

Programlama Dilleri

Proje 2

Sercan Bayram 05170000038

Oğuz Türk 05170000079

## Gerçekleştirilen Platform ve Sürüm Adı:

Codeblocks 20.03

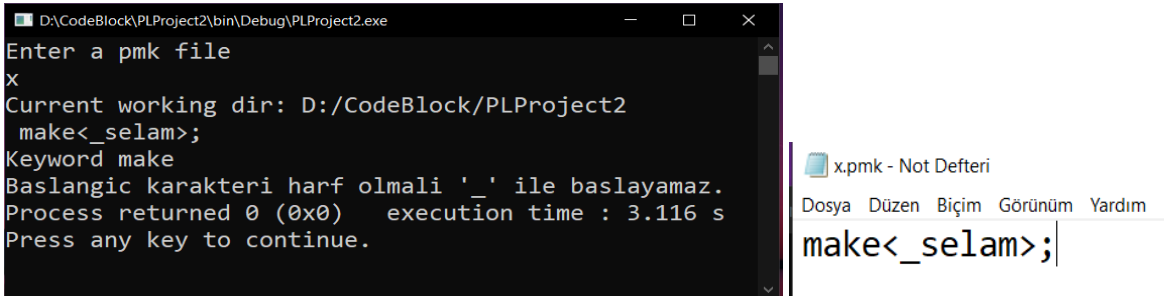
## Programın Tanımı:

Path\_maker dizin ağaçları oluşturmak için temel bir scripting dilidir. (Dil kuralları aşağıdadır) Bizden path\_maker için temel bir interpreter istenmektedir.

**Data Types:** Tek veri tipi Path'dir. Path sabitleri, formda yazılan **görelî dizin yolu ifadeleridir:** <dir1/dir2/dir3> dir1,dir2,dir3 klasör adlarıdır.

- Dizin adları bir harfle başlar (büyük veya küçük harf) ve herhangi bir harf, rakam ve alt çizgi karakteri birleşiminden oluşur (yalnızca). (Noktalama karakterlerine izin verilmez. Boş karakterlere de izin verilmez.) Dizin adları büyük / küçük harfe duyarlı değildir, bu nedenle <AA> ve <aa> temel olarak aynıdır. (çünkü çoğu işletim sisteminin politikası budur)

## Örnek Ekran Görüntüsü:



The screenshot shows a CodeBlocks IDE with a console window and a Notepad file. The console window displays the following text:

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
Enter a pmk file
x
Current working dir: D:/CodeBlock/PLProject2
make<_selam>;
Keyword make
Baslangic karakteri harf olmalı '_' ile baslayamaz.
Process returned 0 (0x0)   execution time : 3.116 s
Press any key to continue.
```

The Notepad file, titled 'x.pmk - Not Defteri', contains the following text:

```
make<_selam>;
```

**Operatör \***: Yalnızca path ifadelerinin başlangıcında kullanılabilir.

<hi / \* / there> öğesine izin verilmez.

- Operatör / herhangi bir yolun başında veya sonunda kullanılamaz. Bu yüzden </ hi / there> öğesine izin verilmez. <Hi / there /> öğesine de izin verilmez.
- Bir yol ifadesindeki boşluklar yok sayılır (dizin adında (izin verilmez) bulunmadıkları sürece), bu nedenle <\* / \* / dizinim> sorun olmaz.

### Örnek Ekran Görüntüsü:

x.pmk - Not Defteri  
Dosya Düzen Biçim Görünüm Yardım  
make</hi/there>;

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
Enter a pmk file
x
Current working dir: D:/CodeBlock/PLProject2
make</hi/there>;
Keyword make
'(/hi/there) is not allowed!'

Process returned 1 (0x1)    execution time : 2.440 s
```

x.pmk - Not Defteri  
Dosya Düzen Biçim Görünüm Yardım  
make<hi/\*/there>;

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
Enter a pmk file
x
Current working dir: D:/CodeBlock/PLProject2
make<hi/*/there>;
Keyword make
'(hi/*/there) is not allowed!'

Process returned 1 (0x1)    execution time : 2.250 s
Press any key to continue.
```

**Temel Komutlar:** İki temel komut sadece “make” ve “go” dur.

- Sadece myDirectoryPath içindeki dizinleri oluşturur. Path zaten varsa hiçbir şey yapmaz (ancak bir uyarı mesajı verir). Path kısmen mevcutsa, yolu tamamlar.
- “Make” geçerli (çalışma) dizini değiştirmez. (Go komutu ile yaptığımız şey budur).

### Örnek make kullanımları ve oluşan dosyalar:

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
Enter a pmk file
X
Current working dir: D:/CodeBlock/PLProject2
make <myDirectoryPath>;

Keyword make
Directory created
D:\CodeBlock\PLProject2\myDirectoryPath
Current Directory: D:\CodeBlock\PLProject2
make <beste/ammam>;

Keyword make
Directory created
D:\CodeBlock\PLProject2\beste
Directory created
D:\CodeBlock\PLProject2\beste\ammam
Current Directory: D:\CodeBlock\PLProject2
make <beste/ammam/ensar>;

Keyword make
Directory is created before!
Directory is created before!
Directory created
D:\CodeBlock\PLProject2\beste\ammam\ensar
Current Directory: D:\CodeBlock\PLProject2
make <*/projectx>;

Keyword make
D:\CodeBlock
Directory created
D:\CodeBlock\projectx
Current Directory: D:\CodeBlock\PLProject2

Process returned 0 (0x0)   execution time : 2.230 s
Press any key to continue.
```

```
x.pmk - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
make <myDirectoryPath>;
make <beste/ammam>;
make <beste/ammam/ensar>;
make <*/projectx>;
```

## Oluşan dosyalar:

Bu bilgisayar > Depo (D:) > CodeBlock > PLProject2			
Ad	Değiştirme tarihi	Tür	Boyut
beste	16.06.2020 18:04	Dosya klasörü	
bin	13.06.2020 01:19	Dosya klasörü	
myDirectoryPath	16.06.2020 18:04	Dosya klasörü	
obj	13.06.2020 01:19	Dosya klasörü	
main.c	16.06.2020 15:22	C Source File	17 KB
PLProject2.cbp	10.06.2020 01:42	project file	2 KB
PLProject2.depend	16.06.2020 17:40	DEPEND Dosyası	1 KB
PLProject2.layout	16.06.2020 15:22	LAYOUT Dosyası	1 KB
x.pmk	16.06.2020 18:04	PMK Dosyası	1 KB

Depo (D:) > CodeBlock > PLProject2 > beste > ammar		
Ad	Değiştirme tarihi	Tür
ensar	16.06.2020 18:04	Dosya klasörü

Bu bilgisayar > Depo (D:) > CodeBlock		
Ad	Değiştirme tarihi	Tür
1	4.05.2020 01:37	Dosya klasörü
hiçbişi	15.06.2020 00:27	Dosya klasörü
hw3	11.05.2020 22:44	Dosya klasörü
nb	16.06.2020 15:18	Dosya klasörü
pascal	26.04.2020 21:00	Dosya klasörü
pascal1	26.04.2020 21:19	Dosya klasörü
paskal	26.04.2020 21:41	Dosya klasörü
paskalIIIIII	26.04.2020 21:57	Dosya klasörü
PLProje1	15.05.2020 22:32	Dosya klasörü
PLProject2	16.06.2020 18:04	Dosya klasörü
PRoje1	10.05.2020 03:01	Dosya klasörü
project1	13.06.2020 23:00	Dosya klasörü
projectx	16.06.2020 18:04	Dosya klasörü

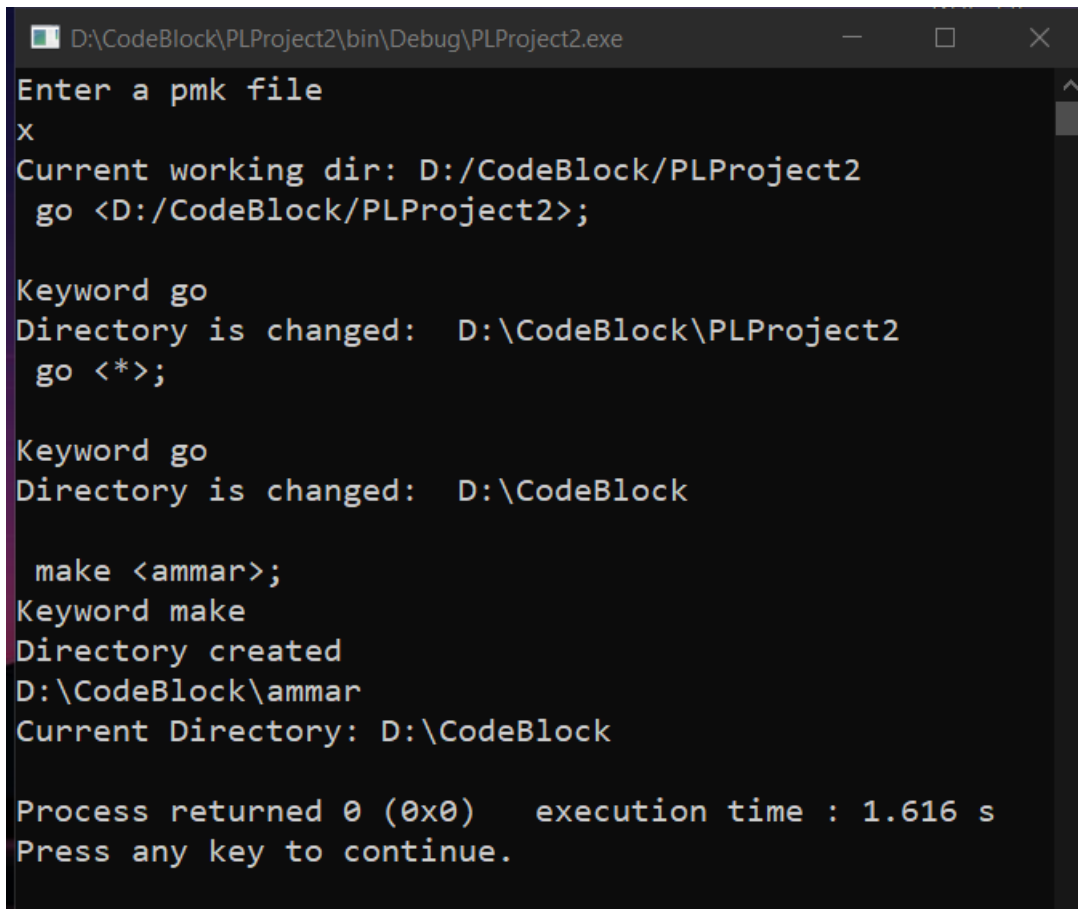
**"Go":** Sadece geçerli dizini değiştirir.

- Syntax: go <myPathExpression>;
- Path yoksa, go hiçbir şey yapmaz. (Bir hata mesajı verir, ancak yürütmeden çıkmaz) Kısmen bir yolu izlemez. Herhangi bir yolun kısmi varlığı varolmamak olarak kabul edilir.

### Örnek go kullanımları ve oluşan dosyalar:



```
x.pmk - Not Defteri
Dosya  Düzen  Biçim  Görünüm  Yardım
go <D:/CodeBlock/PLProject2>;
go <*>;
make <ammar>;
```



```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
Enter a pmk file
x
Current working dir: D:/CodeBlock/PLProject2
go <D:/CodeBlock/PLProject2>;

Keyword go
Directory is changed: D:\CodeBlock\PLProject2
go <*>;

Keyword go
Directory is changed: D:\CodeBlock

make <ammar>;
Keyword make
Directory created
D:\CodeBlock\ammar
Current Directory: D:\CodeBlock

Process returned 0 (0x0) execution time : 1.616 s
Press any key to continue.
```

## Oluşan dosya:

Bu bilgisayar > Depo (D:) > CodeBlock			
Ad	Değiştirme tarihi	Tür	Boyut
1	4.05.2020 01:37	Dosya klasörü	
ammar	16.06.2020 18:27	Dosya klasörü	

**Kontrol Yapıları:** Bir "if" deyimi ve benzer bir "ifnot" deyimi vardır.

- if <path\_expression> komutu

komutun temel bir komut veya bir blok olabileceği bu cümlelerin temel biçimidir. "if" yan tümcesi, geçerli dizinden <path\_expression> yolu varsa komutu yürütür.

- "if" geçerli dizini değiştirmez.
- "ifnot" cümlesi tamamen aynı yapıya sahiptir.
- ifnot <path\_expression> komutu ancak <path\_expression> yolu yoksa çalışır.
- "if" veya "ifnot" blokları veya bloksuz kullanılabilir bloksuz komut olarak kullanılırsa bir alt satırdaki komut if/ifnot 'a bağlı çalışır.

**Bloklar:** Bir komut temel bir komut olabilir ("make" veya "go"), fakat aynı zamanda bir blok olabilir. Blok, {} set köşeli parantez içindeki kod satırlarının listesidir. Bloklar ayrıca iç içe de yerleştirilebilir.

**Satır sonu karakteri:** Yalnızca "make" ve "go" komutları, satır sonu karakteri gerektirir ve ';' (noktalı virgül)

**Anahtar Kelimeler:** Anahtar kelimeler büyük / küçük harfe duyarlıdır ve tümü küçük harflidir: if,make,go,ifnot.

**Semboller:** <,>, {}, /, \*

## Örnek if kullanımları ve oluşan dosyalar:

```
x.pmk - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
if<*>
{ go <*>;
make <data/doctors>;
if<D:/CodeBlock/boylebirdosyayok>
go <D:/CodeBlock/PLProject2>;
}
```

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
Enter a pmk file
x
Current working dir: D:/CodeBlock/PLProject2
if<*>

Keyword if
Current Directory: D:\CodeBlock\PLProject2
{ go <*>;

Keyword go
Directory is changed: D:\CodeBlock

    make <data/doctors>;

Keyword make
Directory created
D:\CodeBlock\data
Directory created
D:\CodeBlock\data\doctors
Current Directory: D:\CodeBlock
    if<D:/CodeBlock/boylebirdosyayok>

Keyword if
    go <D:/CodeBlock/PLProject2>;

Keyword go
Directory can't be changed..
}
```

Bu bilgisayar > Depo (D:) > CodeBlock > data			
Ad	Değiştirme tarihi	Tür	Boyut
doctors	16.06.2020 20:20	Dosya klasörü	



## Durumların ortak kullanımı ve oluşan dosyalar:

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
Current working dir: D:/CodeBlock/PLProject2
if<D:/CodeBlock/PLProject2>

Keyword if
  make<ser/sercan/oguz>;

Keyword make
Directory created
D:\CodeBlock\PLProject2\ser
Directory created
D:\CodeBlock\PLProject2\ser\sercan
Directory created
D:\CodeBlock\PLProject2\ser\sercan\oguz
Current Directory: D:\CodeBlock\PLProject2
  if<*>{

Keyword if
Current Directory: D:\CodeBlock\PLProject2
  go<*>;

Keyword go
Directory is changed:  D:\CodeBlock

  make<hello/deneme>;

Keyword make
Directory created
D:\CodeBlock\hello
Directory created
D:\CodeBlock\hello\deneme
Current Directory: D:\CodeBlock
  if<D:/CodeBlock>

Keyword if
  go<D:/CodeBlock/PLProject2/ser>;}
```

```
x.pmk - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
if<D:/CodeBlock/PLProject2>
make<ser/sercan/oguz>;
if<*>{
go<*>;
make<hello/deneme>;
if<D:/CodeBlock>
go<D:/CodeBlock/PLProject2/ser>;}
go<D:/CodeBlock>;
if<*>
make<pl/great/again>;|
ifnot<*>{
go<D:/CodeBlock/PLProject2>;
}
make<data/doctors>;
ifnot<yok/yok>
make<data/doctors>;
```

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe

Keyword if
go<D:/CodeBlock/PLProject2/ser>;}

Keyword go
Directory is changed: D:\CodeBlock\PLProject2\ser
go<D:/CodeBlock>;

Keyword go
Directory is changed: D:\CodeBlock
if<*>

Keyword if
Current Directory: D:\CodeBlock
make<pl/great/again>;

ifnot<*>{

Keyword: ifnot
Current Directory: D:\CodeBlock
go<D:/CodeBlock/PLProject2>;

Keyword go
Directory is changed: D:\CodeBlock\PLProject2
}

make<data/doctors>;

Keyword make
Directory created
D:\CodeBlock\PLProject2\data
Directory created
D:\CodeBlock\PLProject2\data\doctors
Current Directory: D:\CodeBlock\PLProject2
ifnot<yok/yok>

Keyword: ifnot
```

```
D:\CodeBlock\PLProject2\bin\Debug\PLProject2.exe
}

make<data/doctors>;

Keyword make
Directory created
D:\CodeBlock\PLProject2\data
Directory created
D:\CodeBlock\PLProject2\data\doctors
Current Directory: D:\CodeBlock\PLProject2
ifnot<yok/yok>

Keyword: ifnot
make<data/doctors>;

Keyword make
Directory is created before!
Directory is created before!
Current Directory: D:\CodeBlock\PLProject2

Process returned 0 (0x0)   execution time : 1.600 s
Press any key to continue.
```

3

### Oluşan dosyalar:

Bu bilgisayar > Depo (D:) > CodeBlock > PLProject2 > ser > sercan				
Ad	Değiştirme tarihi	Tür	Boyut	
oguz	16.06.2020 20:34	Dosya klasörü		

Bu bilgisayar > Depo (D:) > CodeBlock > hello				
Ad	Değiştirme tarihi	Tür	Boyut	
deneme	16.06.2020 20:34	Dosya klasörü		

Bu bilgisayar > Depo (D:) > CodeBlock > PLProject2 > data				
Ad	Değiştirme tarihi	Tür	Boyut	
doctors	16.06.2020 20:34	Dosya klasörü		

## Kaynak Kod:

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <stdbool.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include <conio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
```

```
char* substr(const char *src, int m, int n)
{
    int len = n - m;                                /* keywordleri ve pathleri
                                                    satir dizisinden almiza yarayan substring fonksiyonumuz... */

    char *dest = (char*)malloc(sizeof(char) * (len + 1));

    for (int i = m; i < n && (*(src + i) != '\0'); i++)
    {
        *dest = *(src + i);
        dest++;
    }

    *dest = '\0';

    return dest - len;
}
```

```
int main()
{
    char filename[50];
    char *test;

    printf("Enter a pmk file\n");                //kullanicidan pmk ismi aliyoruz.
    scanf("%s",&filename);
    char * new_str ;
    new_str = malloc(strlen(filename)+4);
    new_str[0] = '\0';
    strcat(new_str,filename);
    strcat(new_str,".pmk");
    //char changing[] ;
    char listee [10][50];
    FILE *fp;
    int bufferLength = 255;                        //identiflerimiz.
    char satir[bufferLength];
    int substringlength=0;
```

```

char* subs;
char* dest;
char* path;
char slash[10]="/";
int check;
int compare1;
int compare2;
int compare3;
bool yildizliif=true;
bool yildizliifnot=true;
int ifde_mi=0;
int ifnot_mi=0;
bool flag1=false;
bool flag2=false;
bool icicemi=false;
int satirlength=0;
fp = fopen(new_str, "r");
char cwd[PATH_MAX];
char kopyacwd[PATH_MAX];
char kopyaacwd[PATH_MAX];

if (getcwd(cwd, sizeof(cwd)) != NULL) {
    //printf("Current working dir: %s\n", cwd);
    int i=0;
    while(cwd[i]!='\0')
    {
        if(cwd[i]=='\\')
        {
            cwd[i]='/';
        }
        i++;
    }
    printf("Current working dir: %s\n", cwd);    //current dir basiliyor.
} else {
    perror("getcwd() error");
    return 1;
}
if ((fp) == NULL){
printf("Dosya okumada hata var");
}
else{
while(fgets(satir, bufferLength, fp)) {    //bu dongude dosyayi satir satir okuyoruz...
    int pathlength=0;
    bool hatali=false;

    printf(" %s\n", satir );

    int i;

    for(i=0;i<strlen(satir);++i){
        int result;
        int begin=0;
        if(isalpha(satir[i]))

```

```

{ begin=i;

while(isalpha(satir[i]) || satir[i]=='_' || isdigit(satir[i]) || satir[i]==' '){ //keywordlarimizi bulmak için
//kullanimiz dongu.

    substrlength++;
    i++;
}

dest = substr(satir, begin, begin+substrlength); //keywordlar için substring fonksiyonumuz kullanilir.
substrlength=0;

begin=0;

}

if(yildizliif&&yildizliifnot){ //keywordlarin if ifnot durumlarinda kontrolu bu iflerle saglanmistir.
if((ifde_mi!=2 || (ifde_mi==1&&flag1==true)) || (ifnot_mi!=2 || (ifnot_mi==1&&flag1==true))){
if(strcmp(dest,"go")==0 || strcmp(dest,"go")==0){
    printf("Keyword %s\n",dest);
    if(ifde_mi==0&&ifnot_mi==2){
        printf("Directory can't be changed..\n");
        continue;
    }
    if(ifde_mi==2&&ifnot_mi==0){
        printf("Directory can't be changed..\n");
        continue;
    }
}

if(satir[i]=='<'){ //bu dongude
    i++; //o satirdaki path alinir.
    begin=i;

    while(satir[i]!='>'){

        substrlength++;
        i++;

    }

    path=substr(satir, begin, begin+substrlength);
    substrlength=0;

    pathlength=strlen(path);

    if(path[0]=='/' || path[pathlength-1]=='/'){ /* hatali yazimlarda programi exit yapiyoruz.*/

        printf("(hi/there) is not allowed!\n");

```

```

        exit(1);

    }
    int yildizmi;
    yildizmi=strcmp(path,"*");
    if(yildizmi==0){          /* 'go <*>' komutunun ust dizine gitmesini saglayan kod.*/

        chdir("../");
        printf("Directory is changed: %s\n",getcwd(cwd, sizeof(cwd)));

    }

    int b =0;
    if((isalpha(path[0])==0)&&(path[0]!='*')){
        printf("Baslangic karakteri harf olmali ' %c ' ile baslayamazsiniz ", path[0]);
        exit(0);
    }

    struct stat s;
    int err = stat(path, &s);
    if(-1 == err) {
        if(ENOENT == errno) {
            /* does not exist */
            if(path[0]=='*'){
                printf("\n");
            }
            else{printf("Boyle bir dizin yok\n");}
        } else {
            perror("stat");
            exit(1);
        }
    } else {
        if(S_ISDIR(s.st_mode)) {

            chdir(path);
            printf("Directory is changed: %s\n",getcwd(cwd, sizeof(cwd)));

            /* it's a dir */
            //printf("var.");
        } else {
            /* exists but is no dir */
        }
    }
}

}}}}

```

```

        if(yildizliif&&yieldizliifnot){
if((ifde_mi!=2 || (ifde_mi==1&&flag1==true)) || (ifnot_mi!=2 || (ifnot_mi==1&&flag1==true))){
if(strcmp(dest,"make")==0 || strcmp(dest,"make ")==0){

    printf("Keyword %s\n",dest);
    if(ifde_mi==0&&ifnot_mi==2){
        //printf("gec burayi..\n");
        printf("Directory can't be created..\n");
        continue;
    }
    if(ifde_mi==2&&ifnot_mi==0){

        printf("Directory can't be created..\n");
        continue;
    }

    size_t destination_size = sizeof (cwd);
    strncpy(kopyacwd, cwd, destination_size);
    //printf("kopya %s\n",kopyacwd);
//i++;
    if(satir[i]=='<'){
        i++;
        begin=i;

        while(satir[i]!='>'){

            substrlength++;
            i++;

        }

        path=substr(satir, begin, begin+substrlength);
        substrlength=0;
        pathlength=strlen(path);

        if(path[0]=='/' || path[pathlength-1]=='/'){

            printf("(hi/there) is not allowed!\n");

            exit(1);

        }

        char* token = strtok(path, "/");

int b =0;
while (token != NULL) {
    // printf("%s\n", token);

strcpy(listee[b], token);

```



```

//a[b] = token;
token = strtok(NULL, "/");
//printf("Liste[0]: %c \n",listee[b][0]);
if((isalpha(listee[b][0])==0)&&(listee[b][0]!='*')){
    printf("Baslangic karakteri harf olmalı '%c' ile baslayamaz.",listee[b][0]); //hata mesajlari bastirilir.
    exit(0);
}
b++;
}
//compare3=strcmp(listee[0][0],"*");

bool yildiz ;
compare1=strcmp(listee[0],"*");
if(compare1!=0){

    yildiz==false;
}

compare2=strcmp(listee[1],"*");

if(compare1!=0&& compare2==0){

    printf("(hi/*there) is not allowed!\n");

    exit(1);

}

int indeks=0;
int a=0;

while(indeks<b){

    if(compare1==0&&indeks==0&& hatali==false){

        chdir("..");
        printf("%s \n",getcwd(cwd, sizeof(cwd)));

        indeks++;
    }
    if(compare2==0&&indeks==1&& hatali==false){

        chdir("..");
        printf("%s \n",getcwd(cwd, sizeof(cwd)));

        indeks++;
    }

}

if(!hatali){

```

```

        char* dirname = listee[indeks];
        check = mkdir(dirname);

        // check if directory is created or not
        if (!check)
            printf("Directory created\n");
        else {
            printf("Directory is created before!\n");
            strcat(cwd,slash);
        strcat(cwd,dirname);
            chdir(cwd);
            //token = strtok(NULL, "/");
            indeks++;
            continue;
            //exit(1);
            getch();

            system("dir");
            getch();

        }

        strcat(cwd,slash); //make komutunda currenta dir ekleme işlemi.
        strcat(cwd,dirname);
        chdir(cwd);

        printf("%s\n",getcwd(cwd, sizeof(cwd)));

        //listee[indeks];

        indeks++;
        yildiz=true;
        //hatali=true;
    }
    //printf("%d",indeks);
}

    indeks==0;
    b=0;

//int b =0;
//while(b<a){

        chdir(kopyacwd);
        //b++;
    //}
    printf("Current Directory: %s\n",getcwd(cwd, sizeof(cwd)));

//printf("%d",a);

}

```

```
}}
```

```
if(ifde_mi!=2 || (ifde_mi==1&&flag1==true) || (ifde_mi!=2&&flag1==false)){  
if(strcmp(dest,"if")==0 || strcmp(dest,"if ")==0){  
printf("Keyword %s\n",dest);
```

```
if(satir[i]=='<'){  
i++;  
begin=i;
```

```
while(satir[i]!='>'){
```

```
    substrlength++;  
    i++;
```

```
}  
path=substr(satir, begin, begin+substrlength);  
substrlength=0;  
pathlength=strlen(path);
```

```
if(path[0]=='/' || path[pathlength-1]=='/'){
```

```
    printf("(\"\"/hi/there) or (hi/there/) is not allowed!\n");
```

```
    exit(1);
```

```
}  
// printf("%s pathh\n",path);
```

```
int d;  
if(strcmp(path,"*")==0){  
    //printf("girdi..");  
    size_t destination_size = sizeof(cwd);  
    strncpy(kopyaacwd, cwd, destination_size);  
    //printf("kopya %s\n",kopyaacwd);  
printf("Current Directory: %s\n",getcwd(cwd, sizeof(cwd)));
```

```
    //chdir(kopyaacwd);  
    int i=0;  
    int slashesayi=0;  
    int chdirsayi;
```

```
if (getcwd(cwd, sizeof(cwd)) != NULL) {  
    //printf("Current working dir: %s\n", cwd);  
    int i=0;  
    while(cwd[i]!='\0')          /*if keywordunde üst dizinin varligini kontrol  
                                ettigimiz kod bolumu*/
```

```
{  
    if(cwd[i]=='\\')  
    {
```

```

        slashesayi++;
    }
    i++;
}
//printf("%d\n",slashesayi);
chdirsayi=slashesayi-1;

if(chdirsayi>0){

    yildizliif=true;        //go ve make keywordlerine girisi kontrol icin kullanilacak bool yapıları.
    ifde_mi=1;

}
else{

    yildizliif=false;
    ifde_mi=2;

}

}

}

}

//char* token = strtok(path, "/");

int b =0;
//printf("Liste[0]: %c \n",path[0]);
if((isalpha(path[0])==0)&&(path[0]!='*')){
    printf("Baslangic karakteri harf olmalı ' %c ' ile baslayamazsiniz ", path[0]);
    exit(0);
}

struct stat s;
int err = stat(path, &s);
if(ifde_mi!=1){                /*pathin varligini kontrol eden if-else yapisi...*/
if(-1 == err) {
    if(ENOENT == errno) {
        /* does not exist */
        //printf("ife girilemedi\n");
        ifde_mi=2;
    } else {
        perror("stat");
        exit(1);
    }
} else {

//ifde_mi=true;

```

```

if(S_ISDIR(s.st_mode)) {

    if(ificemi&&ifde_mi==2){

        ifde_mi=2;

    }
else{
    ifde_mi=1;

    /* it's a dir */

}
else {
    /* exists but is no dir */
}
}

}

}

if(strcmp(dest,"ifnot")==0 || strcmp(dest,"ifnot ")==0){
    printf("Keyword: %s\n",dest);
    if(satir[i]=='<'){
        i++;
        begin=i;

        while(satir[i]!='>'){

            substrlength++;
            i++;

        }
        path=substr(satir, begin, begin+substrlength);
        substrlength=0;
        pathlength=strlen(path);

        if(path[0]=='/' || path[pathlength-1]=='/'){

            printf("/hi/there) or (hi/there/) is not allowed!\n");

            exit(1);

        }

        if(strcmp(path,"*")==0){
            //printf("girdi..");
            size_t destination_size = sizeof (cwd);
            strncpy(kopyaacwd, cwd, destination_size);
            //printf("kopya %s\n",kopyaacwd);

```

```

printf("Current Directory: %s\n",getcwd(cwd, sizeof(cwd)));

//chdir(kopyaacwd);
int i=0;
int slashesayi=0;
int chdirsayi;

if (getcwd(cwd, sizeof(cwd)) != NULL) {
    //printf("Current working dir: %s\n", cwd);
    int i=0;
    while(cwd[i]!='\0')
    {
        if(cwd[i]=='\\')
        {
            slashesayi++;
        }
        i++;
    }
    //printf("%d\n",slashesayi);
    chdirsayi=slashesayi-1;
    //printf("bu kadar chdir yapabilir..%d\n",chdirsayi);

    if(chdirsayi>0){

        yildizliifnot=false;
        ifnot_mi=1;

    }
    else{

        yildizliif=true;
        ifnot_mi=2;

    }

}

//printf("%s pathh\n",path);

//char* token = strtok(path, "/");
int b =0;

//printf("Liste[0]: %c \n",path[0]);
if((isalpha(path[0])==0)&&(path[0]!='*')){
    printf("Baslangic karakteri harf olmali ' %c ' ile baslayamazsiniz ", path[0]);
    exit(0);
}

struct stat s;
int err = stat(path, &s);

```

```

if(-1 == err) {
    if(ENOENT == errno) {
        /* does not exist */

        ifnot_mi=1;
    } else {
        perror("stat");

        exit(1);
    }
} else {

    //ifde_mi=true;
    if(S_ISDIR(s.st_mode)) {
        ifnot_mi=2;

    } else {
        /* exists but is no dir */
    }
}

}
}

if((ifnot_mi==1&&satir[i]=='{') || (ifde_mi==1&&satir[i]=='')){
    flag1=true;
    if(ifnot_mi==1){
        //printf("ifnota girdi!!!");
    }
    if(ifde_mi==1){
        //printf("ife girdi...\n");
    }
}

}

if((ifde_mi==1 || ifde_mi==2&&satir[i]=='}') || (ifnot_mi==1 || ifnot_mi==2&&satir[i]=='')){
    flag1=false;
    ifde_mi=0;
    ifnot_mi=0;

    //printf("iften/ifnottan ciktik\n");
}
satirlength ++;

if(satir[i]=='{'){                //if ve ifnotta parantez durumu kontrolleri.

flag2=true;
}

if(satir[i]=='}')&&flag2){
    //printf("if bitimi\n");
    ifde_mi=0;
    ifnot_mi=0;
}

```

```
    flag2=false;
    yildizliif=true;
    yildizliifnot=true;

}
if(satir[i]=='&&flag1==false&&flag2==false){
    //printf("if bitimi ...\n");
    //ifde_mi=0;
    flag2=false;
    ifde_mi=0;
    ifnot_mi=0;
    yildizliif=true;
    yildizliifnot=true;

}

dest=" ";

}

}
}

fclose(fp);
return 0;
}
```