시스템SW실습2 PA2

2016310936 우승민

이번 과제는 이전 과제에서 구현한 hash table에서 사이즈 크기의 단어쌍이 입력되면 파일로 저장하여 메모리를 계속 비워주면서 계속 사용하게 하는 과제입니다. 기존 코드에 추가된 부분만 기술하겠습니다.

```
#include "db.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>
#include <fcntl.h>

int t;
char max[100];
char nu[100];
int num;
```

우선 추가적으로 사용한 헤더파일로는 파일을 open하고 read를 하기 위해 **fcntl.h**를 추가하였습니다. 그리고 db폴더를 생성하고 기존 파일들을 확인하기위해 fork, execl함수를 사용할 목적으로 **unistd.h**와 **wait.h**를 추가하였습니다.

전역변수인 max와 nu가 기존 파일들의 이름을 확인하여 가장 최신 파일의 이름을 저장하기위해 사용한 변수입니다.

Db_open에서 추가된 내용을 보면

```
int child_status;
if(fork() == 0){
        execl("/bin/mkdir", "mkdir ", "-p", "db", NULL);

}
wait(&child_status);

const char *file = "a.txt";


if(fork()==0){
        int fd =open(file, O_RDWR|O_CREAT, 0755);
        dup2(fd,1);
        execl("/bin/ls", "ls", "db", NULL);
}
wait(&child_status);
```

Fork 함수를 사용하여 mkdir과 ls를 사용할 child프로세서를 만들어주었고, dup2함수를 사용하여 stdout 대신 a.txt에 db의 파일 목록을 출력하게 만들었습니다.

```
int fd =open(file, O_RDONLY, 0755);
int z;
int p=0;
max[0]='0';
char buffer[1];
```

a.txt를 열고 max에 초기값을 0으로 설정해 주었습니다. 저는 파일을 0부터 숫자를 하나씩 늘려주며 저장하였기 때문에 숫자가 클수록 최신파일입니다.

```
while(1){
        while((z = read(fd, buffer,1))==1){
                nu[p++]=buffer[0];
                if(buffer[0]=='\n')
                        break;
        }

        if(p>strlen(max)){
                for(int t=0; t<p; t++)
                        max[t]=nu[t];
                }
        else if(p==strlen(max)){
                if(strcmp(nu,max)==1){
                        for(int t=0; t<p; t++)
                                max[t]=nu[t];
                        }
        }
        p=0;
        if(z==0)
                break;
}
close(fd);
num = atoi(max);
```

파일이름을 한줄씩 받아서 기존의 max값과 비교하주고 만약 파일이름이 더 크면 max에 넣어주는 것을 파일이 끝날때까지 반복해주었습니다.

최종적으로 나온 max값을 int로 바꾸어주어 num에 넣어주었습니다.

```
char *db_get(db_t *db, char *key, int key_len,
                        int *val_len)
{
        char *value = NULL;
        int x=key[0]-65;
        if(x>=t)
                x = x%t;
        db_t * cur = &db[x];
```

다음으로 db_get함수에서 이전 과제와 다른점중 가장 먼저나오는 것은 이전 과제는 사이즈를 충분히 크게 만들어 알파벳순으로 모두 구분이 가능했지만, 이번 과제는 사이즈가 작을수도 있기때문에 알파벳을 사이즈로 나누어 나머지를 넣어주었습니다.

```
if(db[x].word==NULL) {
        if(num==0){
                return value;
                }
        const char *file_src1 = NULL;
        for(i=num; i>=0; i--){
                char *b= (char*)malloc(sizeof(char)*100);
                strcat(b,"./db/");
                char *ff = (char*)malloc(sizeof(char)*10);
                sprintf(ff,"%d",i);
                strcat(b,ff);
                file_src1 = b;
                fd_src1 = open(file_src1, O_RDONLY, 0755);
                int a;
```

key 값의 head structure 가 NULL 인 상황에서 만약 num 이 0 이라면 파일이 없는 것이므로 단어를 받은 적이 없는 것으로 바로 value 를 return 해주면 됩니다. Num 이 0 이 아니면 최신파일부터 탐색을 시작합니다.

```
while(1){
        int p=0;
        while((a=read(fd_src1,buffer,1))==1){
                if(buffer[0]==' ')
                        break;
                save[p++]=buffer[0];
                }
        save[p]='₩0';
        if(save[0]=='₩0')
                break;
        if(strcmp(save, key)==0){
                int k=0;
                while((a=read(fd_src1,buffer,1))==1){
                        if(buffer[0]=='₩n')
                                break;
                        c[k++]=buffer[0];
                        }
                c[k]='₩0';
                int temp = atoi(c);
                value = (char*)malloc(sizeof(int));
                *(int*)value = temp;
                close(fd_src1);
                free(b);
                free(ff);
                return value;
                }
```

파일 형식이 "word 7₩n" 여러 개로 되어있으므로 띄어쓰기까지 save 에 저장해주면 단어부분만 가져옵니다.

단어와 key 를 비교하여 같으면 띄어쓰기 이후의 숫자를 c 에 받아 int 형으로 바꾸어 temp 에 넣은 후 value 에 넣어주고 return 을 해줍니다.

만약 save 와 key 가 다르면 다음줄로 넘어가서 파일을 다시확인 합니다.

```
        else{
                while((a=read(fd_src1,buffer,1))==1){
                        if(buffer[0]=='₩n')
                                break;
                        }
                }
        }
free(b);
free(ff);
close(fd_src1);
}
```

파일 확인이 끝나면 malloc 한 메모리를 free 하고 파일을 닫은 후 위의 for 문을 다시 실행합니다.

```
else{
        while(1){
                if(cur->next == NULL){
                        break;
                }
                if(strcmp(key,cur->word)!=0)
                        cur = cur->next;
                else
                        break;
                }

        if(cur->word!=NULL){
                if(strcmp(key,cur->word)==0){
                        value = (char*)malloc(sizeof(int));
                        *(int*)value = cur -> count;
                        return value;
                        }
```

여기서 else 문은 key 값의 head structure 에 단어가 들어있을 경우입니다.

이런 경우에는 key 값의 table 을 우선적으로 확인하고 만약 key 와 같은 단어를 가진 structure 가 있으면 value 값을 structure 의 count 값으로 받고 return 해줍니다. 만약 table 에 같은 단어가 없을 경우는 이전과 동일하게 파일을 탐색합니다.

Db_put 함수로 넘어가겠습니다.

우선 ss 는 전역변수로 db_put 에서 table 에 단어를 하나씩 추가할때마다 0 에서부터 증가하게 만들어주었습니다. 만약 ss 와 t 가 같다면 size 만큼의 단어를 받았다는 것이고 이 상태에서 value 값이 1 인 즉 새로운 단어를 받은 상황을 코드로 표현한 것입니다.

```
if(*(int*)val==1){
        if(ss==t){
                const char *file_src2 = NULL;
                int fd_src2;
                char *b= (char*)malloc(sizeof(char)*100);
                strcat(b,"./db/");
                char *ff = (char*)malloc(sizeof(char)*10);
                num++;
                sprintf(ff,"%d",num);
                strcat(b,ff);
                file_src2 =b;
                fd_src2 = open(file_src2, O_RDWR|O_CREAT, 0755);
                int s=0;
                while(s<t){
                        db_t *sa = &db[s];
                        while(sa->word){
                                char *cc = (char*)malloc(sizeof(char)*5);
                                write(fd_src2, sa->word, strlen(sa->word));
                                write(fd_src2, " ",1);
                                sprintf(cc,"%d", sa->count);
                                write(fd_src2, cc, strlen(cc));
                                write(fd_src2, "₩n",1);
                                free(cc);
                                if(sa->next == NULL)
                                        break;
                                sa = sa->next;
                        }
                        s++;
                }
```

파일을 추가했다는 의미로 num 을 1 키우고 그 값을 파일이름으로 정하고 structure 내 word 값이 있으면 반복하여 "EMMA 1₩n" 형태로 write 하였습니다.

```
db_t *cc = (db_t *)malloc(sizeof(db_t));
db_t *tt = (db_t *)malloc(sizeof(db_t));
int i;
for(i=0; i<t; i++){
        if(db[i].word == NULL)
                continue;
        if(db[i].next != NULL){
                cc = db[i].next;
                while(cc->next != NULL){
                        tt = cc ->next;
                        free(cc->word);
                        free(cc);
                        cc = tt;
                }
                tt = cc;
                free(tt->word);
                free(tt);
        }
}
for(i=0; i<t; i++){
        if(db[i].word !=NULL){
                free(db[i].word);
                }
        db[i].word=0;
        db[i].next=NULL;
        }
close(fd_src2);
free(b);
free(ff);
ss=0;
num++;
```

그 후 기존에 있던 메모리를 free 하는 부분입니다. 이전 과제의 close 와 동일한 방법으로 free 해주고 ss 는 0 으로 초기화하였습니다.

```
if(db[x].word==0){
        db[x].word=temp->word;
        db[x].count=temp->count;
        db[x].next=NULL;
        free(temp);
        ss++;
        return ;
        }
else{
        while(1){
                if(cur -> next == NULL)
                        break;
                cur = cur ->next;
                }
        cur -> next = temp;
        ss++;
        return ;
        }
```

Free 이후 새로 추가된 단어를 table 에 추가하는 부분입니다.

만약 key 값에 해당하는 head structure 가 비워있으면 추가된 단어를 head 로 만들어주고 아니면 비워져있는데 까지 next 로 넘겨 마지막에 temp 를 추가해줍니다.

단어를 추가했음으로 ss 를 1 키워 return 해주었습니다.

```
else{
        while(cur->word!=NULL){
                if(strcmp(cur->word,key)==0){
                        cur->count = cur->count+1;
                        free(temp);
                        return;
                        }
                if(cur->next==NULL)
                        break;
                cur = cur-> next;
                }

        if(ss==t){
```

else 구문은 넘어온 value 값이 1 이 아닌경우입니다. 우선 테이블에 단어가 있는지부터 확인해보고 만약 없으면 위와 같은방법으로 기존 테이블이 다 사이즈만큼 차있으면 파일로 저장후 비워준 후에 table 에 단어를 새로 써줍니다.

```
if(db[x].word==0){
        db[x].word=temp->word;
        db[x].count=temp->count;
        db[x].next=NULL;
        free(temp);
        ss++;
        return ;
        }
else{
        cur ->next = temp;
        ss++;
        return ;
        }
```

아까와 한가지 다른점이 있다면 else 문 처음에 cur 를 이미 마지막까지 넘겨주었기 때문에 next 과정을 생략했다는 것입니다.

마지막으로 db_close 함수를 보면

```
void db_close(db_t *db)
{

        const char *file_src2 = NULL;
        int fd_src2;
        char *b= (char*)malloc(sizeof(char)*100);
        strcat(b,"./db/");
        char *ff = (char*)malloc(sizeof(char)*10);
        num++;
        sprintf(ff,"%d",num);
        strcat(b,ff);
        file_src2 =b;
        fd_src2 = open(file_src2, O_RDWR|O_CREAT, 0755);
        int s=0;
        while(s<t){
                db_t *sa = &db[s];
                while(sa->word){
                        char *cc = (char*)malloc(sizeof(char)*5);
                        write(fd_src2, sa->word, strlen(sa->word));
                        write(fd_src2, " ",1);
                        sprintf(cc,"%d", sa->count);
                        write(fd_src2, cc, strlen(cc));
                        write(fd_src2, "\n",1);
                        free(cc);
                        if(sa->next == NULL)
                                break;
                        sa = sa->next;
                        }
                s++;
        }
        close(fd_src2);
        free(b);
        free(ff);
```

기존 함수에서 단순히 남아있는 table의 데이터를 파일로 저장하는 것만 추가하였습니다.