Ex. No.: 12
Date: 24/4/25

File Organization Technique- Single and Two level directory

AIM:

To implement File Organization Structures in C are

a. Single Level Directory
b. Two-Level Directory
c. Hierarchical Directory Structure
d. Directed Acyclic Graph Structure

a. Single Level

Directory

ALGORITHM

1. Start
2. Declare the number, names and size of the directories and file names.
3. Get the values for the declared variables.
4. Display the files that are available in the directories.
5. Stop.

PROGRAM:

```
import turtle
def draw_directory (count, filenames):
    Screen = turtle.Screen ()
    Screen.bgcolor ("green")
    t = turtle.Turtle ()
    t.speed (0)
    t.goto (-100, 50)
    t.setup
    t.pendown ()
    t.color ("magneta")
```

```
t.begin_fill()
for _ in range(2):
    t.forward(200)
    t.left(90)
    t.forward(50)
    t.left(90)
t.end_fill()

t.penup()
t.goto(0,25)
t.color("blue")
t.write("Root Directory", align="center", font=
("Arial", 12, "normal"))

mid = 600/count
cir_x = -200 + mid/3

for i in range(count):
    t.penup()
    t.goto(0,50)
    t.pendown()
    t.setheading(t.towards(cir_x,-100))
    t.goto(cir_x,-100)
    t.penup()
    t.goto(cir_x,-100)
    t.pendown()
    t.begin_fill()
    t.circle(30)
    t.end_fill()
```

```
t.penup()
t.goto(cir_x, -120)
t.write(filenames[i], align="center", font=("Arial", 10, "normal"))
cir_x += mid
t.hideturtle()
turtle.done()

def main():
    count = int(input("Enter the number of files:"))

    filesname = []
    for i in range(count):
        filename = input(f"Enter the name of file {i+1}:")

        filenames.append(filename)

    draw_directory(count, filenames)
if __name__ == "__main__":
    main()
```

b. Two-level directory Structure

ALGORITHM:

1. Start
2. Declare the number, names and size of the directories and subdirectories and file names.
3. Get the values for the declared variables.
4. Display the files that are available in the directories and subdirectories.
5. Stop.

PROGRAM:

```
import tkinter as tk

class TreeNode:
    def __init__(self, name, level, x, y, lx, rx, ftype):
        self.name = name
        self.level = level
        self.x = x
        self.y = y
        self.lx = lx
        self.rx = rx
        self.ftype = ftype
        self.children = []

def create_node(level, dname, lx, rx, x):
    name = input(f"Enter name of dir / file (under {dname}): ")
    ftype = 1 if level in [0, 1] else 2
    y = 50 + level * 50
    node = TreeNode(name, level, x, y, lx, rx, ftype)
```

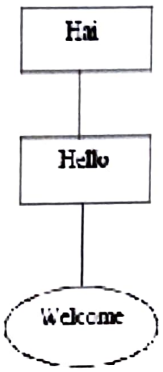
```
if ftype == 1:
    if level == 0:
        nc = int(input(f"How many users (for {name}): "))
    Else:
        nc = int(input(f"How many files (for {name}): "))
    gap = (rx - lx)
    for i in range(nc):
        child_x = lx + i * gap + gap
        child = create_node(level+1, name, lx+i*gap,
                lx + (i+1) * gap, child_x)
        node.children.append(child)
    return node

def display_tree(canvas, node):
    if node:
        for child in node.children:
            canvas.create_line(node.x, node.y, child.x,
                               child.y)
            display_tree(canvas, child)
        if node.ftype == 1:
            canvas.create_rectangle(node.x - 20,
node.y - 10, node.x + 20, node.y + 10, fill = "blue")
        else:
            canvas.create_oval(node.x - 20, node.y - 20,
node.x + 20, node.y + 20, fill = "green")
```

```
        canvas.create_text(node.x, node.y, text = node.name,
        fill = "white")
if __name__ == "__main__":
    root_node = create_node(0, "null", 0, 630, 326)
    window = tk.Tk()
    window.title("Tree Structure")
    canvas = tk.Canvas(window, width = 640,
    height = 480, bg = "black")
    canvas.pack()
    display_tree(canvas, root_node)
    window.mainloop()
```

Sample Output:

Enter the name of dir/file(under null): Hai
How many users(for Hai):1
Enter name of dir/file(under Hai):Hello
How many files(for Hello):1
Enter name of dir/file(under Hello):welcome



Result:

Thus the Python code for file Structure organization was executed successfully