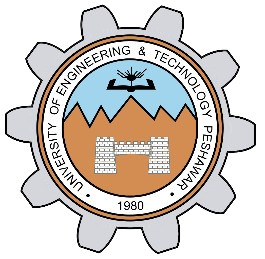
**TRAVERSING DIRECTORIES**

**LAB # 09**



**Fall 2022**

**CSE302L System Programming Lab**

Submitted by: **Maaz Habib**

Registration No. : **20PWCSE1952**

Class Section: **C**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.” Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Abdullah Hamid**

Tuesday, January 9 , 2023

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Task # 01:**

**Traverse directory tree in depth-first order.**

**Code:**

#include <stdio.h>

#include <unistd.h>

#include <dirent.h>

#include <sys/stat.h>

#include <time.h>

#include <pwd.h>

int i=0;

void depthfirst(char \*name)

{

i++;

DIR \*dirp = opendir(name);

if(dirp==NULL)

{

perror("Failed to open directory");

}

struct dirent \*direntp;

struct stat buffer;

int retchd = chdir(name);

if(retchd==-1)

perror("Failed to change directory");

while((direntp=readdir(dirp))!=NULL)

{

if(direntp->d\_name[0]=='.')

continue;

for(int j=0;j<i;j++)

printf(" ");

printf("%s\n",direntp->d\_name);

int ret = stat(direntp->d\_name,&buffer);

if(ret==-1)

perror("Error using stat");

if(S\_ISDIR(buffer.st\_mode))

{

depthfirst(direntp->d\_name);

chdir("..");

i--;

}

}

}

int main(int argc, char \*argv[])

{

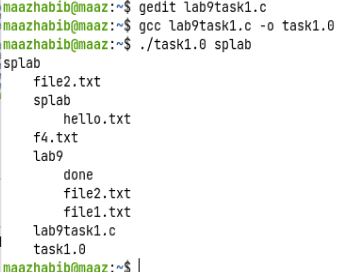
printf("%s\n",argv[1]);

depthfirst(argv[1]);

return 0;

}

**Output:**



**Task # 02:**

**Traverse directory tree in breadth-first order.**

**Code:**

#include <stdio.h>

#include <unistd.h>

#include <dirent.h>

#include <sys/stat.h>

#include <time.h>

#include <pwd.h>

void breadthfirst(char \*name)

{

DIR \*dirp = opendir(name);

if(dirp==NULL)

{

perror("Failed to open directory");

}

struct dirent \*direntp;

struct stat buffer;

char cwd[400];

int retchd = chdir(name);

if(retchd==-1)

perror("Failed to change directory");

if(getcwd(cwd,sizeof(cwd))==NULL)

perror("Failed to get cwd");

while((direntp=readdir(dirp))!=NULL)

{

if(direntp->d\_name[0]=='.')

continue;

printf("%s/%s\n",cwd,direntp->d\_name);

}

rewinddir(dirp);

while((direntp=readdir(dirp))!=NULL)

{

if(direntp->d\_name[0]=='.')

continue;

int ret = stat(direntp->d\_name,&buffer);

if(ret==-1)

perror("Error using stat");

if(S\_ISDIR(buffer.st\_mode))

{

breadthfirst(direntp->d\_name);

chdir("..");

}

}

}

int main(int argc, char \*argv[])

{

breadthfirst(argv[1]);

return 0;

}

**Output:**



**Task # 03:**

**Implement the find utility.**

**Code:**

#include <stdio.h>

#include <unistd.h>

#include <dirent.h>

#include <sys/stat.h>

#include <time.h>

#include <pwd.h>

#include <string.h>

int found=0;

void pfind(char \*dir, char \*file)

{

DIR \*dirp = opendir(dir);

if(dirp==NULL)

{

perror("Failed to open directory");

}

struct dirent \*direntp;

struct stat buffer;

if(chdir(dir)==-1){

perror("Unable to change directory");

}

char cwd[400];

while((direntp=readdir(dirp))!=NULL)

{

int ret = stat(direntp->d\_name,&buffer);

if(ret==-1)

{

perror("Stat Function Error");

}

if(direntp->d\_name[0]=='.')

continue;

if(strcmp(direntp->d\_name,file)==0)

{

if(getcwd(cwd,sizeof(cwd))==NULL)

perror("Failed to get cwd");

printf("File location: %s\n",cwd);

found=1;

}

if(S\_ISDIR(buffer.st\_mode))

{

pfind(direntp->d\_name,file);

chdir("..");

}

}

}

int main(int argc, char \*argv[])

{

pfind(argv[1],argv[2]);

if(!found)

printf("File not found\n");

return 0;

}

**Output:**

****