

1) 4-1

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

#define MAX_FILES 16
#define MAX_BLOCKS 208
#define BLOCK_SIZE 16
#define MAX_FILENAME_LEN 24
#define MAX_FILE_BLOCKS 3

typedef struct {
    int fsSize;
    int inode_cnt;
    int block_cnt;
    int block_size;
    int free_inode;
    int free_block;

    unsigned char inode_bitmap[MAX_FILES];  //inode 사용여부 표시
    unsigned char block_bitmap[MAX_BLOCKS];  //데이터 사용여부 표시
    unsigned char padding[8];  //정렬용
} superblock;
```

Superblock 구조체는 파일 시스템 전체의 정보를 저장한다. 해당 구조체에는 fssize(전체 파일 시스템 크기), inode_cnt(전체 inode의 개수), block_cnt(전체 데이터블록 개수), block_size(블록 크기), free_inode(사용가능한 inode 개수), free_block(사용가능한 데이터 블록 개수), inode_bitmap(inode 사용여부 표시), block_bitmap(데이터블록 사용여부 표시), padding(구조체 크기 맞추기용 패딩)으로 이루어져있다.

```c
typedef struct {
    unsigned int size;
    unsigned char indirect_block;
    unsigned char block[MAX_FILE_BLOCKS];
    char file_name[MAX_FILENAME_LEN];
} inode;

//파일 시스템 구조
typedef struct {
    superblock SB; //슈퍼블록
    inode inodes[MAX_FILES];  //inode 블록
    unsigned char data_block[MAX_BLOCKS][BLOCK_SIZE]; //data 블록
} filesystem;

static filesystem fs;
```

Inode는 하나의 파일에 대한 정보를 저장한다. Inode 구조체에는 size(파일 크기),

indirect_block(indirect block이 저장되어있는 블록 번호), block(직접 데이터 블록 번호), file_name(파일 이름)로 이루어져있다.

Filesystem 구조체는 전체 파일 시스템을 의미한다. Fread를 통해서 전체 상태를 fs에 저장하는 역할을 한다. filesystem에는 fs(super block 정보), inode(inode 배열), data_block(데이터 블록) 이 저장되어있다.

```
40
41  //수퍼블록 정보
42  void superblock_info() {
43      fprintf(stdout, "Filesystem Status: \n");
44      fprintf(stdout, "Superblock Information: \n");
45      fprintf(stdout, "\tFilesystem Size: %d bytes \n", fs.SB.fsSize);
46      fprintf(stdout, "\tBlock Size: %d bytes\n", fs.SB.block_size);
47      fprintf(stdout, "\tAvailable Inodes: %d/%d\n", fs.SB.free_inode, fs.SB.inode_cnt);
48      fprintf(stdout, "\tAvailable Blocks: %d/%d\n", fs.SB.free_block, fs.SB.block_cnt);
49  }
50
```

Superblock_info()함수는 파일 시스템 전체 구조를 출력하고, superblock에 있는 데이터를 해석한다. 출력사항에 포함되는 변수는 fssize(전체 파일 시스템의 크기), block_size(블록 하나의 크기), inode_cnt(전체 inode의 개수), free_inode(사용가능한 inode수), block_cnt(전체 데이터 블록의 수), free_block(사용가능한 데이터 블록 수)로 이루어져있다.

Fs.SB에 직접 접근하여 출력하는 방식으로 이루어져 해당데이터들에 대한 정보를 가져와 출력한다.

```
51  //indirect block 정보 출력
52  void indirect_block_info(const inode *node) {
53      if(node->indirect_block != 0xFF) { //indirect block이 있는 경우
54          fprintf(stdout, "\t\tIndirect block: %u\n", node->indirect_block);
55          unsigned char *indirect = fs.data_block[node->indirect_block];
56          fprintf(stdout, "\t\tIndirect data blocks: ");
57          for(int j = 0;j<BLOCK_SIZE; ++j) {
58              if(indirect[j]!=0xFF) {  //유효한 indirect block인지지 확인
59                  if(j!=0) fprintf(stdout, ", ");
60                  fprintf(stdout, "%u", indirect[j]);
61              }
62          }
63          fprintf(stdout, "\n");
64      }
65  }
```

파일의 indirect block의 정보를 출력한다. Indirect block==0xFF일 경우 사용되지 않은 것이므로 출력하지 않는다. 해당 블록 번호의 데이터를 fs.data_block을 통해서 가져와서 indirect[]로 받아들인다.

```
67  //direct block 출력
68  void direct_block_info(const inode *node) {
69      fprintf(stdout, "\t\tDirect blocks: ");
70      for(int j = 0;j<MAX_FILE_BLOCKS; ++j) {
71          if(node->block[j] != 0xFF) { //유효한 direct block인지 확인
72              if(j!=0) fprintf(stdout, ", ");
73              fprintf(stdout, "%u", node->block[j]);
74          }
75      }
76      fprintf(stdout, "\n");
77  }
```

Direct_block_info()함수는 각 파일들의 direct block 정보를 출력한다. Inode의 block의 크기만큼 for문으로 순회하여 0xFF를 통해서 비어있는지에 대한 여부를 파악한다. Node->block[j]==0xFF일 경우에는 비어있는 경우이므로 유효한 블록만 형식에 맞추어 출력한다.

```
79  //inode 상세정보 출력
80  void inode_detail_info(int idx, const inode *node){
81      fprintf(stdout, "\tFile: %s\n", node->file_name);
82      fprintf(stdout, "\tSize: %d\n", node->size);
83      fprintf(stdout, "\t\tInode: %d\n", idx);
84
85      direct_block_info(node);
86      indirect_block_info(node);
87  }
```

파일에 해당하는 inode 의 상세 정보를 출력한다. 출력되는 정보는 node->file_name(파일의 이름), node->size(파일의 크기), idx(inode 의 번호)로 이루어져있고, direct block 에 대한 정보 출력과 indirect block 에 대한 정보 출력은 direct_block_info()와 indirect_block_info()를 통해서 출력한다.

```
int main() {
    if(fread(&fs, sizeof(filesystem),1,stdin)!=1) {
        fprintf(stderr, "Read error\n");
        return 1;
    }
    superblock_info();

    fprintf(stdout, "Detailed File Information:\n");

    for(int i = 0; i<MAX_FILES; ++i) {
        if(fs.SB.inode_bitmap[i]) inode_detail_info(i, &fs.inodes[i]);
    }
    return 0;
}
```

Main 은 filesystem 의 크기만큼 입력을 받는데, 제대로 입력되지 않을 경우에는 error 문구를 출력한 뒤, 실행이 완료된다. 만약 제대로 입력된 경우에는 입력된 fs 에 대한 상세한 정보를 출력한다. 먼저 superblock_info()를 통해서 전제 메타데이터를 출력하고, for 문을 통해서 각 inode_bitmap 을 확인하면서 각 inode 에 대한 상세 정보를 출력한다.

2) 4-2

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

#define MAX_FILES 16
#define MAX_BLOCKS 208
#define BLOCK_SIZE 16
#define MAX_FILENAME_LEN 24
#define MAX_FILE_BLOCKS 3

typedef struct {
    int fsSize;
    int inode_cnt;
    int block_cnt;
    int block_size;
    int free_inode;
    int free_block;

    unsigned char inode_bitmap[MAX_FILES];
    unsigned char block_bitmap[MAX_BLOCKS];
    unsigned char padding[8];
} superblock;

typedef struct {
    unsigned int size;
    unsigned char indirect_block;
    unsigned char block[MAX_FILE_BLOCKS];
    char file_name[MAX_FILENAME_LEN];
} inode;

typedef struct {
    superblock SB;
    inode inodes[MAX_FILES];
    unsigned char data_block[MAX_BLOCKS][BLOCK_SIZE];
} filesystem;

```

```
38  static filesystem fs;
39
40  //--------과제 2번 해당 함수-------------------------------
41
42  //파일 이름 기반 inode 인덱스 탐색
43  int find_inode(const char *name) {
44      for(int i = 0; i<MAX_FILES; ++i) {
45          if(strcmp(fs.inodes[i].file_name, name)==0) return i;
46      }
47      return -1; //해당 파일 없음
48  }
49
50  //direct block 데이터 출력
51  void direct_block(const inode *node) {
52      for (int i = 0; i < MAX_FILE_BLOCKS; ++i) {
53          if(node->block[i]!=0xFF) { //invalid block check
54              fprintf(stdout, "Block [%02d] Direct: %.*s\n", node->block[i], BLOCK_SIZE, fs.data_block[node->block[i]]);
55          }
56      }
57  }
58
59  //indirect block 테이블 + 블록 내의 데이터 출력
60  void indirect_block(const inode *node) {
61      if(node->indirect_block == 0xFF) return ; //indirect block 없음음
62
63      unsigned char *indirect = fs.data_block[node->indirect_block];
64      fprintf(stdout, "Block [%02d] Indirect Table: ", node->indirect_block);
65
66      for(int i = 0;i<BLOCK_SIZE; ++i) {
67
68          if(indirect[i]!=0xFF) {
69              if(i!=0) fprintf(stdout, ", ");
70              fprintf(stdout, "%u", indirect[i]);
71          }
72      }
73      fprintf(stdout, "\n");
74
75      //indirect block 번호에 해당하는 데이터 출력
76      for(int i = 0; i<BLOCK_SIZE; ++i) {
77          if(indirect[i]==0xFF) break;
78          fprintf(stdout, "Block [%02d] Indirect: %.*s\n", indirect[i], BLOCK_SIZE, fs.data_block[indirect[i]]);
79      }
80  }
81
```

```
82  //direct + indirect 블록으로 전체 파일내용 출력
83  void file_content(const inode *node) {
84      char content[4096] = {0};
85      int offset = 0;
86
87      //direct 내용
88      for(int i = 0;i<MAX_FILE_BLOCKS; ++i) {
89          if(node->block[i]!=0xFF) {
90              memcpy(content + offset, fs.data_block[node->block[i]], BLOCK_SIZE);
91              offset += BLOCK_SIZE;
92          }
93      }
94      //indirect 내용용
95      if(node->indirect_block!=0xFF){
96          unsigned char *indirect = fs.data_block[node->indirect_block];
97          for(int i = 0; i<BLOCK_SIZE; ++i) {
98              if(indirect[i]==0xFF) continue;
99              memcpy(content+offset, fs.data_block[indirect[i]], BLOCK_SIZE);
100             offset+=BLOCK_SIZE;
101         }
102     }
103
104     content[node->size] = '\0';  //파일 크기까지만 자름
105     fprintf(stdout, "File Contents: %s\n\n", content);
106
107 }
108
109 //파일 내용 결과 모두 출력
110 void print_result(const inode *node, int idx) {
111     fprintf(stdout, "Inode [%02d]: %s (file size : %u B)\n", idx, node->file_name, node->size);
112
113     direct_blocks(node);
114     indirect_block(node);
115     file_content(node);
116 }
117
118 //stdin에서 read 명령어 파싱 후 출력
119 void read_commands(FILE *input) {
120     unsigned char cmd;
121
122     while(fread(&cmd, 1, 1, input)==1) {
123         if (cmd!=0x01) break;
124
125         int name_length = 0;
126         if (fread(&name_length, sizeof(int), 1, input)!=1) break;
127
128         char file_name[256] = {0};
129         if(fread(file_name, 1, name_length, input)!=name_length) break;
130
131         fprintf(stdout, "CMD : READ %s \n", file_name);
132
133         int idx = find_inode(file_name);
134         if(idx == -1) continue;
135
136         print_result(&fs.inodes[idx],idx);
137     }
138 }
139 //---------------------------------------
```

```
140
141 //---------과제 1번 코드와 동일-------------
142 void superblock_info() {
143     fprintf(stdout, "Filesystem Status: \n");
144     fprintf(stdout, "Superblock Information: \n");
145     fprintf(stdout, "\tFilesystem Size: %d bytes \n", fs.SB.fsSize);
146     fprintf(stdout, "\tBlock Size: %d bytes\n", fs.SB.block_size);
147     fprintf(stdout, "\tAvailable Inodes: %d/%d\n", fs.SB.free_inode, fs.SB.inode_cnt);
148     fprintf(stdout, "\tAvailable Blocks: %d/%d\n", fs.SB.free_block, fs.SB.block_cnt);
149 }
150
151 void indirect_block_info(const inode *node) {
152     if(node->indirect_block != 0xFF) {
153         fprintf(stdout, "\t\tIndirect block: %u\n", node->indirect_block);
154         unsigned char *indirect = fs.data_block[node->indirect_block];
155         fprintf(stdout, "\t\tIndirect data blocks: ");
156         for(int j = 0;j<BLOCK_SIZE; ++j) {
157             if(indirect[j]!=0xFF) {
158                 if(j!=0) fprintf(stdout, ", ");
159                 fprintf(stdout, "%u", indirect[j]);
160             }
161         }
162         fprintf(stdout, "\n");
163     }
164 }
165
166 void direct_block_info(const inode *node) {
167     fprintf(stdout, "\t\tDirect blocks: ");
168     for(int j = 0;j<MAX_FILE_BLOCKS; ++j) {
169         if(node->block[j] != 0xFF) {
170             if(j!=0) fprintf(stdout, ", ");
171             fprintf(stdout, "%u", node->block[j]);
172         }
173     }
174     fprintf(stdout, "\n");
175 }
176
177 void inode_detail_info(int idx, const inode *node){
178     fprintf(stdout, "\tFile: %s\n", node->file_name);
179     fprintf(stdout, "\tSize: %d\n", node->size);
180     fprintf(stdout, "\t\tInode: %d\n", idx);
181
182     direct_block_info(node);
183     indirect_block_info(node);
184 }
185 //-----------------------------------------------------
186
187 int main() {
188     if(fread(&fs, sizeof(filesystem),1,stdin)!=1) {
189         fprintf(stderr, "Read error\n");
190         return 1;
191     }
192
193     read_commands(stdin);
194
195     superblock_info();
196     fprintf(stdout, "Detailed File Information:\n");
197     for(int i = 0; i<MAX_FILES; ++i) {
198         if(fs.SB.inode_bitmap[i]) {
199             inode_detail_info(i,&fs.inodes[i]);
200         }
201     }
202
203     return 0;
204 }
```

과제 4-2 에 새롭게 만들어진 함수는 find_inode(), direct_block(), indirect_block(), file_content(), print_result(), read_commands()이다.

Find_inode 함수는 파일 이름에 해당하는 inode 인덱스를 찾는 함수로, fs.inodes[] 배열을 탐색하여 file_name 과 동일한지 여부를 확인하고 찾을 경우에는 해당 인덱스를, 찾지 못한 경우에는 -1 을 반환한다.

Direct_block 함수는 inode 가 가지고 있는 direct block 의 데이터 내용을 출력한다.

Node->block 을 통해서 순회한다. 순회하면서 0xFF 가 아닌 유효한 블록에 대해서 해당 블록의 데이터를 block_size 만큼 fprintf 를 통해 출력한다.

Indirect_block 함수는 inode 의 indirect block 구조를 해석한 뒤, 데이터를 출력한다. 0xFF 인지를 확인하여 유효한 indirect block 에 대해서 fs.data_block[idx]를 통해서 각 내용을 출력한다.

File_content 함수는 전체 내용을 메모리에 복사한 뒤에 출력하는 방식으로 direct block 과 indirect block 에 대해서 내용을 복사한 뒤에 node->size 만큼 출력을 진행한다.

Print_results 는 read 명령을 수행할 경우에 한 inode 에 대해서 정보를 출력한다.

출력하는 정보는 inode 정보를 출력한 뒤에 각 함수를 통해서 세부 내용을 출력한다.

Read_commands 함수는 명령어에 대해서 파싱과 처리를 진행한다. 1 바이트를 읽는데, 0x01 일 경우에는 READ 이므로, 파일 이름 길이를 읽은 뒤에 해당 파일 이름 길이 만큼 문자열을 읽는다. 이후 find_inode 를 통해서 inode 를 찾고, 없을 경우에는 continue 를 통해서 탐색을 진행한다. Print_result 를 통해서 전체 결과를 출력한다.

main 함수에서는 과제 4-1 을 출력하기 전에 read_commands 를 작성하여 해당 읽은 파일에 대해서 과제 4-2 에 해당 하는 함수로 이동하도록 만들었다.

2. Litmus 제출 및 제출결과

202323007의2025 운영체제 과제 4-1제출

6 월 21 일 오후 3 시 30 분 제출

202323007의2025 운영체제 과제 4-2제출

6 월 21 일 오후 3 시 56 분

3. 수행 결과

1) 4-1

```
ubuntu@jcode-os-5-202323007-8575467d4f-hkfkx:~/project/hw4$ gcc -o os4-1 os4-1.c
ubuntu@jcode-os-5-202323007-8575467d4f-hkfkx:~/project/hw4$ ./os4-1 < tests4-1/test4-2.bin
Filesystem Status:
Superblock Information:
        Filesystem Size: 4096 bytes
        Block Size: 16 bytes
        Available Inodes: 12/16
        Available Blocks: 170/208
Detailed File Information:
        File: test_file_1
        Size: 42
                Inode: 0
                Direct blocks: 0, 1, 2
        File: test_file_2
        Size: 91
                Inode: 1
                Direct blocks: 3, 4, 5
                Indirect block: 6
                Indirect data blocks: 7, 8, 9
        File: to_student_osta_message
        Size: 269
                Inode: 2
                Direct blocks: 10, 11, 12
                Indirect block: 13
                Indirect data blocks: 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
        File: bakery_info
        Size: 139
                Inode: 3
                Direct blocks: 28, 29, 30
                Indirect block: 31
                Indirect data blocks: 32, 33, 34, 35, 36, 37
```

2)4-2

```
ubuntu@jcode-os-5-202323007-8575467d4f-hkfkx:~/project/hw4$ gcc -o os4-2 os4-2.c
ubuntu@jcode-os-5-202323007-8575467d4f-hkfkx:~/project/hw4$ ./os4-2 < tests4-2/test4-2.bin
CMD : READ test_file_1
Inode [00]: test_file_1 (file size : 42 B)
Block [00] Direct: Not all treasure
Block [01] Direct:  is silver and g
Block [02] Direct: old, mate.
File Contents: Not all treasure is silver and gold, mate.

CMD : READ test_file_2
Inode [01]: test_file_2 (file size : 91 B)
Block [03] Direct: This is the day
Block [04] Direct: you will always
Block [05] Direct: remember as the
Block [06] Indirect Table: 7, 8, 9
Block [07] Indirect: day you almost c
Block [08] Indirect: aught Captain Ja
Block [09] Indirect: ck Sparrow!
File Contents: This is the day you will always remember as the day you almost caught Captain Jack Sparrow!

CMD : READ to_student_osta_message
Inode [02]: to_student_osta_message (file size : 269 B)
Block [10] Direct: You've really be
Block [11] Direct: en through a lot
Block [12] Direct:  this semester w
Block [13] Indirect Table: 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
Block [14] Indirect: ith your OS assi
Block [15] Indirect: gnments. Those w
Block [16] Indirect: ere some challen
Block [17] Indirect: ging problems, b
Block [18] Indirect: ut I'm sure othe
Block [19] Indirect: r coursework wil
Block [20] Indirect: l feel much more
Block [21] Indirect:  manageable afte
Block [22] Indirect: r this experienc
Block [23] Indirect: e. I totally fee
Block [24] Indirect: l your pain - be
Block [25] Indirect: ing an OS TA was
Block [26] Indirect:  incredibly toug
Block [27] Indirect: h for me too.
```

```
File Contents: You've really been through a lot this semester with your OS assignments. Those were some challenging problems, but I'm sure other coursework will feel much more manageable after this experience. I totally feel your pain - being an OS TA was incredibly tough for me too.

CMD : READ bakery_info
Inode [03]: bakery_info (file size : 139 B)
Block [28] Direct: Sungsimdang: Kor
Block [29] Direct: ea's Beloved Bak
Block [30] Direct: ery Institution.
Block [31] Indirect Table: 32, 33, 34, 35, 36, 37
Block [32] Indirect:  Sungsimdang is
Block [33] Indirect: one of South Kor
Block [34] Indirect: ea's most iconic
Block [35] Indirect:  bakery chains,
Block [36] Indirect: founded in 1956
Block [37] Indirect: in Daejeon.
File Contents: Sungsimdang: Korea's Beloved Bakery Institution. Sungsimdang is one of South Korea's most iconic bakery chains, founded in 1956 in Daejeon.

Filesystem Status:
Superblock Information:
        Filesystem Size: 4096 bytes
        Block Size: 16 bytes
        Available Inodes: 12/16
        Available Blocks: 170/208
Detailed File Information:
        File: test_file_1
        Size: 42
                Inode: 0
                Direct blocks: 0, 1, 2
        File: test_file_2
        Size: 91
                Inode: 1
                Direct blocks: 3, 4, 5
                Indirect block: 6
                Indirect data blocks: 7, 8, 9
        File: to_student_osta_message
        Size: 269
                Inode: 2
                Direct blocks: 10, 11, 12
                Indirect block: 13
                Indirect data blocks: 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
```

Ln 1, Col 1 Spaces: 2 UTF-8 LF Plain Te

```
                Indirect data blocks: 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
        File: bakery_info
        Size: 139
                Inode: 3
                Direct blocks: 28, 29, 30
                Indirect block: 31
                Indirect data blocks: 32, 33, 34, 35, 36, 37
ubuntu@jcode-os-5-202323007-8575467d4f-hkfkx:~/project/hw4$
```

4. 과제 수행 시 어려웠던 점 및 해결 방안

1) 과제 4-1

어려웠던 점: 구조체를 만든 과정에서 데이터가 어떠한 형식과 방법으로 위치해 있는지에 대해서 이해하기 어려웠고, 이에 대한 구조를 짜는 것에 대해서 어려움을 겪었다.

해결 방안 : 출력해야하는 요소들에 대해서 먼저 분석한 뒤에 해당 정보들이 어떻게 나뉘는지 파악했다. 그래서 출력해야하는 요소들을 먼저 함수로 구분하고 나니 데이터를 어떻게 구분해서 배치할 수 있을지 파악할 수 있게 되었다.

2) 과제 3-2

어려웠던 점: 코드에 대해서 각 함수를 어떻게 분리할지에 대해서 고민이 많았다. 추가해야하는 코드에 대해서 정확하게 이해하지 못하니 발생하는 문제점이였다.

해결 방안 : 출력해야하는 부분에 대해서 먼저 파트를 구분한 뒤에 각 파트마다 연관되어있는 요소를 통해서 함수를 어떻게 구분할지에 대해서 파악할 수 있었다.