[PA1 Report] 2017313008

<IndexBuilder.c>

```c
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

struct node{
        char name[30];
        int chap, clau, place;
        int count, wordLen;
        struct node* next;
};

struct node *hashTable[90000000];

struct node* createNode(char str[], int chap, int clau, int place, int len){
        struct node* newNode=(struct node*)malloc(sizeof(struct node));
        int i;
        for(i=0;i<len;i++) newNode->name[i]=str[i];
        newNode->name[i]='\0';
        newNode->wordLen=len;
        newNode->chap=chap;
        newNode->clau=clau;
        newNode->place=place;

        newNode->next=NULL;

        return newNode;
}

int hashCode(char str[], int len){
        int r=0;
        for(int i=0;i<len;i++) r=r*65599+str[i];
        return r^(r>>16);
}


void insert(char str[], int chap, int clau, int place, int len){
        int hashIndex=hashCode(str,len)%90000000;

        struct node *newNode=createNode(str, chap, clau, place, len);
        struct node *n1;

        if(hashTable[hashIndex]==NULL){
                hashTable[hashIndex]=newNode;
                hashTable[hashIndex]->count=1;
        }
        else{
                for(n1=hashTable[hashIndex];n1->next!=NULL;n1=n1->next);
                n1->next=newNode;
                hashTable[hashIndex]->count++;
        }
}

int intToChar(int num, char *c, int start){
        int cnt=0, div=1, m=start;
        while((num/div)>0) cnt++, div*=10;
        div/=10;
        for(;cnt>0;cnt--,m++,div/=10){
                c[m]=(num/div)+'0';
                num-=((num/div)*div);
        }

        return m-start;
}
```

- node 구조체 { name: 단어, chap: 장, clau: 절, place:위치, count: 출현횟수, wordLen: 단어길이 }
- createNode : 단어를 insert할 때 그 단어의 정보를 담은 node를 create한다.
- hashCode : 단어를 insert할 때 그 단어가 hashTable에서 몇번째 자리에 올지를 결정해줌. 같은 단어이면 같은 hashTable[idx]에 들어간다. 성경파일들의 단어들에서는 충돌이 일어나지 않는다. str은 단어이고 len은 단어의 길이이다.
- Insert: 성경 txt파일을 읽어들이다가 단어가 등장할 때마다 그 단어의 정보를 담은 node를 hashTable에 insert한다.
- intToChar: 정수 num을 c에 문자열로 저장한다. start는 c index의 시작지점으로, 만약 start=3이면 c[3]부터 문자열이 저장된다. 그리고 문자열의 길이(m-start)를 return 한다.

```
void indexBuilder(const char* inputFileNm, const char* indexFileNm)
{
        /* Write your own C code */

        int fd=open(inputFileNm,O_RDONLY);
        int ifd=open(indexFileNm, O_RDWR | O_CREAT, 0755);
        int now=0, start=0, cntle=0, chap=0, clau=0, pastChap=1,pastClau=1, m=0, k=-1, byte=0, ok=0,
numOfChap=0, numOfClau=-1, numOfWord=0, numOfIndex=0, cn=0, ch=0, cl=0, pl=0, id=0, wd=0, kh=0, gh=0;
        char a[2], b[3], c[4], word[30], num[30];


        while(read(fd,a,2)>0){
                if(a[0]=='\n' || a[1]=='\n') start=1, cntle=0, kh=1;
                if(start==1 && 47<a[0] && a[0]<58 && 47<a[1] && a[1]<58){
                        chap=(a[0]-'0')*10+(a[1]-'0'), read(fd,a,2);
                        if(a[0]==':' && 47<a[1] && a[1]<58){
                                cntle=1, start=0, k=-1;
                                clau=a[1]-'0', read(fd,a,2);
                                if(47<a[0] && a[0]<58 && a[1]==':')
                                        clau=clau*10+(a[0]-'0');
                        }
                        else if(47<a[0] && a[0]<58 && a[1]==':'){
                                cntle=1, start=0, k=-1;
                                chap=chap*10+(a[0]-'0'), read(fd,b,3);
                                if(47<b[0] && b[0]<58 && b[1]==':')
                                        clau=(b[0]-'0');
                                else if(47<b[0] && b[0]<58 && 47<b[1] && b[1]<58 && b[2]==':')
                                        clau=(b[0]-'0')*10+(b[1]-'0');
                                else if(47<b[0] && b[0]<58 && 47<b[1] && b[1]<58 && 47<b[2] && b[2]<58
                                        clau=(b[0]-'0')*100+(b[1]-'0')*10+(b[2]-'0');
                                lseek(fd,-1, SEEK_CUR);
                        }
                        else chap=0;
                }

                else if(start==1 && 47<a[0] && a[0]<58 && a[1]==':'){
                        chap=a[0]-'0', read(fd,a,2);
                        if(47<a[0] && a[0]<58 && 47<a[1] && a[1]<58)
                                clau=(a[0]-'0')*10+(a[1]-'0'), cntle=1, start=0, k=-1;
                        else if(47<a[0] && a[0]<58 && a[1]==':')
                                clau=a[0]-'0', cntle=1, start=0, k=-1;
                        else chap=0;
                }
```

```
else if(start==1 && (a[0]=='\n' || a[0]==' ') && 47<a[1] && a[1]<58){
        chap=a[1]-'0', read(fd,a,2);
        if(47<a[0] && a[0]<58 && a[1]==':'){
                cntle=1, start=0, k=-1;
                chap=chap*10+(a[0]-'0'), read(fd,a,2);
                if(47<a[0] && a[0]<58 && a[1]==':')
                        clau=a[0]-'0';
                else if(47<a[0] && a[0]<58 && 47<a[1] && a[1]<58)
                        clau=(a[0]-'0')*10+(a[1]-'0');
        }
        else if(a[0]==':' && 47<a[1] && a[1]<58){
                cntle=1, start=0, k=-1;
                clau=a[1]-'0', read(fd,a,2);
                if(47<a[0] && a[0]<58 && a[1]==':')
                        clau=clau*10+(a[0]-'0');
        }
        else if(47<a[0] && a[0]<58 && 47<a[1] && a[1]<58){
                cntle=1, start=0, k=-1;
                chap=chap*100+(a[0]-'0')*10+(a[1]-'0'), read(fd,c,4);
                if(47<c[1] && c[1]<58 && c[2]==':')
                        clau=(c[1]-'0');
                else if(47<c[1] && c[1]<58 && 47<c[2] && c[2]<58 && c[3]==':')
                        clau=(c[1]-'0')*10+(c[2]-'0');
                else if(47<c[1] && c[1]<58 && 47<c[2] && c[2]<58 && 47<c[3] && c[3]<58)
                        clau=(c[1]-'0')*100+(c[2]-'0')*10+(c[3]-'0');
                lseek(fd,-1, SEEK_CUR);
        }
        else chap=0;
}

if(numOfChap<chap)
        numOfChap=chap, numOfClau+=pastClau;
```

- indexBuilder

inputFileNm 파일을 2byte씩 읽어들인다. 2byte씩 읽을 때 [장:절]이 나오는 경우의 수는 총 3가지로 나뉜다. 1) a[0]:숫자, a[1]:숫자, 2) a[0]:숫자, a[1]: ":", 3) a[0]: 공백 or 개행, a[1]: 숫자. 이 세가지 경우를 생각하고, 장의 자릿수, 절의 자릿수도 생각한다면 위와 같은 코드를 작성할 수 있다. 이 때 chap에는 현재의 장, clau에는 현재의 절을 저장한다.

```
        if(cntle==1){
                for(int i=0;i<2;i++){
                        if(a[i]>='A' && a[i]<='z'){
                                now=1, k++;
                                if(a[i]>='A' && a[i]<='Z') a[i]=a[i]+32;
                                word[m++]=a[i];
                                if(ok==1) {
                                        byte=k, ok=0;
                                        pastChap=chap, pastClau=clau;
                                }
                        }
                        if(a[i]=='(') {
                                if(kh==1 && k==1) k=0,kh=0;
                                else k++;
                        }
                        if(now==1){
                                if(a[i]=='\'' || a[i]=='-') word[m++]=a[i], k++, gh++;
                                else if(a[i]=='.'||a[i]==','||a[i]==':'||a[i]==';'||a[i]==')'||a[i]=='?'||a[i]=='!') k++, gh++;
                                else if(a[i]==' ' || a[i]=='\n'){
                                        k++;
                                        if(kh==1 && byte==2) byte=0, k=m+gh, kh=0;
                                        if(kh==1 && byte==0) kh=0;
                                        if(a[i]==' ' && i==0 && a[1]==' ') k++;
                                        else if(a[i]==' ' && i==1){
                                                read(fd,a,2);
                                                if(a[0]==' ') k++;
                                                lseek(fd,-2,SEEK_CUR);
                                        }
                                        insert(word, pastChap, pastClau, byte,m);
                                        m=0, ok=1, now=0, numOfWord++, gh=0;
                                }
                        }
                }
        }
}
numOfClau+=clau;
if(m>0){
        insert(word, pastChap, pastClau, byte,m);
        numOfWord++;
}
```

cntle=1의 뜻은 개행문자가 나온 후 장:절이 나온 라인이므로 이제 단어를 읽어들여도 된다는 의미이다. 여기서 영문자가 나올 때마다 word[30]에 저장해주고, 그 후에 공백이 나온다면 단어가 하나 완성되었다는 뜻이므로 insert하여 노드를 생성해준다. Insert 할 때는 문자가 처음 시작될 지점에서의 chap, clau으로 넘겨주어야 하므로 이 때 pastChap, pastClau를 사용하여 처음 바이트의 chap, clau를 저장한다. k는 한 라인에서 byte를 세는 기능을 한다. byte 변수의 경우 단어가 시작할 때 첫번째 영문자의 위치를 저장해주는 용도이고, 단어가 처음 시작될 때의 k를 이용하여 구한다. 장:절이 있는 라인에서 영문자가 시작되기도 전에 기호나 공백들을 byte에 포함시키면 안되므로, 영문자가 등장한 경우에만 셀 수 있도록 now 변수를 사용하였다.

```
struct node *horse;
ok=0;
char len[4], result[15];
for(int i=0;i<90000000;i++){
        horse=hashTable[i];
        if(hashTable[i]!=NULL){
                ok=1;
                while(horse!=NULL){
                        if(ok==1){
                                numOfIndex++;
                                write(ifd, horse->name, horse->wordLen);
                                write(ifd, ": ",2);
                                cn=intToChar(horse->count, len, 0);
                                write(ifd, len, cn);
                                write(ifd, ", ",2);
                                ok=0;
                        }
                        ch=intToChar(horse->chap,result,0);
                        result[ch]=':';
                        cl=intToChar(horse->clau,result,ch+1);
                        result[ch+cl+1]=':';
                        pl=intToChar(horse->place,result,ch+cl+2);
                        if(pl==0) pl=1, result[ch+cl+2]='0';
                        write(ifd,result, ch+cl+pl+2);

                        horse=horse->next;

                        if(horse!=NULL) write(ifd, ", ",2);
                }
                write(ifd, " \n",2);
        }
}
```

hashTable 90000000의 원소 중 NULL이 아닌 것만 write한다. node* horse을 이용하여 각 hashTable[i] 마다 그에 연결된 리스트들을 차례대로 출력해준다. 이 때 단어이름, 출현횟수는 한 번만 출력하고 장:절:위치만 단어가 등장할 때 마다 출력하도록 한다. 여기서 horse->name을 제외한 horse->count, horse->chap, horse->clau, horse->place는 정수이므로 intToChar 함수를 이용하여 result[15]에 문자열로 변환시켜 저장한다. 연결된 노드가 하나씩 읽힐 때마다 [장:절:위치]를 indexFileNm 파일에 write한다.

```
        num[0]='<';
        ch=intToChar(numOfChap,num,1);
        num[ch+1]='>', num[ch+2]='(';
        cl=intToChar(numOfClau,num,ch+3);
        num[ch+cl+3]=')', num[ch+cl+4]='{';
        id=intToChar(numOfIndex,num,ch+cl+5);
        num[ch+cl+id+5]='}', num[ch+cl+id+6]='[';
        wd=intToChar(numOfWord,num,ch+cl+id+7);
        num[ch+cl+id+wd+7]=']';
        write(ifd,num, ch+cl+id+wd+8);

        close(ifd);
        close(fd);
}
```

위에서 구한 numOfChap, numOfClau, numOfIndex, numOfWord를 intToChar 함수를 이용하여 num[30]에 문자열로 변환시켜 저장한다. 후에 indexPrinter에서 index file을 읽을 때 쉽게 구별하기 위해 기호<>,(),{},[]들을 문자열 사이에 넣어주었다. 그리고 indexFileNm 파일과 inputFileNm 파일을 close한다.

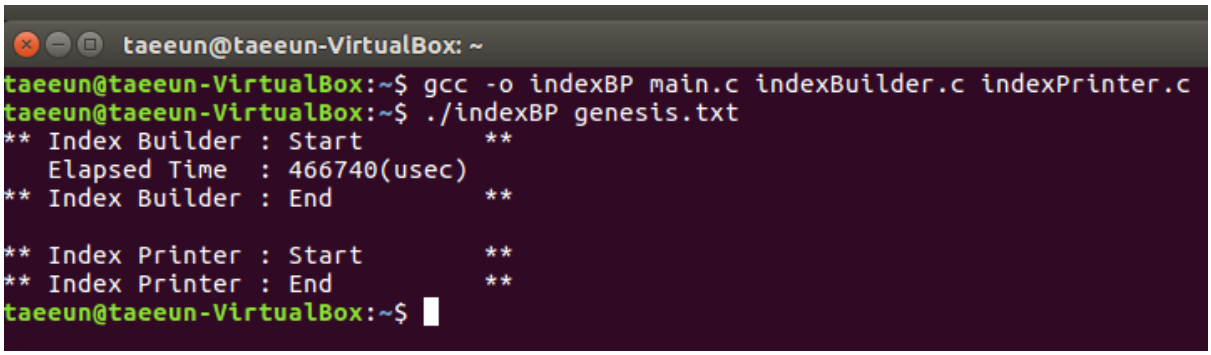
<IndexPrinter.c>

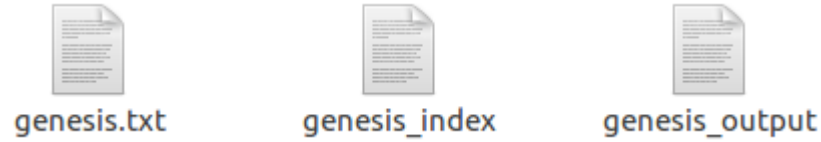
```
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

int intToChar(int num, char *c, int start);

void indexPrinter(const char* indexFileNm, const char* outputFileNm)
{
        /* Write your own C code */

        int ifd=open(indexFileNm,O_RDONLY);
        int ofd=open(outputFileNm, O_RDWR | O_CREAT, 0755);

        int len=0, i=0, m=0, ok=0, stop=0, size=0;
        char a[2], bibleNm[30], result[30];

        while(indexFileNm[i++]!='\0') len++;
        for(i=0;i<len-6;i++) bibleNm[i]=indexFileNm[i];


        write(ofd,bibleNm,len-6);
        write(ofd,": ",2);


        while(read(ifd,a,2)>0){
                for(i=0;i<2;i++){
                        if(a[i]=='<') ok=1;
                        else if(ok==1 && (a[i]=='('||a[i]=='{'||a[i]=='[')) result[m++]=' ';
                        else if(ok==1 && 47<a[i] && a[i]<58) result[m++]=a[i];
                        else if(ok==1 && a[i]==']') break;
                }
        }
        result[m]='\n';
        write(ofd, result, m+1);


        lseek(ifd,0,SEEK_SET);
```

```
        char *buf=(char *)malloc(sizeof(char)*1000);

        while(read(ifd,a,2)>0) {
                for(i=0;i<2;i++)
                        if(a[i]=='<') stop=1;
                        else buf[size++]=a[i];
                if(stop==1) break;
                if(size==1000){
                        write(ofd,buf,size);
                        size=0;
                }
        }
        write(ofd,buf,size);

        close(ifd);
        close(ofd);
}
```

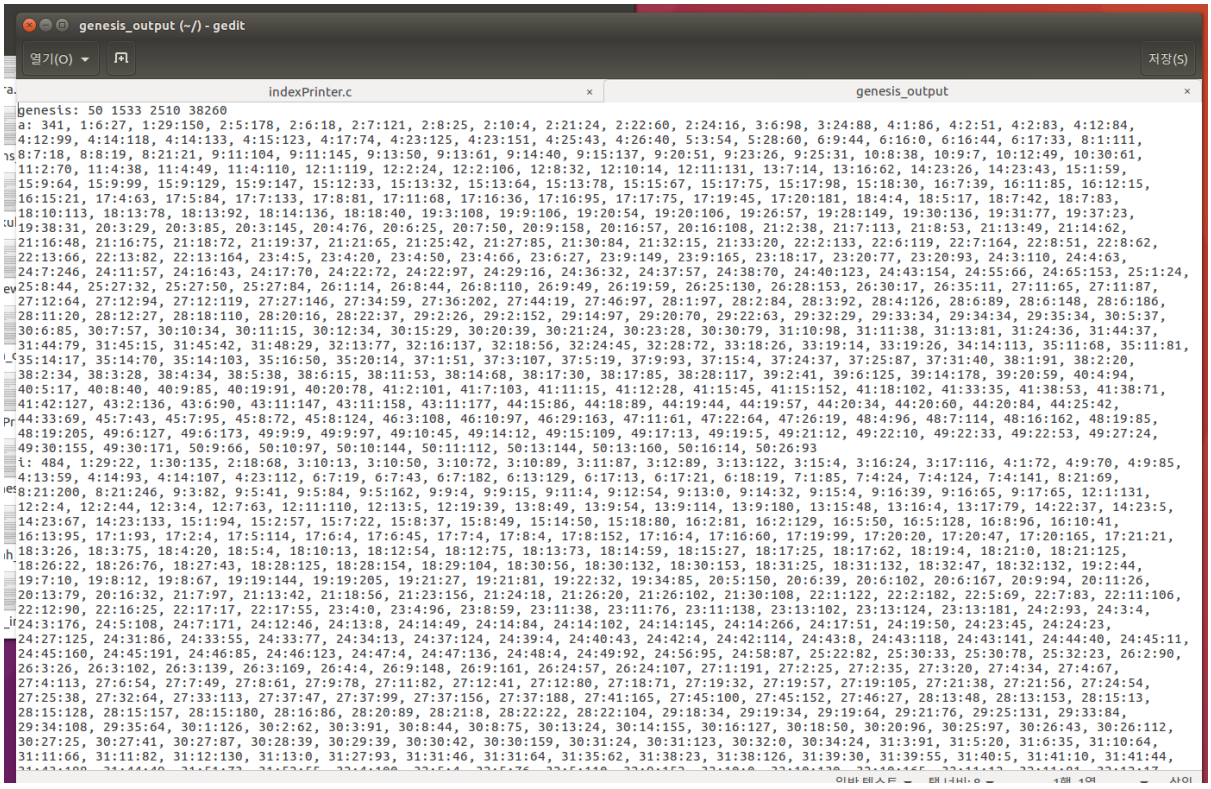
- indexPrinter: indexFileNm 파일을 읽어들인 후 ouputFileNm 파일에 정해진 형식으로 출력한다. 앞서 indexBuilder 함수에서 indexFileNm 파일의 가장 마지막에 "<장 수>(총 절 수){총 인덱스 수}[총 워드 수]" 형식으로 write해놓았기 때문에, 이 함수에서 indexFileNm을 읽어들일 때 이 기호들 (,),{,},<,>,[,]을 이용하여 char result[30]에 장 수, 총 절 수, 총 인덱스 수, 총 워드 수를 저장하여 outputFileNm 파일에 write한다. 또한 bible의 이름은 indexFileNm에서 _index 부분만 뺀 것을 char bibleNm[30]에 저장하여 outputFileNm 파일에 write한다. 각 단어마다 출현횟수, 장:절:위치를 출력하는 것은 indexFileNm에 적힌 것을 읽어 오기만 하면 된다. 여기서 Buffering을 이용하여 buf의 size가 1000이 될 때마다 outputFileNm 파일에 write하도록 하였다. 만약 <기호가 나온다면 이것은 모든 단어의 출현횟수, 장:절:위치를 읽었다는 것이므로 break로 while문을 종료시킨다. 그리고 마지막으로 남은 buf를 outputFileNm 파일에 write한다.

터미널에서 실행시킨 화면은 다음과 같다. 예시로 genesis.txt 파일을 사용하였다.





터미널에서 실행시킨 후, genesis.txt 파일 옆에 genesis_index 파일과 genesis_output 파일이 생성되었다.



genesis_output 파일 안의 내용이다. 첫 줄에는 bible의 이름과 장 수, 총 절 수, 총 인덱스 수, 총 워드 수가 출력되었다. 밑으로는 각 단어마다 출현횟수, 장:절:위치가 출력되었다.

(단어 sorting은 되어있지 않음)