Muchamad Rif'an

17.01.53.2021

Teknik Informatika R2

EXPERIMENT 7

Objective

**Write a C program to simulate the following file organization techniques.

1. Single level directory.
2. Two level directory.
3. Hierarchical

Description

The directory structure is the organization of files into a hierarchy of folders. In a single-level directory system, all the files are placed in one directory. There is a root directory which has all files. It has a simple architecture and there are no sub directories. Advantage of single level directory system is that it is easy to find a file in the directory. In the two-level directory system, each user has own user file directory (UFD). The system maintains a master block that has one entry for each user. This master block contains the addresses of the directory of the users. When a user job starts or a user logs in, the system's master file directory (MFD) is searched. When a user refers to a particular file, only his own UFD is searched. This effectively solves the name collision problem and isolates users from one another. Hierarchical directory structure allows users to create their own subdirectories and to organize their files accordingly. A tree is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name. A directory (or subdirectory) contains a set of files or subdirectories.

Program

1. SINGLE LEVEL DIRECTORY ORGANIZATION

```
#include<stdio.h>
struct { char dname[10],fname[10][10]; int fcnt;
}
dir;
void main()
{
int i,ch;
char f[30];
clrscr();
dir.fcnt = 0;
printf("\nEnter name of directory -- ");
scanf("%s", dir.dname);
while(1)
```

```
{
printf("\n\n1. Create File\t2. Delete File\t3. Search File \n 4. Display Files\t5. Exit\nEnter your choice --
");
scanf("%d",&ch);
switch(ch)
{
case 1: printf("\nEnter the name of the file -- ");
scanf("%s",dir.fname[dir.fcnt]);
dir.fcnt++;
break;
case 2: printf("\nEnter the name of the file -- ");
scanf("%s",f);
for(i=0;i<dir.fcnt;i++)
{
if(strcmp(f, dir.fname[i])==0)
{
printf("File %s is deleted ",f);
strcpy(dir.fname[i],dir.fname[dir.fcnt-1]);
break;
}
}
if(i==dir.fcnt)
printf("File %s not found",f);
else
dir.fcnt--;
break;
case 3: printf("\nEnter the name of the file -- ");
scanf("%s",f);
for(i=0;i<dir.fcnt;i++)
{
if(strcmp(f, dir.fname[i])==0)
{
printf("File %s is found ", f);
```

```
break;
}
}
if(i==dir.fcnt)
printf("File %s not found",f);
break;
case 4: if(dir.fcnt==0) printf("\nDirectory Empty");
else
{
printf("\nThe Files are -- ");
for(i=0;i<dir.fcnt;i++)
printf("\t%s",dir.fname[i]);
}
break;
default: exit(0);
}
}
getch();
}
```

OUPUT :

Enter name of directory -- CSE

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit

Enter your choice -- 1

Enter the name of the file -- RFNFILE1

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit

Enter your choice -- 1

Enter the name of the file -- RFNFILE2

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit

Enter your choice -- 1

Enter the name of the file -- RFNFILE3

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit

Enter your choice -- 4

The Files are -- RFNFILE1 RFNFILE2 RFNFILE3

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit

Enter your choice -- 3

Enter the name of the file -- RIFANFILE123

File RIFANFILE123 not found

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit

Enter your choice -- 2

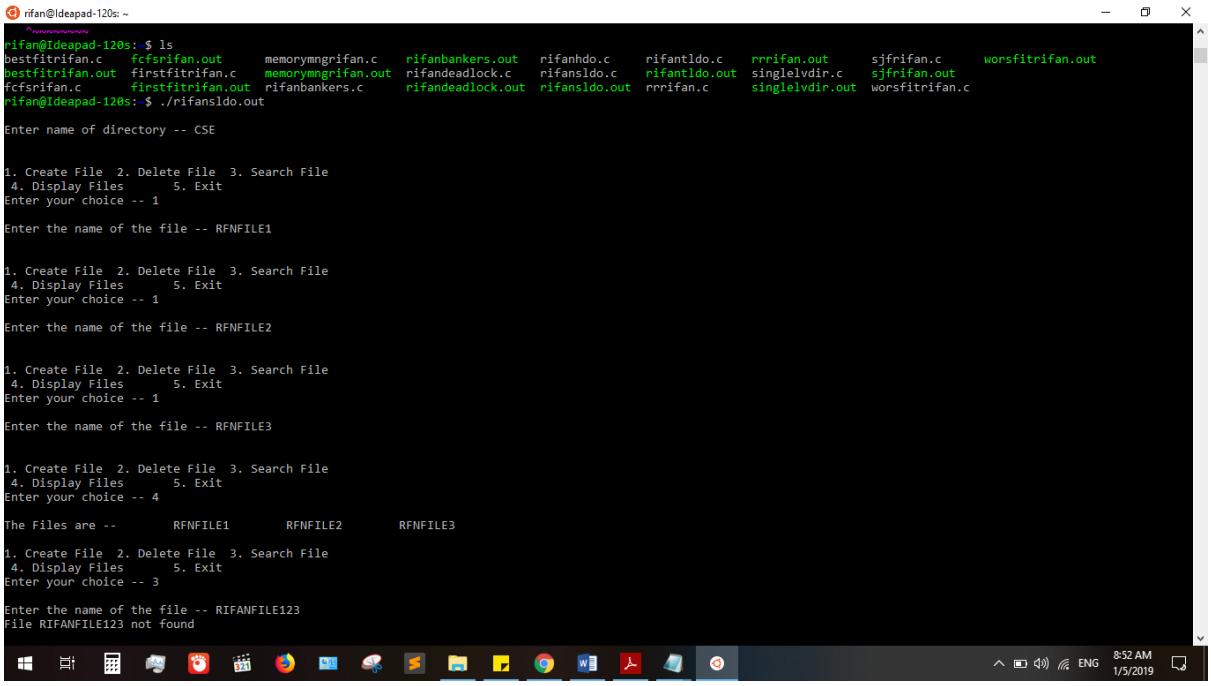
Enter the name of the file -- RIFANFILE2

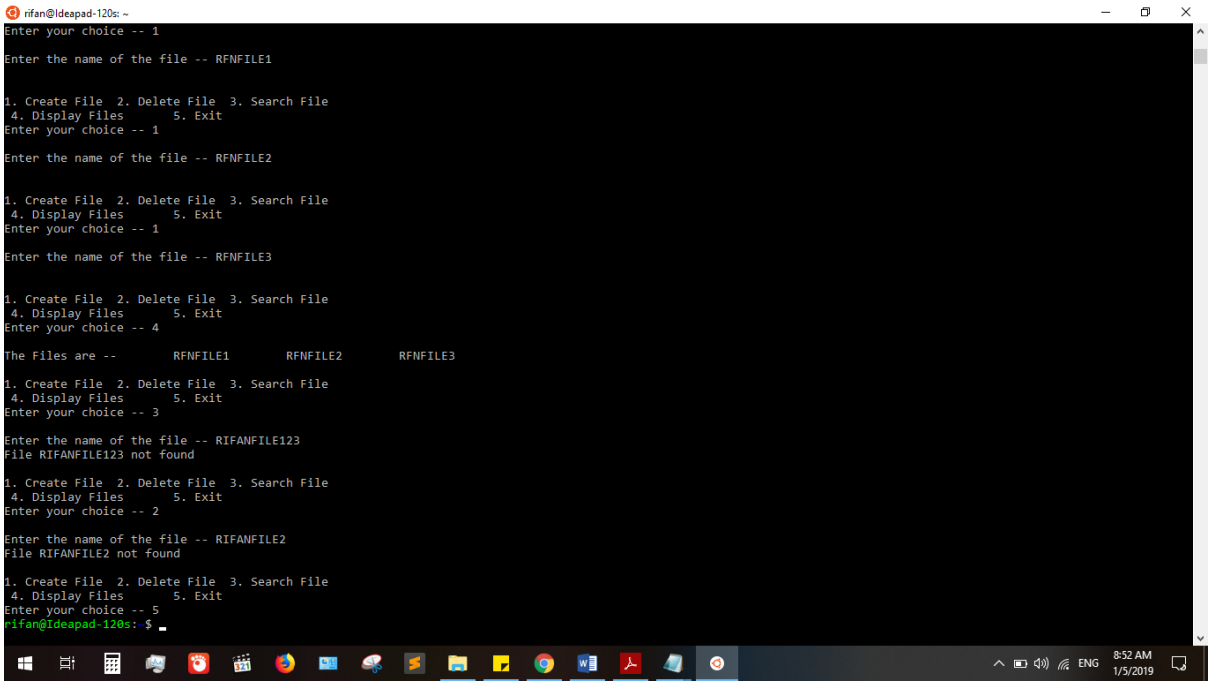
File RIFANFILE2 not found

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit

Enter your choice -- 5





2. TWO LEVEL DIRECTORY ORGANIZATION

```
#include<stdio.h>
struct
{
char dname[10],fname[10][10];
int fcnt;
}
dir[10];
void main()
{
int i,ch,dcnt,k;
char f[30], d[30];
dcnt=0;
while(1)
{
printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");
printf("\n4. Search File\t\t5. Display\t6. Exit\t Enter your choice -- ");
scanf("%d",&ch); switch(ch) { case 1: printf("\nEnter name of directory -- ");
scanf("%s", dir[dcnt].dname);
dir[dcnt].fcnt=0; dcnt++;
printf("Directory created");
break;
case 2: printf("\nEnter name of the directory -- ");
scanf("%s",d);
```

```
for(i=0;i<dcnt;i++) if(strcmp(d,dir[i].dname)==0) { printf("Enter name of the file -- ");
scanf("%s",dir[i].fname[dir[i].fcnt]); dir[i].fcnt++; printf("File created");
break;
} if(i==dcnt) printf("Directory %s not found",d);
break;
case 3: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)
{
printf("Enter name of the file -- ");
scanf("%s",f);
for(k=0;k<dir[i].fcnt;k++)
{
if
(strcmp(f, dir[i].fname[k])==0)
{
printf("File %s is deleted ",f);
dir[i].fcnt--;
strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);
goto jmp;
}
}
printf("File %s not found",f);
goto jmp;
}
}
printf("Directory %s not found",d);
jmp : break;
case 4: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)
{
printf("Enter the name of the file -- ");
scanf("%s",f);
for(k=0;k<dir[i].fcnt;k++)
{
if(strcmp(f, dir[i].fname[k])==0)
{
printf("File %s is found ",f);
goto jmp1;
}
}
printf("File %s not found",f);
goto jmp1;
```

```
}
}
printf("Directory %s not found",d);
jmp1: break;
case 5: if(dcnt==0) printf("\nNo Directory's ");
else { printf("\nDirectory\tFiles");
for(i=0;i<dcnt;i++)
{
printf("\n%s\t\t",dir[i].dname);
for(k=0;k<dir[i].fcnt;k++)
printf("\t%s",dir[i].fname[k]);
}
}
break;
default:exit(0);
}
}
getch();
}
```

OUTPUT

```
1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice -- 1

Enter name of directory -- DIRRIFAN1
Directory created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice -- 1

Enter name of directory -- DIRRIFAN2
Directory created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice -- 2

Enter name of the directory -- DIRRIFAN1
Enter name of the file -- FILERIFAN1
File created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice -- 2

Enter name of the directory -- DIRRIFAN1
Enter name of the file -- FILERIFAN2
File created
```

1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 5

Directory Files
DIRRIFAN1 FILERIFAN1FILERIFAN2 FILERIFAN2
DIRRIFAN2

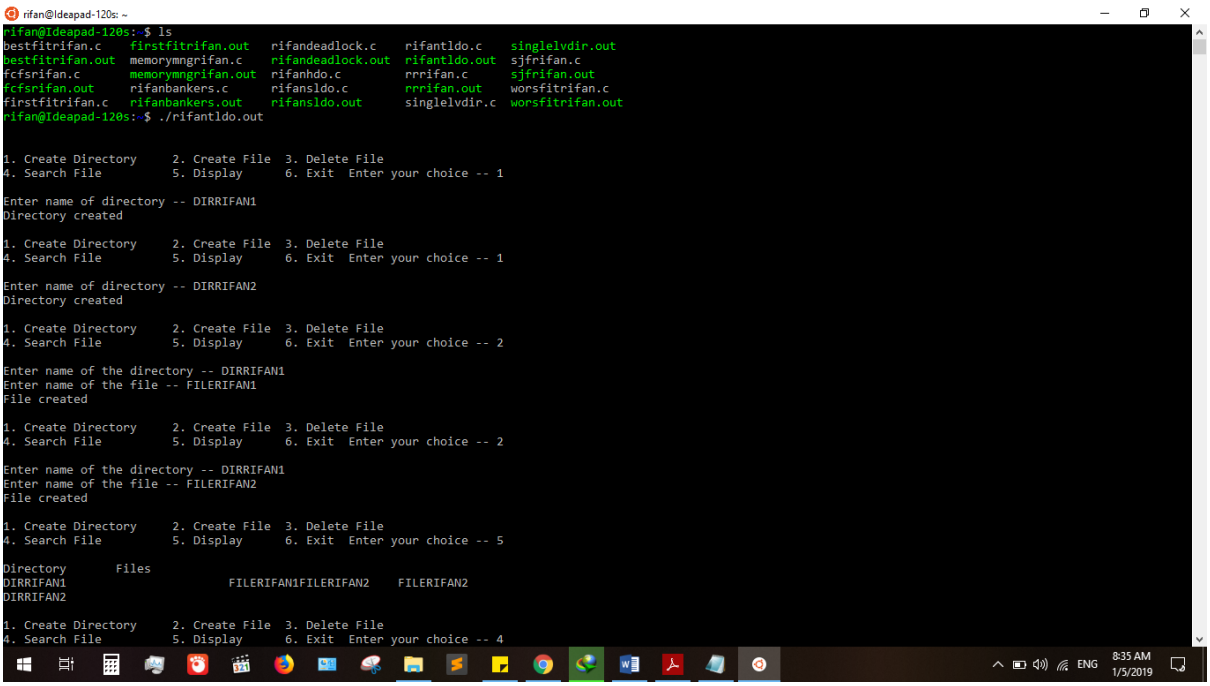
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 4

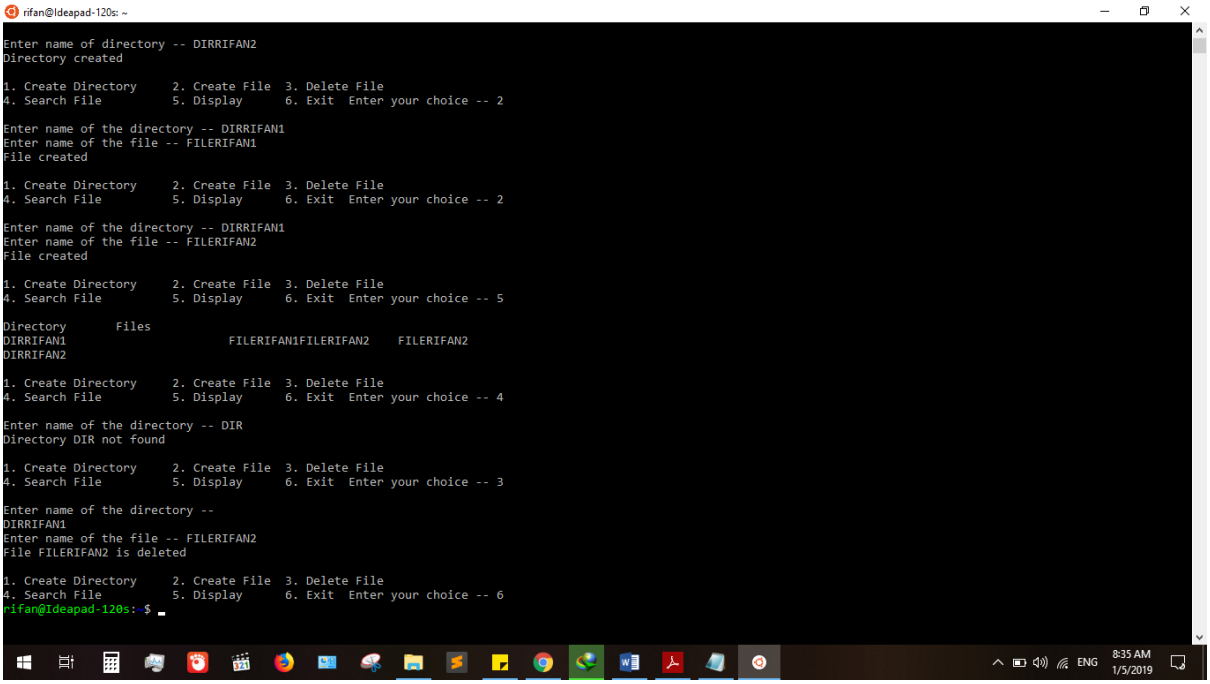
Enter name of the directory -- DIR
Directory DIR not found

1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 3

Enter name of the directory --
DIRRIFAN1
Enter name of the file -- FILERIFAN2
File FILERIFAN2 is deleted

1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 6

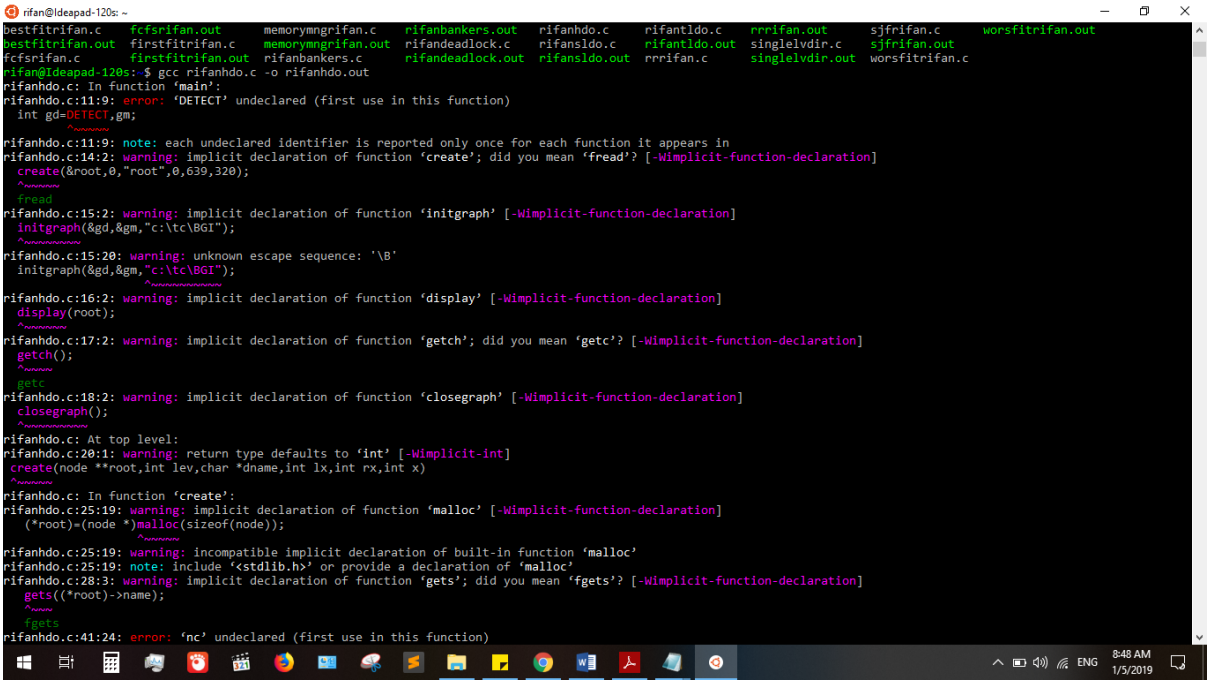




3. HIERARCHICAL DIRECTORY ORGANIZATION

```
#include<stdio.h>
#include<graphics.h>
struct tree_element
{
char name[20];
int x, y, ftype, lx, rx, nc, level;
struct tree_element *link[5];
};
typedef struct tree_element node;
void main()
{
int gd=DETECT,gm;
node *root;
root=NULL;
create(&root,0,"root",0,639,320);
initgraph(&gd,&gm,"c:\tc\BGI")display(root)entergetch();
closegraph();
```

```
}
create(node **root,int lev,char *dname,int lx,int rx,int x)
{
int i, gap;
if(*root==NULL)
{
(*root)=(node *)malloc(sizeof(node));
printf("Enter name of dir/file(under %s) : ",dname);
fflush(stdin);
gets((*root)->name);
printf("enter 1 for Dir/2 for file :");
scanf("%d",&(*root)->ftype);
(*root)->level=lev;
(*root)->y=50+lev*50;
(*root)->x=x;
(*root)->lx=lx;
(*root)->rx=rx;
for(i=0;i<5;i++) (*root)->link[i]=NULL;
if((*root)->ftype==1)
{
printf("No of sub directories/files(for %s):",(*root)->name);
scanf("%d",&(*root)>nc);
if((*root)->nc==0) gap=rx-lx;
else gap=(rx-lx)/(*root)->nc;
for(i=0;i<(*root)->nc;i++)
create(&((*root)>link[i]),lev+1,(*root)>name,lx+gap*i,lx+gap*i+gap, lx+gap*i+gap/2);
}
else (*root)->nc=0;
}
}
display(node *root)
{
int i;
settextstyle(2,0,4);
settextjustify(1,1);
setfillstyle(1,BLUE);
setcolor(14);
if(root !=NULL) { for(i=0;i<root->nc;i++) line(root->x,root->y,root->link[i]->x,root->link[i]->y);
if(root->ftype==1) bar3d(root->x-20,root->y-10,root->x+20,root>y+10,0,0);
else fillellipse(root->x,root->y,20,20);
outtextxy(root->x,root->y,root->name);
for(i=0;i<root->nc;i++) display(root->link[i]);
}
}
```


rifan@Ideapad-120s: ~
bestfitrifan.c fcfsrifan.out memorymngrifan.c rifanbankers.out rifanhdo.c rifantldo.c rrrifan.out sjfrifan.c worsfitrifan.out
bestfitrifan.out firstfitrifan.c memorymngrifan.out rifandeadlock.c rifansldo.c rifantldo.out singlelvdir.c sjfrifan.out
fcfsrifan.c firstfitrifan.out rifanbankers.c rifandeadlock.out rifansldo.out rrrifan.c singlelvdir.out worsfitrifan.c
rifan@Ideapad-120s:~$ gcc rifanhdo.c -o rifanhdo.out
rifanhdo.c: In function 'main':
rifanhdo.c:11:9: error: 'DETECT' undeclared (first use in this function)
int gd=DETECT,gm;
^~~~~~
rifanhdo.c:11:9: note: each undeclared identifier is reported only once for each function it appears in
rifanhdo.c:14:2: warning: implicit declaration of function 'create'; did you mean 'fread'? [-Wimplicit-function-declaration]
create(&root,0,"root",0,639,320);
^~~~~~
fread
rifanhdo.c:15:2: warning: implicit declaration of function 'initgraph' [-Wimplicit-function-declaration]
initgraph(&gd,&gm,"c:\tc\BGI");
^~~~~~~~~
rifanhdo.c:15:20: warning: unknown escape sequence: '\B'
initgraph(&gd,&gm,"c:\tc\BGI");
^~~~~~~~~~
rifanhdo.c:16:2: warning: implicit declaration of function 'display' [-Wimplicit-function-declaration]
display(root);
^~~~~~~
rifanhdo.c:17:2: warning: implicit declaration of function 'getch'; did you mean 'getc'? [-Wimplicit-function-declaration]
getch();
^~~~~
getc
rifanhdo.c:18:2: warning: implicit declaration of function 'closegraph' [-Wimplicit-function-declaration]
closegraph();
^~~~~~~~~~
rifanhdo.c: At top level:
rifanhdo.c:20:1: warning: return type defaults to 'int' [-Wimplicit-int]
create(node **root,int lev,char *dname,int lx,int rx,int x)
^~~~~~
rifanhdo.c: In function 'create':
rifanhdo.c:25:19: warning: implicit declaration of function 'malloc' [-Wimplicit-function-declaration]
(*root)=(node *)malloc(sizeof(node));
^~~~~~
rifanhdo.c:25:19: warning: incompatible implicit declaration of built-in function 'malloc'
rifanhdo.c:25:19: note: include '<stdlib.h>' or provide a declaration of 'malloc'
rifanhdo.c:28:3: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
gets((*root)->name);
^~~~
fgets
rifanhdo.c:41:24: error: 'nc' undeclared (first use in this function)
ENG
8:48 AM
1/5/2019