/* 6. Write a C program to simulate the following file organization techniques:
a) Single level directory b) Two level directory c) Hierarchical */

// Single level directory - All files contained in same directory
// Easy to support and understand, large number of names requirend,
// But how to support different user?
// When represented as tree structure, tree height = 1
// Tree height = max (number of edges encountered when traversing from root to leaf/last
// file/directory)

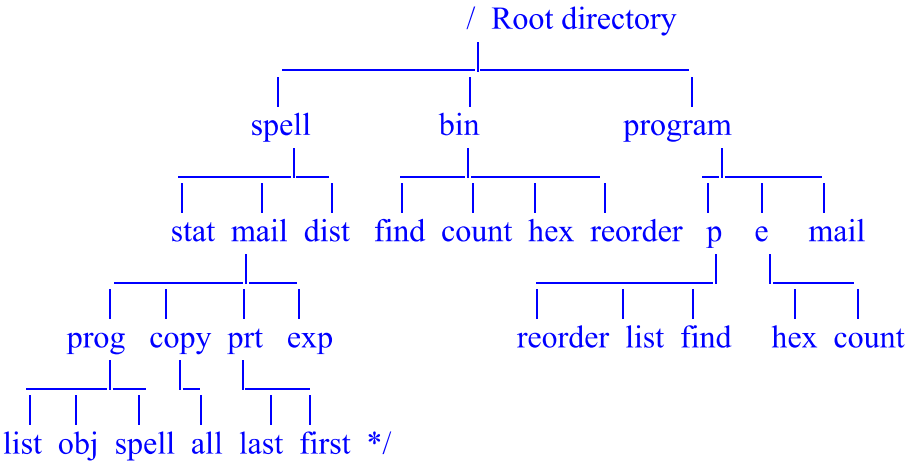
// Two level directory-one Master File Directory(MFD), supports separate directory for
// each user, User has own User File Directory(UFD), MFD has User name pointing to UFD
// When represented as tree structure, tree height = 2
// Root = MFD , direct descendants / sub directories = UFD

// Hierarchical, Tree Structure - Directory / Tree structure with arbitary height
// Single and Two level directory are specific case of Hierarchical directory with
// height 1 and 2 respectively; Users may create their own subdirectories
// One bit in each directory entry defines the entry as a file (0) or as a subdirectory (1)
// Path to a file in a tree-strucured directory can be longer than in a two-level directory

// Single level : All files have have parent as root directory, inode 0
// sub directory can not be created, only regular files, file with same names not allowed
// / Root directory , the only directory
// _____|_____
// | | | | | | | | |
// cat bo a test data mail cont hex records ; all are regular files

// Two level : Root directory with indoe 0 , can have file or sub directory, not sub directories
// / Root directory , can have sub directory
// _____|_____
// | | | |
// user1 user2 user3 user4 Directories, they can have files, but no subdir
// _____|__ __|__ __|__ __|_____
// | | | | | | | | | | | |
// cat bo a test a data a test x data a ; all are regular files

/* Hierarchical, Tree Structure - Directory / Tree , will have Root directory, no restrictions on files or sub directories



// Generic program that can maintain details of file organization of arbitary height.

```
#include <stdio.h>
#include <string.h>

int  currentInodeNumber=0; // Keep the count of allocated inode numbers,
                           // Hence we can calculate next free inode number

struct metadataOfFile  {
   int  parent; // Parent directory inode, -1 if no parent, example root node
   int  type; // 0 for regular, 1 for directory
   int  level; // level of file, root directory level is 0
   char name[9]; // File name, maximum 8 characters
 } fileMetadata[256];
// iNode + metadataOfFile structure rougly implementing Inode table structure and directory
// file structure with details inode number , type of file , level the file is on and name of file
int level;

void displayFileOrganisation( )  { // Function to display file organisation
   printf("\n\n File organisation\n");

   for( int i=0; i<currentInodeNumber; i++ )
    {
     printf(" Inode = %d , Parent = %d , Type = %d , Name = %s\n",i , fileMetadata[i].parent,
fileMetadata[i].type, fileMetadata[i].name );
    }
 }

void  create( int fileType )// Function reads file name, parent directory
 { // gets first free inode number, update inode as used, update file metadata
   char name[9]; // variable to read new file name
   int parentDirInode; // variable to save parent directory inode number
   printf("\n For the following file organisation : \n ");

   displayFileOrganisation();

   printf("\n Please enter inode number of directory where you want to create file or (sub)direc
tory= ");
   scanf("%d",&parentDirInode); // Lets assume valid inode of directory is entered

   if(  fileType == 1 && level == 1 ) { // Creating new directory not allowed in Sinlge level
      printf("\n Creating sub directory not possible in Single Level");
      return;
    }
   else if( fileType == 1 && fileMetadata[parentDirInode].level == 1 && level == 2 )
    {// In two level, further sub directory not allowed in sub directory of root
      printf("\n Creating further sub directories not possible under this directory");
      return;
    }

   printf("\n Enter file or (sub) directory name = "); // How would you check if file name alrea
dy exists
   scanf("%s", name);//Assume non existing name of eight characters is entered

   int free = currentInodeNumber++; // Check for first free inode
                                    // Post Increment inode
```

```
    // address of fileMetadata , parent dir inode, file type, parent level + 1 and name
    fileMetadata[free].parent = parentDirInode;
    fileMetadata[free].type = fileType;
    fileMetadata[free].level = fileMetadata[parentDirInode].level+1;
    strcpy( fileMetadata[free].name, name );
    printf("\n File or (sub) Directory %s added", name);
  }

 int main()
  { int i, operation;

    // Root directory initialization, assume root directory has inode number 0
    fileMetadata[0].parent = -1;
    fileMetadata[0].type = 1;
    fileMetadata[0].level = 0;
    strcpy( fileMetadata[0].name, "rootDir" );

    printf("\n\n Root directory \"rootDir\" , with inode = 0 created\n");

    currentInodeNumber++; // Increment inode as root directory created

    printf("\n Enter\n 1 to simulate Single level directory,\n 2 for Two level directory, or\n 3 fo
r Hierarchical\n ");
    scanf("%d",&level);

    while(1) {
      printf("\n\n Enter: 1 to Create file\t 2 for Directory\t 3 to Display\t 4 to Exit\n");
      scanf("%d",&operation);
      switch( operation ) { // create( fileType ) 0 for file, 1 for directory
        case 1: create( 0 );
                break;
        case 2: create( 1 );
                break;
        case 3: displayFileOrganisation();
                break;
        case 4: return 0;
       }
     }
    return(0);//Can data structure design be improved, what is the run time for file add, display
  } // With above data structure : How will you implement delete file and directory

/*

 Root directory "rootDir" , with inode = 0 created

 Enter
 1 to simulate Single level directory,
 2 for Two level directory, or
 3 for Hierarchical
 3


 Enter: 1 to Create file     2 for Directory       3 to Display    4 to Exit
```

2

For the following file organisation :

File organisation
Inode = 0 , Parent = -1 , Type = 1 , Name = rootDir

Please enter inode number of directory where you want to create file or (sub)directory= 0

Enter file or (sub) directory name = A

File or (sub) Directory A added

Enter: 1 to Create file 2 for Directory 3 to Display 4 to Exit
1

For the following file organisation :

File organisation
Inode = 0 , Parent = -1 , Type = 1 , Name = rootDir
Inode = 1 , Parent = 0 , Type = 1 , Name = A

Please enter inode number of directory where you want to create file or (sub)directory= 1

Enter file or (sub) directory name = c

File or (sub) Directory c added

Enter: 1 to Create file 2 for Directory 3 to Display 4 to Exit
1

For the following file organisation :

File organisation
Inode = 0 , Parent = -1 , Type = 1 , Name = rootDir
Inode = 1 , Parent = 0 , Type = 1 , Name = A
Inode = 2 , Parent = 1 , Type = 0 , Name = c

Please enter inode number of directory where you want to create file or (sub)directory= 1

Enter file or (sub) directory name = d

File or (sub) Directory d added

Enter: 1 to Create file 2 for Directory 3 to Display 4 to Exit
2

For the following file organisation :

File organisation

```
Inode = 0 , Parent = -1 , Type = 1 , Name = rootDir
Inode = 1 , Parent = 0 , Type = 1 , Name = A
Inode = 2 , Parent = 1 , Type = 0 , Name = c
Inode = 3 , Parent = 1 , Type = 0 , Name = d

Please enter inode number of directory where you want to create file or (sub)directory= 0

Enter file or (sub) directory name = b

File or (sub) Directory b added

Enter: 1 to Create file     2 for Directory       3 to Display    4 to Exit
1

For the following file organisation :


File organisation
Inode = 0 , Parent = -1 , Type = 1 , Name = rootDir
Inode = 1 , Parent = 0 , Type = 1 , Name = A
Inode = 2 , Parent = 1 , Type = 0 , Name = c
Inode = 3 , Parent = 1 , Type = 0 , Name = d
Inode = 4 , Parent = 0 , Type = 1 , Name = b

Please enter inode number of directory where you want to create file or (sub)directory= 4

Enter file or (sub) directory name = e

File or (sub) Directory e added

Enter: 1 to Create file     2 for Directory       3 to Display    4 to Exit
1

For the following file organisation :


File organisation
Inode = 0 , Parent = -1 , Type = 1 , Name = rootDir
Inode = 1 , Parent = 0 , Type = 1 , Name = A
Inode = 2 , Parent = 1 , Type = 0 , Name = c
Inode = 3 , Parent = 1 , Type = 0 , Name = d
Inode = 4 , Parent = 0 , Type = 1 , Name = b
Inode = 5 , Parent = 4 , Type = 0 , Name = e

Please enter inode number of directory where you want to create file or (sub)directory= 4

Enter file or (sub) directory name = f

File or (sub) Directory f added

Enter: 1 to Create file     2 for Directory       3 to Display    4 to Exit
3
```

```
File organisation
Inode = 0 , Parent = -1 , Type = 1 , Name = rootDir
Inode = 1 , Parent = 0 , Type = 1 , Name = A
Inode = 2 , Parent = 1 , Type = 0 , Name = c
Inode = 3 , Parent = 1 , Type = 0 , Name = d
Inode = 4 , Parent = 0 , Type = 1 , Name = b
Inode = 5 , Parent = 4 , Type = 0 , Name = e
Inode = 6 , Parent = 4 , Type = 0 , Name = f


Enter: 1 to Create file     2 for Directory       3 to Display    4 to Exit

*/
```