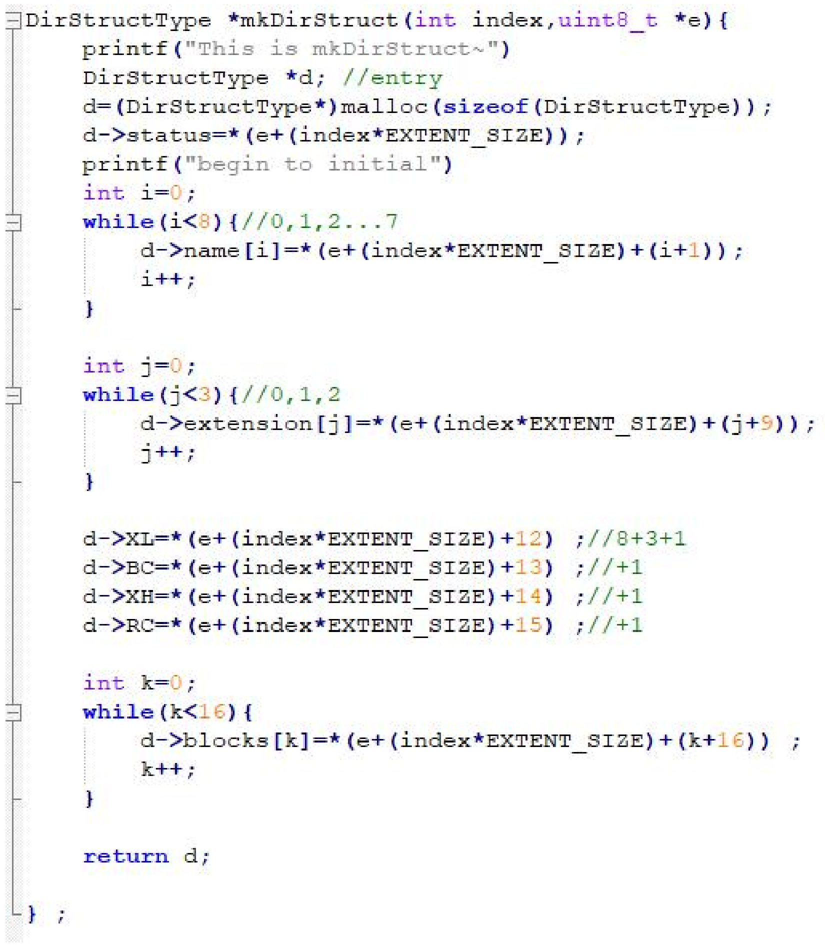
Project 4 cpmFS- A Simple File System Report

Gaoxiang Li gzl0034

Statement： I have finished the design and code of this project and the project compiled successfully. But when I try to run my project, here is a core dump error with my project. I have tried my best to debugging to solve this problem but it doesn't work.

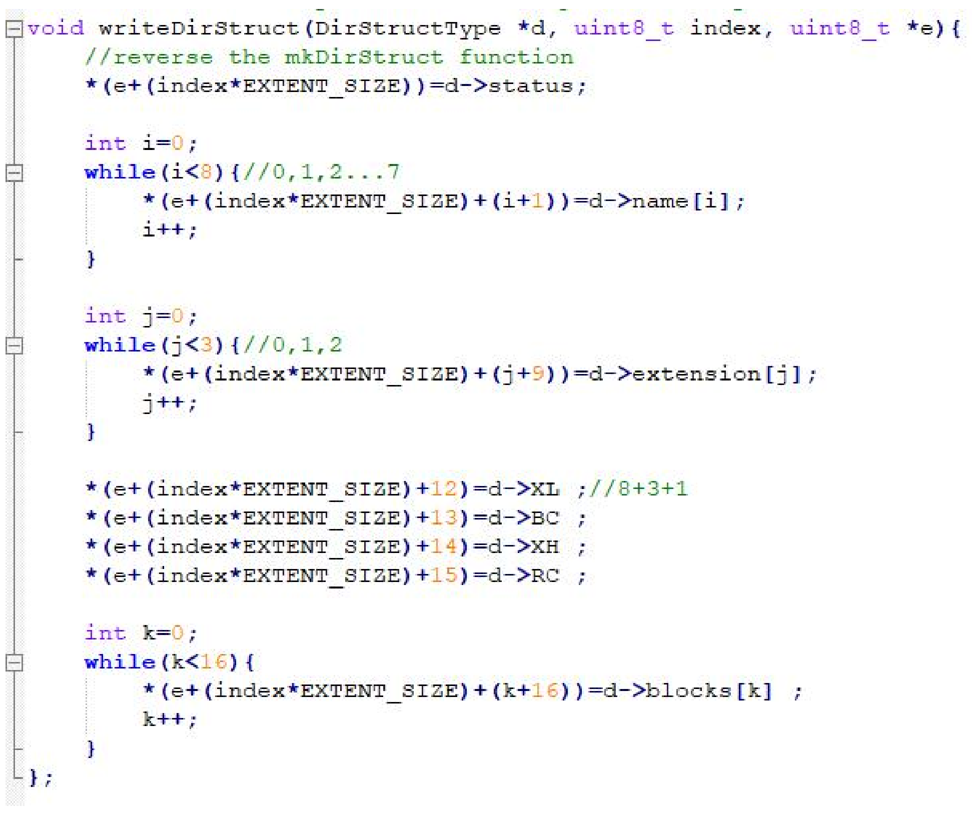
Design:

Function mkDirStruct()



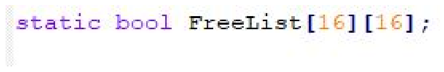
```
DirStructType *mkDirStruct(int index,uint8_t *e){
    printf("This is mkDirStruct~")
    DirStructType *d; //entry
    d=(DirStructType*)malloc(sizeof(DirStructType));
    d->status=*(e+(index*EXTENT_SIZE));
    printf("begin to initial")
    int i=0;
    while(i<8){//0,1,2...7
        d->name[i]=*(e+(index*EXTENT_SIZE)+(i+1));
        i++;
    }

    int j=0;
    while(j<3){//0,1,2
        d->extension[j]=*(e+(index*EXTENT_SIZE)+(j+9));
        j++;
    }

    d->XL=*(e+(index*EXTENT_SIZE)+12) ;//8+3+1
    d->BC=*(e+(index*EXTENT_SIZE)+13) ;//+1
    d->XH=*(e+(index*EXTENT_SIZE)+14) ;//+1
    d->RC=*(e+(index*EXTENT_SIZE)+15) ;//+1

    int k=0;
    while(k<16){
        d->blocks[k]=*(e+(index*EXTENT_SIZE)+(k+16)) ;
        k++;
    }

    return d;

} ;
```

Function writeDirStruct()

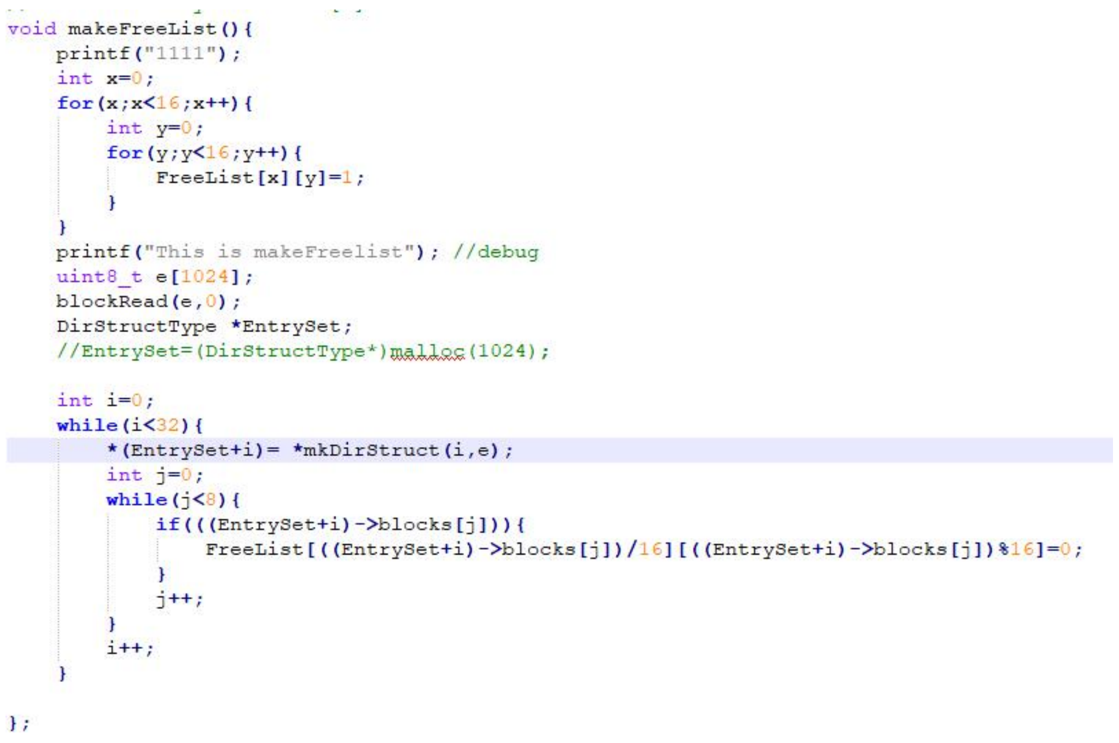


```c
void writeDirStruct(DirStructType *d, uint8_t index, uint8_t *e){
    //reverse the mkDirStruct function
    *(e+(index*EXTENT_SIZE))=d->status;

    int i=0;
    while(i<8){//0,1,2...7
        *(e+(index*EXTENT_SIZE)+(i+1))=d->name[i];
        i++;
    }

    int j=0;
    while(j<3){//0,1,2
        *(e+(index*EXTENT_SIZE)+(j+9))=d->extension[j];
        j++;
    }

    *(e+(index*EXTENT_SIZE)+12)=d->XL  ;//8+3+1
    *(e+(index*EXTENT_SIZE)+13)=d->BC  ;
    *(e+(index*EXTENT_SIZE)+14)=d->XH  ;
    *(e+(index*EXTENT_SIZE)+15)=d->RC  ;

    int k=0;
    while(k<16){
        *(e+(index*EXTENT_SIZE)+(k+16))=d->blocks[k]  ;
        k++;
    }
};
```

Function makeFreeList()

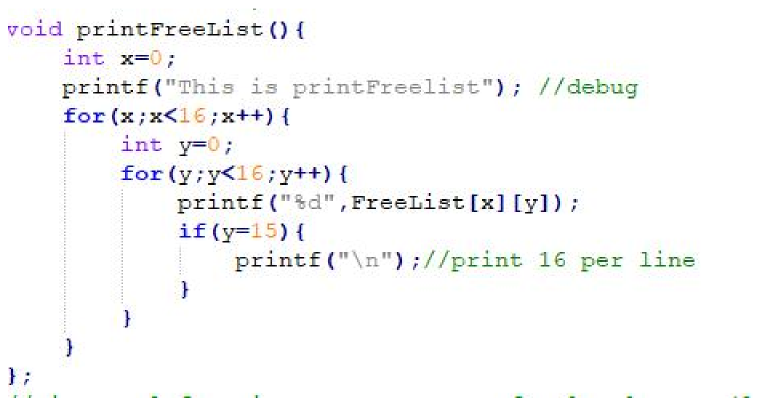


```c
static bool FreeList[16][16];
```



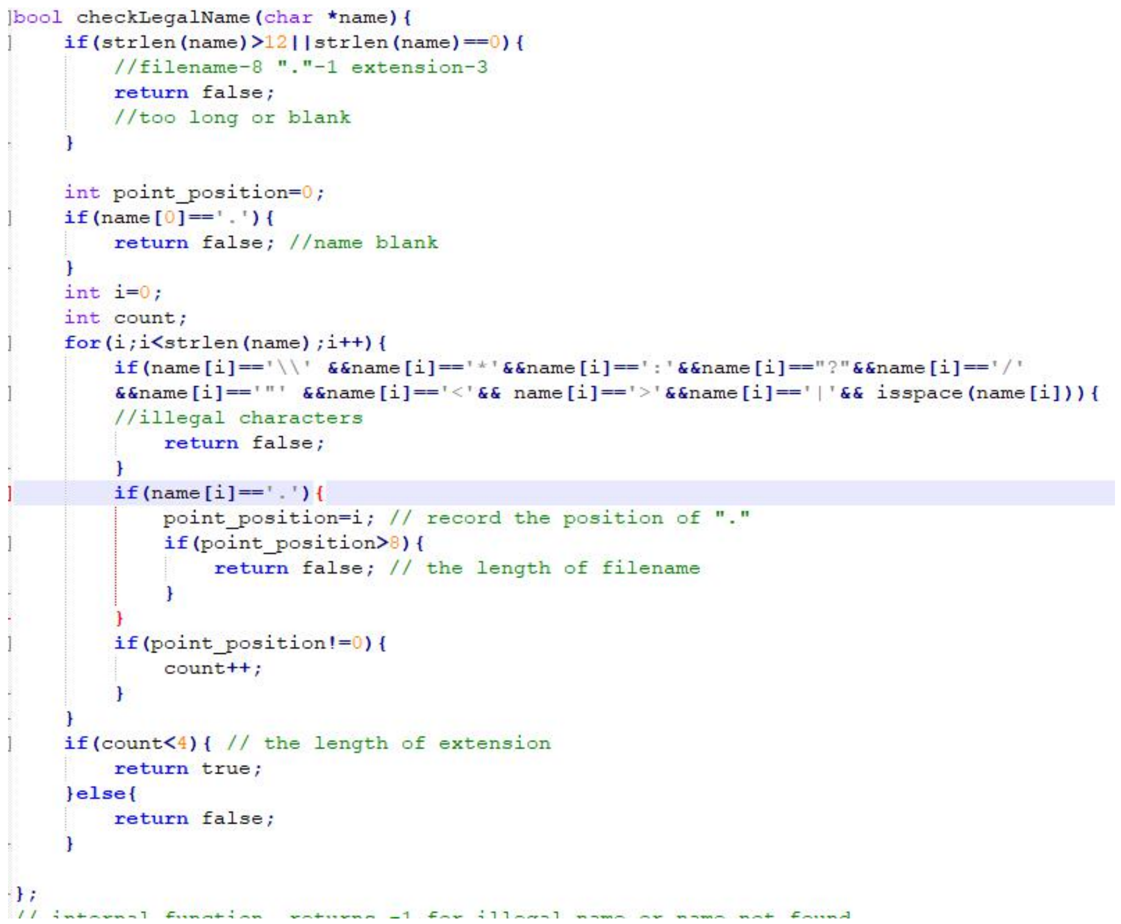
```c
void makeFreeList(){
    printf("1111");
    int x=0;
    for(x;x<16;x++){
        int y=0;
        for(y;y<16;y++){
            FreeList[x][y]=1;
        }
    }
    printf("This is makeFreelist"); //debug
    uint8_t e[1024];
    blockRead(e,0);
    DirStructType *EntrySet;
    //EntrySet=(DirStructType*)malloc(1024);

    int i=0;
    while(i<32){
        *(EntrySet+i)= *mkDirStruct(i,e);
        int j=0;
        while(j<8){
            if(((EntrySet+i)->blocks[j])){
                FreeList[((EntrySet+i)->blocks[j])/16][((EntrySet+i)->blocks[j])%16]=0;
            }
            j++;
        }
        i++;
    }

};
```

Function printFreeList()



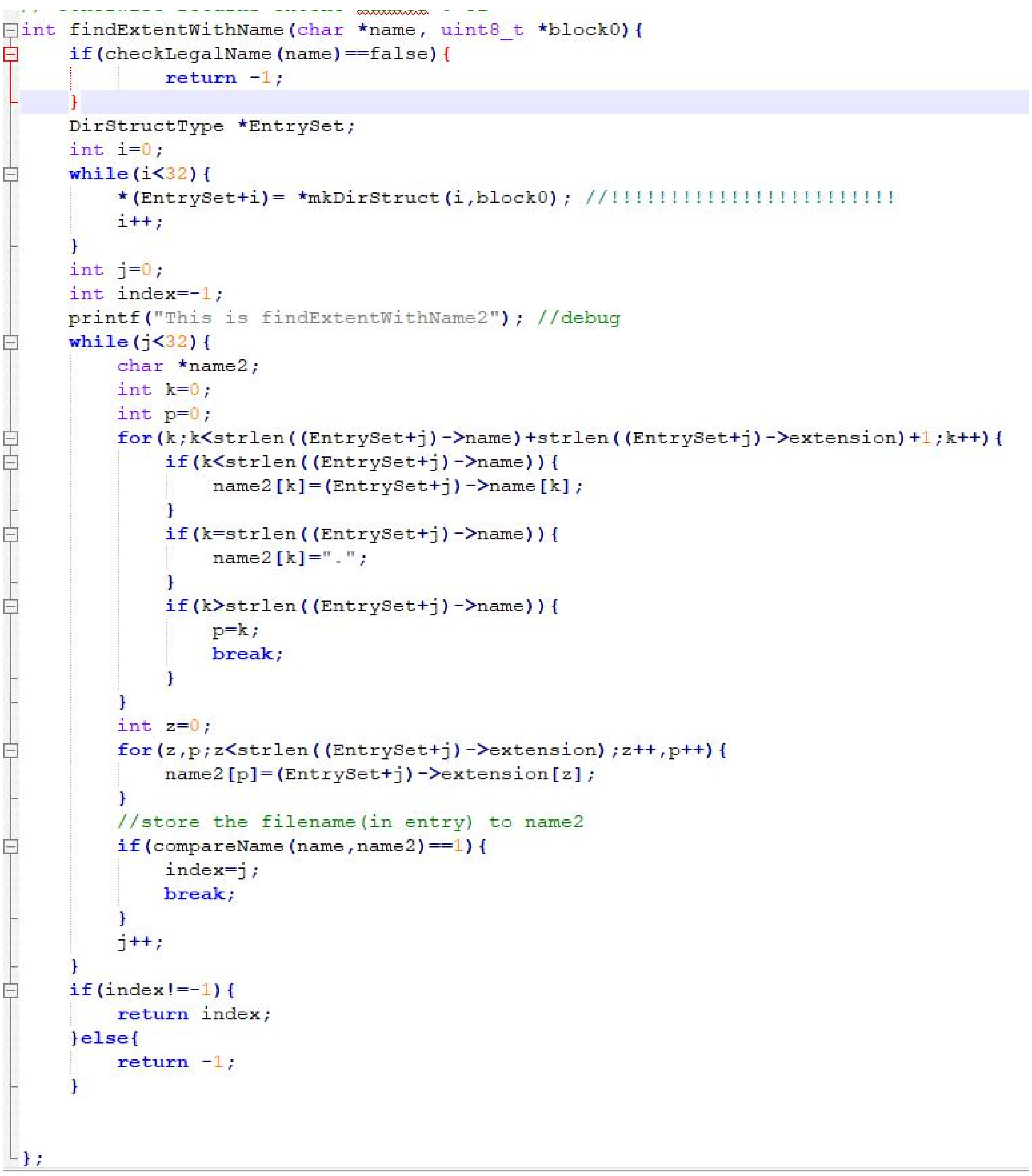
```c
void printFreeList(){
    int x=0;
    printf("This is printFreelist"); //debug
    for(x;x<16;x++){
        int y=0;
        for(y;y<16;y++){
            printf("%d",FreeList[x][y]);
            if(y=15){
                printf("\n");//print 16 per line
            }
        }
    }
};
```

Function checkLegalName()



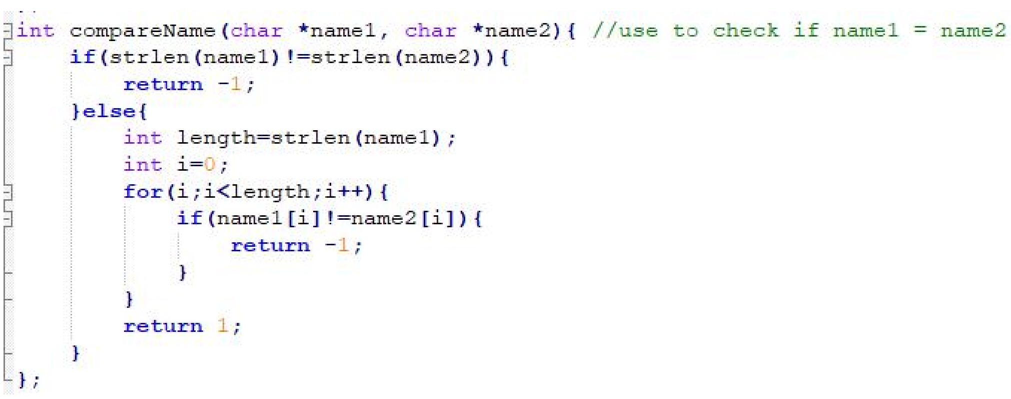
```c
bool checkLegalName(char *name){
    if(strlen(name)>12||strlen(name)==0){
        //filename-8 "."-1 extension-3
        return false;
        //too long or blank
    }

    int point_position=0;
    if(name[0]=='.'){
        return false; //name blank
    }
    int i=0;
    int count;
    for(i;i<strlen(name);i++){
        if(name[i]=='\\' &&name[i]=='*'&&name[i]==':'&&name[i]=="?"&&name[i]=='/'
        &&name[i]=='"' &&name[i]=='<'&& name[i]=='>'&&name[i]=='|'&& isspace(name[i])){
        //illegal characters
            return false;
        }
        if(name[i]=='.'){
            point_position=i; // record the position of "."
            if(point_position>8){
                return false; // the length of filename
            }
        }
        if(point_position!=0){
            count++;
        }
    }
    if(count<4){ // the length of extension
        return true;
    }else{
        return false;
    }

};
// internal function, returns -1 for illegal name or name not found
```

Function FindExtentWithName()



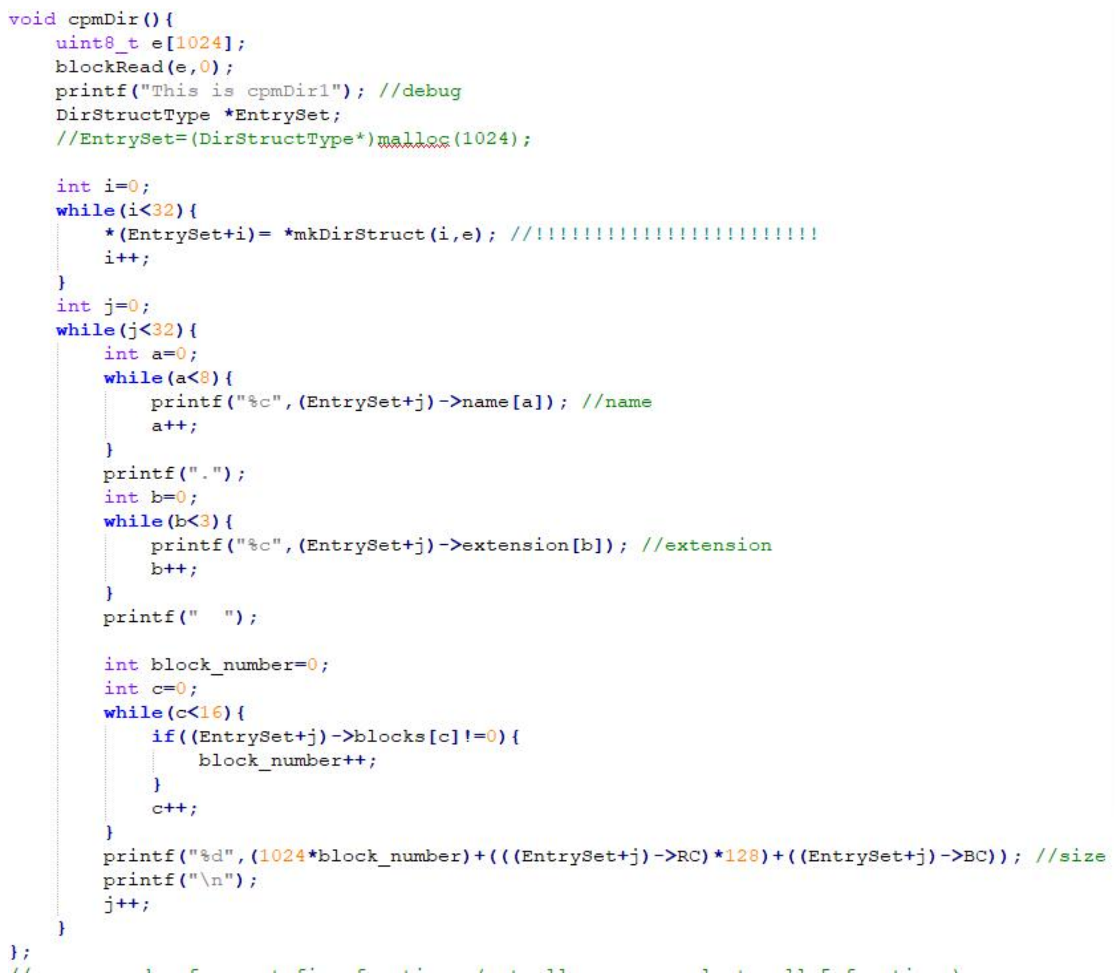
```
int findExtentWithName(char *name, uint8_t *block0){
    if(checkLegalName(name)==false){
            return -1;
    }
    DirStructType *EntrySet;
    int i=0;
    while(i<32){
        *(EntrySet+i)= *mkDirStruct(i,block0); //!!!!!!!!!!!!!!!!!!!!!!!!!
        i++;
    }
    int j=0;
    int index=-1;
    printf("This is findExtentWithName2"); //debug
    while(j<32){
        char *name2;
        int k=0;
        int p=0;
        for(k;k<strlen((EntrySet+j)->name)+strlen((EntrySet+j)->extension)+1;k++){
            if(k<strlen((EntrySet+j)->name)){
                name2[k]=(EntrySet+j)->name[k];
            }
            if(k=strlen((EntrySet+j)->name)){
                name2[k]=".";
            }
            if(k>strlen((EntrySet+j)->name)){
                p=k;
                break;
            }
        }
        int z=0;
        for(z,p;z<strlen((EntrySet+j)->extension);z++,p++){
            name2[p]=(EntrySet+j)->extension[z];
        }
        //store the filename(in entry) to name2
        if(compareName(name,name2)==1){
            index=j;
            break;
        }
        j++;
    }
    if(index!=-1){
        return index;
    }else{
        return -1;
    }


};
```

Function compareName()   (use to compare two chars)



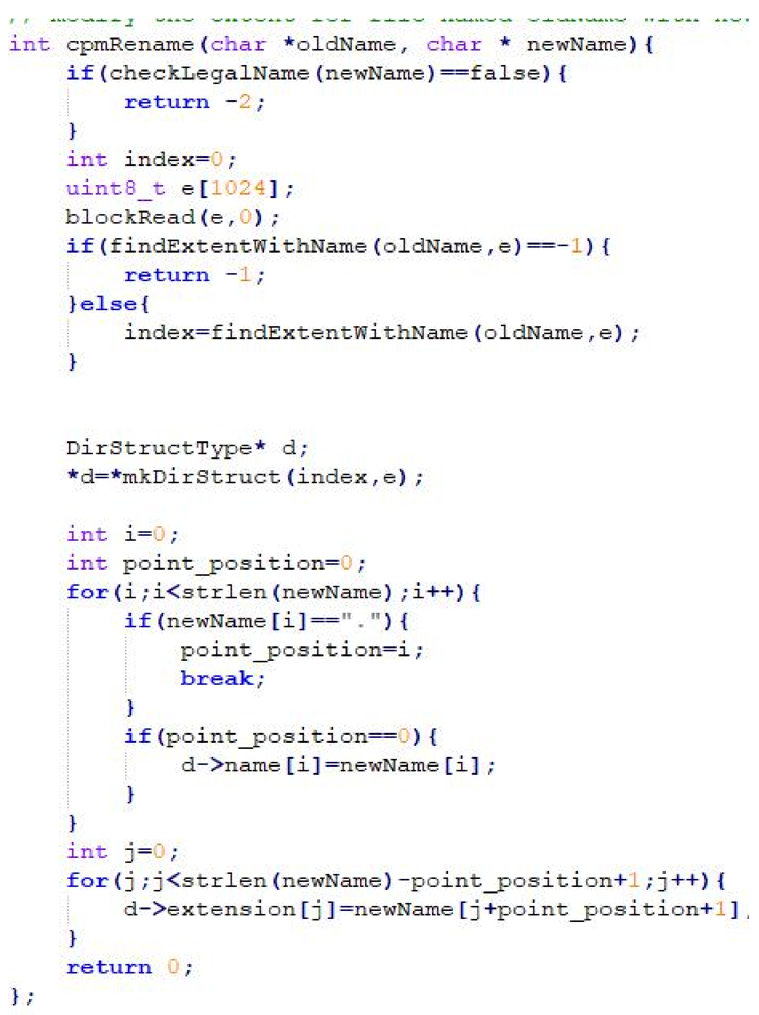
```
int compareName(char *name1, char *name2){ //use to check if name1 = name2
    if(strlen(name1)!=strlen(name2)){
        return -1;
    }else{
        int length=strlen(name1);
        int i=0;
        for(i;i<length;i++){
            if(name1[i]!=name2[i]){
                return -1;
            }
        }
        return 1;
    }
};
```

Function cpmDir()



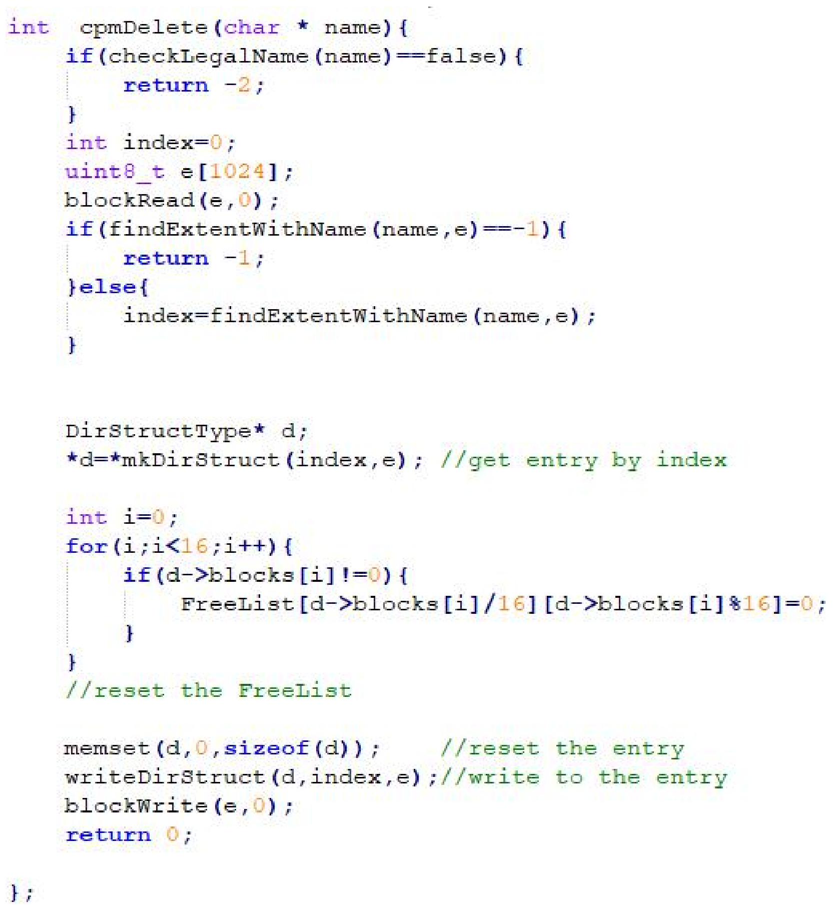
```c
void cpmDir(){
    uint8_t e[1024];
    blockRead(e,0);
    printf("This is cpmDir1"); //debug
    DirStructType *EntrySet;
    //EntrySet=(DirStructType*)malloc(1024);

    int i=0;
    while(i<32){
        *(EntrySet+i)= *mkDirStruct(i,e); //!!!!!!!!!!!!!!!!!!!!!!!!!
        i++;
    }
    int j=0;
    while(j<32){
        int a=0;
        while(a<8){
            printf("%c",(EntrySet+j)->name[a]); //name
            a++;
        }
        printf(".");
        int b=0;
        while(b<3){
            printf("%c",(EntrySet+j)->extension[b]); //extension
            b++;
        }
        printf("  ");

        int block_number=0;
        int c=0;
        while(c<16){
            if((EntrySet+j)->blocks[c]!=0){
                block_number++;
            }
            c++;
        }
        printf("%d",(1024*block_number)+(((EntrySet+j)->RC)*128)+((EntrySet+j)->BC)); //size
        printf("\n");
        j++;
    }
};
```

Function cpmRename()



```
// modify the extent for file named oldName with ne
int cpmRename(char *oldName, char * newName){
    if(checkLegalName(newName)==false){
        return -2;
    }
    int index=0;
    uint8_t e[1024];
    blockRead(e,0);
    if(findExtentWithName(oldName,e)==-1){
        return -1;
    }else{
        index=findExtentWithName(oldName,e);
    }


    DirStructType* d;
    *d=*mkDirStruct(index,e);

    int i=0;
    int point_position=0;
    for(i;i<strlen(newName);i++){
        if(newName[i]=="."){
            point_position=i;
            break;
        }
        if(point_position==0){
            d->name[i]=newName[i];
        }
    }
    int j=0;
    for(j;j<strlen(newName)-point_position+1;j++){
        d->extension[j]=newName[j+point_position+1]
    }
    return 0;
};
```

Function cpmDelete()



```
int  cpmDelete(char * name){
    if(checkLegalName(name)==false){
        return -2;
    }
    int index=0;
    uint8_t e[1024];
    blockRead(e,0);
    if(findExtentWithName(name,e)==-1){
        return -1;
    }else{
        index=findExtentWithName(name,e);
    }


    DirStructType* d;
    *d=*mkDirStruct(index,e); //get entry by index

    int i=0;
    for(i;i<16;i++){
        if(d->blocks[i]!=0){
            FreeList[d->blocks[i]/16][d->blocks[i]%16]=0;
        }
    }
    //reset the FreeList

    memset(d,0,sizeof(d));    //reset the entry
    writeDirStruct(d,index,e);//write to the entry
    blockWrite(e,0);
    return 0;

};
```