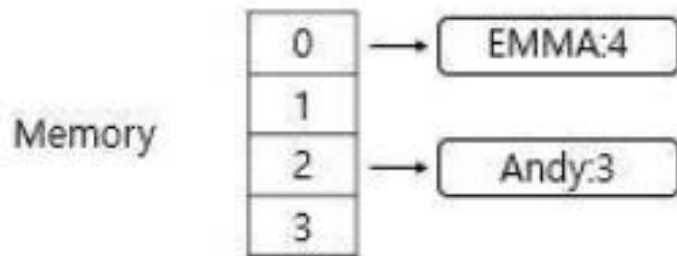


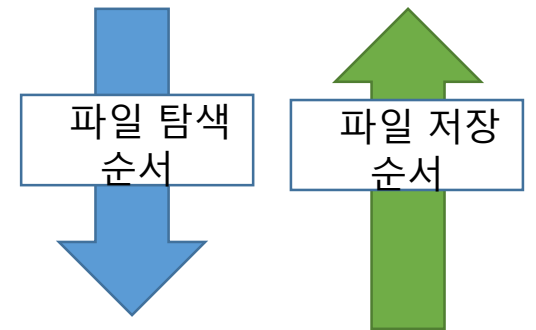
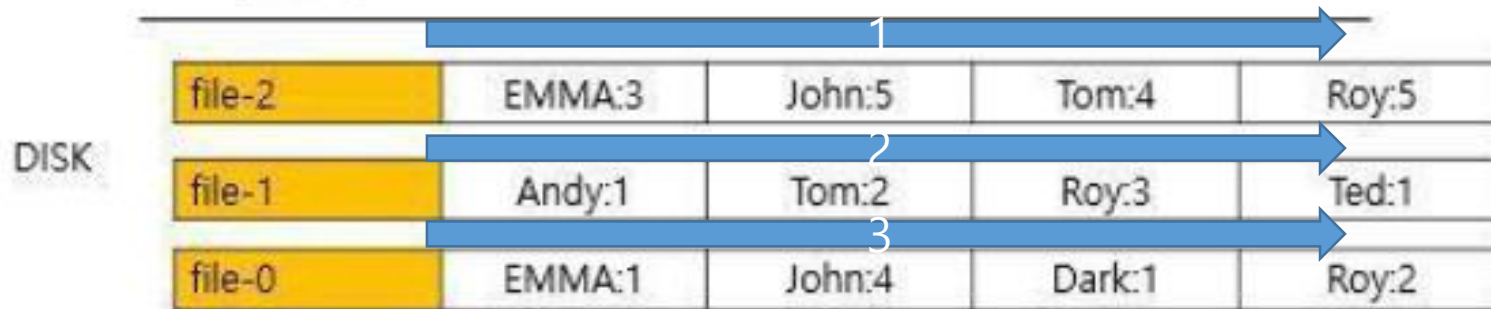
# PA2

2016310932배현웅

# 개요



Memory 에 size와 같은 개수의 데이터가 저장되면, disk에 내린다(file-N 저장을 통해)



※코드설명에서 보여지는 코드의 내용과 실제 제출 코드가 조금 다릅니다.  
하지만 최신 데이터 찾는 방법과 파일 구조는 동일합니다. 이해주시면 감사하겠습니다.

# 파일 구조

- 기본적으로 file-0, file-1... 순으로 저장이 된다.
- 즉 file-%d에서 %d 숫자가 높을수록 최신데이터이다.
- 최신 데이터를 찾는 것은 역순으로 탐색한다. File-10, file-9 ...
- 파일 내부에는 key/value\Wn 형식으로 size 개수가 저장이 된다.

• ex)

```
1 Emma/3
2 passed/1
3 clever/1
4 remembrance/1
5 family/1
6 period/1
7 hardly/1
8 place/1
9 being/1
```

# 파일구조 저장방법

```
130 // table is full
131
132 if(table_num >= Lsize){
133     num++;
134     char * buf1 = malloc(sizeof(char)*100);
135
136     sprintf(buf1, "./db/file-%d", num);
137
138     int fd = open(buf1, O_RDWR | O_CREAT | O_TRUNC, 0755);
139
140     for(int i = 0; i < Lsize; i++){
141         db_t* cu = (db+i); // to search table
142         db_t* tmp = cu->next;
143         if(tmp == NULL) continue;
144         for(cu = tmp; cu != NULL; cu = tmp) { // key/value
145             int trash;
146             sprintf(buf1, "%s/%d\n", cu->key, cu->value);
147             trash = write(fd, buf1, strlen(buf1));
148             trash++;
149             tmp = cu->next;
150             cu->key = NULL;
151
152             free(cu->key);
153             //free(cu);
154         }
155         (db+i)->next = NULL;
156     }
157     free(buf1);
158     table_num=0;
159     close(fd);
160
161 }
```

130 - 161 메모리상의 hashtable 에 올라가있는 key/value 개수가 size(=Lsize)개수가 넘으면 파일에 저장한다.

133 - num++ 통해 파일이름을 하나 증가한다.

134 - buf1 : key/value 저장하기 위한 자료구조

136-138 file-(num값) 파일을 하나 생성하며 작성할 준비를 한다.

140 - Lsize 개수만큼 진행한다.

141-142 - cu에는 db의 i번째 위치에, tmp 에 cu->next 저장한다.

143 - i번째에 데이터가 없으면 스킵 i+1번째로

144-154 - trash는 write의 결과값을 받기위한 쓰레기통같은 역할. Sprintf는 한줄에 write되는 것 (%s/%s\n)을 buf1에 저장하고 write하는 것. 그 이후 다음 번째 데이터에 접근한다.

155 - load 해준 데이터인 db의 i 번째를 NULL로 초기화

157-179 -데이터 초기화 및 데이터 free

# 최신 데이터를 찾아내는 방법(1) 기존

- 파일을 저장할 때 num을 증가시키면서 최신파일의 위치를 num을 통해 저장해둔다.
- 메모리상의 해시테이블에 key가 없으면 파일을 탐색한다.
- Num에 저장되어있는 것을 통해 최신 파일부터 탐색한다.
- Ex) num = 3이라면 최신 파일이 file-3이고, 탐색 file-3, file-2, file-1, file-0 순으로 탐색한다. (역순으로 탐색)

# 최신 데이터를 찾아내는 방법(2) - 중단시

- 파일이 중단에 중단되었을 때 db를 이어서 탐색하는 법
- EX) `./wordcount 128 < 1EMMA.word && ./wordcount 128 < 2EMMA.word`
- 1EMMA 데이터 넣을 때는 기존과 동일
- 2EMMA 데이터를 넣을 때는 자식 프로세스에서 db폴더에서 `exec1`를 이용해 `ls -t` 로 가장 최근에 만들어진 파일을 가장 먼저 받는 순서로 정렬받은 정보를 받는다.
- pipe를 통해 전달, 그 것을 입력받아 첫번째 파일의 숫자(file-num, 여기서 num)을 받아 그것을 지역변수인 num에 저장

# 최신 데이터 찾기(1-1)

```
190 // find in the file
191 for(int k=num; k>=0; k--){
192     char *buf1 = malloc(sizeof(char)*MAXSIZE);
193     sprintf(buf1, "./db/file-%d", k);
194     struct stat file_info;
195     stat(buf1, &file_info);
196     int file_size = file_info.st_size;
197     int fd = open(buf1, O_RDONLY);
198
199     char* s1 = (char*)malloc(sizeof(char)*(file_size));
200     int trash = read(fd, s1, sizeof(char)*file_size);
201     trash++;
202     int sp = 0;
203     while(sp < file_size) { // check line (number: lsize)
204         char* num1 = (char*)malloc(sizeof(char)*100);
205         char* str = (char*)malloc(sizeof(char)*1024);
206         char bar = '/';
207         char enter = '\n';
208         int i;
209
```

191 : file-(num), file-(num-1) 순으로 탐색할 것

192 : buf1 : 파일을 받아오기위한 상대주소를 받는 char\* 자료형 할당

194-196 : stat file\_info를 통해 file정보를 받아서 st\_size로 file의 크기를 받아온다.

197 - 200 : file-(k) 을 열고 s1을 통해 파일에 있는 모든 데이터를 받아온다.

200-201 : trash는 read의 값을 받기위한 쓰레기변수

202 : sp = 0 초기화를 통해 파일 처음부터 읽기

203 : sp가 file\_size 넘어가지 않을때까지 받는다.

204-208 : 변수 설정

# 최신 데이터 찾기(1-2)

```
209
210     for(i=0; (s1[sp] != bar); sp++){
211         str[i] = s1[sp];
212         i++;
213     } str[i] = '\0';
214     sp++;
215
216     for(i=0; s1[sp] != enter; sp++){
217         num1[i] = s1[sp];
218         i++;
219     } num1[i] = '\0';
220     sp++;
221
222     if(strcmp(str, key) == 0){
223         value = (char *)malloc(sizeof(char)*strlen(str));
224         *(int*)value = atoi(num1);
225
226         infile = 1;
227         sp = 0; free(num1); free(str); close(fd); free(s1);
228         return (value);
229     }
230
231     free(num1);
232     free(str);
233
234 }
235 close(fd);
236 free(s1);
237 free(buf1);
238
239 }
240
241 return value;
242 }
```

210-213 : str에 key값저장

216-220 : num1 에 value 값 저장

222-229 : argument 에서 받은 key값이랑 str랑  
같으면 value에 num1값 저장하고 리턴



# 최신 데이터 찾기(2)

```
47 db_t *db_open(int size)
48 {
49     mkdir("./db",0777); // mk file directory
50     int fd[2];
51     pipe(fd);
52     if(fork()==0){ // child
53         close(fd[0]);
54         dup2(fd[1],fileno(stdout));
55         execl("/bin/ls","ls","-t","./db/",NULL);
56     } else {
57         wait(NULL);
58         close(fd[1]);
59         char * cat = (char*)malloc(sizeof(char)*200);
60         read(fd[0],cat,200);
61         if(strlen(cat) != 0) {
62             int i=0;
63             for(; cat[i] != '-'; i++);
64             i++;
65             int start = i;
66             for(; cat[i] != '\n'; i++){
67                 //printf("%c",cat[i]);
68             }
69             char p[140];
70             strncpy(p,cat+start,i-start);
71             num = atoi(p);
72             //printf("num:%d\n",num);
73         }
74     }
75     db_t *db = NULL;
76     db = (db_t*)malloc(sizeof(db_t)*size);
77     lsize = size;
78     for(int i=0; i<size; i++){ // initializer
79         (db+i)->next = NULL;
80     }
81     return db;
82 }
83
```

50-51 : 자식 프로세스와 통신할 파이프 설정

52-55 : execl 에서 ls -t ./db/를 실행하여 최신파일 순서로 받는다.

56-60 : 자식프로세스가 끝날때 까지 기다리고 cat 이라는 변수를 통해 파이프에서 건네지는 값을 받는다.

61: 받은 값이 아무것도 아니라면 파일이 아무것도 없는 것이니 처음처럼 시작하면 되고(if 가 틀릴때) cat 값이 존재한다면 if문을 실행한다.

63 : file-%d 중에서 file을 지나치는 부분

65 : 숫자가 시작되는 부분을 저장하기 위한 start변수

66 : \n으로 구분되어 입력을 받으므로 \n받을 때까지 진행하여 숫자의 범위를 예측한다.

69-71 : 파일이름의 숫자부분을 strncpy로 옮기고 num 값에 p를 int형으로 변환하여 넣는다.