

Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester IV

19 March 2020

UCS1411 - Operating Systems Laboratory

Lab Exercise 12: File Organisation Techniques

Objective:

Develop a C program to implement the following file organisation techniques: a) Single level Directory b) Two level Directory c) Hierarchical Structure d) DAG

Code:

Q.To write a C program to implement the mentioned file organisation techniques.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <time.h>
5 #include <sys/stat.h>
```

```

6 #include<sys/types.h>
7 #include<dirent.h>
8 #include<fcntl.h>
9 #include<unistd.h>
10
11
12 struct Item{
13     char *file_name;
14     int start_address;
15 };
16
17 typedef struct Item File;
18
19 void initialiseFile(File *f){
20     f->file_name=(char*)malloc(sizeof(File));
21     f->start_address=-1;
22 }
23
24 struct base{
25     char *root_name;
26     int no_of_files;
27     File file_list[5];
28     int no_of_dir;
29     struct base* dir_list[5];
30     int start_address;
31 };
32
33 typedef struct base Root;
34
35 void initialiseRoot(Root *r){
36     r->root_name=(char*)calloc(100,sizeof(char));
37     r->no_of_files=0;
38     for(int i=0;i<5;i++){
39         initialiseFile(&r->file_list[i]);
40     }
41     r->no_of_dir=0;
42     for(int i=0;i<5;i++){
43         r->dir_list[i]=(Root*)malloc(sizeof(Root));
44     }
45     r->start_address=-1;
46 }
47
48 //Check if a file exists in a directory
49 int checkDirectory(char *dirname, char *file_name){
50

```

```

51     DIR *dirstream=NULL;
52     struct dirent *dirpointer=NULL;
53
54     dirstream=opendir(dirname);
55     if(dirstream==NULL){
56
57         printf("\n Unable to open directory \n");
58     }
59     else{
60
61         while(NULL != (dirpointer=readdir(dirstream))){
62
63             if(strcmp(dirpointer->d_name,file_name)==0)
64                 return 1;
65
66         }
67         printf("\n");
68
69         closedir(dirstream);
70     }
71     return 0;
72 }
73
74 //Single Level Directory
75 /*
76 Logic:
77 1. Create a directory to act as root. Set its start address
   as 0.
78 2. Create files inside this directory, use random number
   generation for file locations.
79 */
80 void SingleLevelDirectory(){
81     srand(time(0));
82     Root r;
83     initialiseRoot(&r);
84
85     char *dir_name=(char*)calloc(100,sizeof(char));
86     strcat(dir_name,"SingleLevelDirectory");
87     if(mkdir(dir_name,0777) ==-1){
88         printf("\nError:Unable to find directory. \n");
89     }
90     else{
91         strcpy(r.root_name,dir_name);
92         r.start_address=0;
93         int file_action;

```

```

94         do{
95             printf("\n Choose action: \n 1.Create File ");
96             printf("\n 2.List Files \n 0.Back ");
97             printf("\n Your choice: ");scanf("%d",&
file_action);
98             if(file_action==1){
99                 printf("\n Enter name of file: ");
100                 scanf(" %[^\n]",r.file_list[r.no_of_files].
file_name);
101
102                 if(checkDirectory(dir_name,r.file_list[r.
no_of_files].file_name)==0){
103                     char *path_name=(char*)calloc(100,sizeof(
char));
104                     strcat(path_name,dir_name);
105                     strcat(path_name,"/");
106                     strcat(path_name,r.file_list[r.
no_of_files].file_name);
107
108                     int file_fd = open(path_name,O_CREAT|
O_RDWR,0664);
109                     if(file_fd!=-1){
110                         printf("\nError: Unable to create
file. \n");
111                         perror("open error");
112                     }
113                     else{
114                         int flag=0;
115                         int pos;
116                         do{
117                             flag=1;
118                             pos=rand()%49;
119                             pos++;
120                             for(int i=0;i<=r.no_of_files;i++)
{
121                                 if(r.file_list[i].
start_address==pos)
122                                     flag=0;
123                             }
124                         }while(flag==0);
125
126                         r.file_list[r.no_of_files].
start_address=pos;
127
128                         printf("\n File created! \n");

```

```

129         close(file_fd);
130         r.no_of_files++;
131     }
132 }
133 else{
134     printf("\nError: File name already exists
135 . \n");
136 }
137 else if(file_action==2){
138     printf("\n%10s %10s\n", "File name", "Location"
139 );
140     for(int i=0;i<r.no_of_files;i++){
141         printf("\n%10s %10d\n", r.file_list[i].
142 file_name, r.file_list[i].start_address);
143     }
144     else if(file_action){
145         printf("\n Invalid file action. \n");
146     }
147     else;
148 }while(file_action);
149 }
150
151 //Count the number of directories created in a given
152 hierarchy.
153 int countDirectory(Root *r){
154     int count=0;
155     if(r!=NULL){
156         count++;
157         for(int i=0;i<r->no_of_dir;i++){
158             count+=countDirectory(r->dir_list[i]);
159         }
160     }
161     return count;
162 }
163
164 //Two Level Directory
165 /*
166 Logic:
167 1. Create a directory to act as root. Set its start address
168    as 0.
169 2. Create directories inside this directory, count the number

```

```

    of directories,
169 set start address allocating 50 units of memory to each
    directory created.
170 3. Create files inside any of the directories, use random
    number generation for file location,
171 adding it to directory's start address.
172 */
173 void TwoLevelDirectory(){
174     Root *r=(Root*)malloc(sizeof(Root));
175     initialiseRoot(r);
176
177     char *dir_name=(char*)calloc(100,sizeof(char));
178     strcat(dir_name,"TwoLevelDirectory");
179     if(mkdir(dir_name,0777) ==-1){
180         printf("\nError:Unable to find directory. \n");
181     }
182     else{
183         strcpy(r->root_name,dir_name);
184         r->start_address=0;
185         int action;
186         do{
187             printf("\n Choose action: \n 1.Create directory \
n 2.Create File ");
188             printf("\n 3.Search Files \n 4.Display files \n
0.Back ");
189             printf("\n Your choice: ");scanf("%d",&action);
190             if(action==1){
191                 initialiseRoot(r->dir_list[r->no_of_dir]);
192
193                 printf("\n Enter name of directory: ");
194                 scanf(" %[^\\n]",r->dir_list[r->no_of_dir]->
root_name);
195
196                 if(checkDirectory(dir_name,r->dir_list[r->
no_of_dir]->root_name)==1){
197                     printf("\n Error:Directory already exists
. \n");
198                 }
199                 else{
200                     char *path_name=(char*)calloc(100,sizeof(
char));
201                     strcat(path_name,dir_name);strcat(
path_name,"/");
202                     strcat(path_name,r->dir_list[r->no_of_dir
]->root_name);

```

```

203
204         if(mkdir(path_name,0777) ==-1){
205             printf("\nError:Unable to create
directory. \n");
206         }
207         else{
208             int count=countDirectory(r);
209             r->dir_list[r->no_of_dir]->
start_address=(count-1)*50;
210             printf("\n Directory created! \n");
211             r->no_of_dir++;
212         }
213
214     }
215 }
216 else if(action==2){
217     char *file_name=(char*)calloc(100,sizeof(char
));
218     printf("\n Enter name of file: ");
219     scanf("%[^\n]",file_name);
220
221     int dest;
222     printf("\n Choose destination of file: \n 1.
Root ");
223     for(int i=0;i<r->no_of_dir;i++){
224         printf("\n %d.%s ",i+2,r->dir_list[i]->
root_name);
225     }
226     printf("\n 0.back \n Your choice: ");scanf("%
d",&dest);
227     if(dest==1){
228         if(checkDirectory(dir_name,file_name)==0)
{
229             char *path_name=(char*)calloc(100,
sizeof(char));
230             strcat(path_name,dir_name);
231             strcat(path_name,"/");
232             strcat(path_name,file_name);
233
234             int file_fd = open(path_name,O_CREAT|
O_RDWR,0664);
235             if(file_fd==-1){
236                 printf("\nError: Unable to create
file. \n");
237                 perror("open error");

```

```

238         }
239         else{
240             strcpy(r->file_list[r->
no_of_files++].file_name,file_name);
241
242             int flag=0;
243             int pos;
244             do{
245                 flag=1;
246                 pos=rand()%49;
247                 pos++;
248                 for(int i=0;i<=r->no_of_files
; i++){
249                     if(r->file_list[i].
start_address== r->start_address+pos)
250                         flag=0;
251                 }
252             }while(flag==0);
253
254             r->file_list[r->no_of_files].
start_address=r->start_address+pos;
255
256             printf("\n File created! \n");
257             close(file_fd);
258         }
259     }
260     else{
261         printf("\nError: File name already
exists. \n");
262     }
263 }
264 else if(dest>1 && dest-1<=r->no_of_dir){
265     char *path_name=(char*)calloc(100,sizeof(
char));
266     strcat(path_name,dir_name);
267     strcat(path_name,"/");
268     strcat(path_name,r->dir_list[dest-2]->
root_name);
269     if(checkDirectory(path_name,file_name)
==0){
270         strcat(path_name,"/");
271         strcat(path_name,file_name);
272
273         int file_fd = open(path_name,O_CREAT|
O_RDWR,0664);

```



```

274         if(file_fd==-1){
275             printf("\nError: Unable to create
file. \n");
276             perror("open error");
277         }
278         else{
279             strcpy(r->dir_list[dest-2]->
file_list[r->dir_list[dest-2]->no_of_files].file_name,
file_name);
280
281             int flag=0;
282             int pos;
283             do{
284                 flag=1;
285                 pos=rand()%49;
286                 pos++;
287                 for(int i=0;i<=r->no_of_files
;i++){
288                     if(r->dir_list[dest-2]->
file_list[i].start_address== r->dir_list[dest-2]->
start_address+pos)
289                         flag=0;
290                 }
291             }while(flag==0);
292
293             r->dir_list[dest-2]->file_list[r
->dir_list[dest-2]->no_of_files].start_address=r->dir_list
[dest-2]->start_address+pos;
294
295             printf("\n File created! \n");
296             close(file_fd);
297         }
298     }
299     else{
300         printf("\nError: File name already
exists. \n");
301     }
302 }
303 else if(dest){
304     printf("\n Invalid destination. \n");
305 }
306 else;
307 }
308 else if(action==3){
309     char *file_name=(char*) calloc(100, sizeof(char

```

```

));
310         printf("\n Enter name of file: ");
311         scanf(" %[^\\n]",file_name);
312         if(checkDirectory(dir_name,file_name)){
313             printf("\nFound in %s\\n",dir_name);
314         }
315         else{
316             int i=0;
317             char *path_name=(char*)calloc(100,sizeof(
char));
318             strcat(path_name,dir_name);strcat(
path_name,"/");
319             for(i=0;i<r->no_of_dir;i++){
320                 char *path_name=(char*)calloc(100,
sizeof(char));
321                 strcat(path_name,dir_name);strcat(
path_name,"/");
322                 strcat(path_name,r->dir_list[i]->
root_name);
323                 if(checkDirectory(path_name,file_name
)){
324                     printf("\nFound in %s\\n",r->
dir_list[i]->root_name);
325                     break;
326                 }
327             }
328             if(i>=r->no_of_dir){
329                 printf("\n File not found. \\n");
330             }
331         }
332     }
333     else if(action==4){
334         printf("\nRoot: \\n");
335         printf("\n%10s %10s\\n","File name","Location"
);
336         for(int i=0;i<r->no_of_dir;i++){
337             printf("\n%10s %10d\\n",r->dir_list[i]->
root_name,r->dir_list[i]->start_address);
338         }
339         for(int i=0;i<r->no_of_files;i++){
340             printf("\n%10s %10d\\n",r->file_list[i].
file_name,r->file_list[i].start_address);
341         }
342         for(int i=0;i<r->no_of_dir;i++){
343             printf("\n%s: \\n",r->dir_list[i]->

```

```

    root_name);
344         printf("\n%10s %10s\n","File name","
Location");
345
346         for(int j=0;j<r->dir_list[i]->no_of_files
;j++){
347             printf("\n%10s %10d\n",r->dir_list[i
]->file_list[j].file_name,r->dir_list[i]->file_list[j].
start_address);
348         }
349     }
350 }
351     else if(action){
352         printf("\n Invalid action. \n");
353     }
354     else;
355 }while(action);
356 }
357 }
358
359 //Recursive function to create a directory, moving through
the directory structure through the variable path_name
360 int createDirectory(Root *r,char *name_dir,char *path_name){
361     int dest;
362     if(path_name==NULL){
363         printf("\nCurrent path: %s \n",r->root_name);
364     }
365     else{
366         printf("\nCurrent path: %s \n",path_name);
367     }
368     printf("\n Choose destination path of directory: \n 1.%s
",r->root_name);
369     for(int i=0;i<r->no_of_dir;i++){
370         printf("\n %d.%s ",i+2,r->dir_list[i]->root_name);
371     }
372     printf("\n 0.back \n Your choice: ");scanf("%d",&dest);
373
374     if(dest==1){
375         initialiseRoot(r->dir_list[r->no_of_dir]);
376
377         strcat(path_name,r->root_name);
378
379         if(checkDirectory(r->root_name,name_dir)==1){
380             printf("\nError: Directory already exists in %s.
\n",r->root_name);

```

```

381         return 0;
382     }
383     else{
384         strcat(path_name, "/");
385         strcat(path_name, name_dir);
386         if(mkdir(path_name, 0777) == -1){
387             printf("\nError: Unable to create directory. \
n");
388             return 0;
389         }
390         else{
391             strcpy(r->dir_list[r->no_of_dir]->root_name,
name_dir);
392             printf("\n Directory created! \n");
393             r->no_of_dir++;
394             return 1;
395         }
396     }
397
398 }
399 else if(dest>1 && dest-1<=r->no_of_dir){
400     strcat(path_name, r->root_name); strcat(path_name, "/");
401     return createDirectory(r->dir_list[dest-2], name_dir,
path_name);
402 }
403 else if(dest){
404     printf("\nInvalid destination path. \n");
405 }
406 else;
407 return 0;
408
409 }
410
411 //Recursive function to create a file, moving through the
directory structure using the variable path_name
412 int createFile(Root *r, char *file_name, char *path_name){
413     int dest;
414     if(path_name==NULL){
415         printf("\nCurrent path: %s \n", r->root_name);
416     }
417     else{
418         printf("\nCurrent path: %s \n", path_name);
419     }
420     printf("\n Choose destination path of file: \n 1.%s ", r->
root_name);

```

```

421     for(int i=0;i<r->no_of_dir;i++){
422         printf("\n %d.%s ",i+2,r->dir_list[i]->root_name);
423     }
424     printf("\n 0.back \n Your choice: ");scanf("%d",&dest);
425
426     if(dest==1){
427         initialiseRoot(r->dir_list[r->no_of_dir]);
428
429         strcat(path_name,r->root_name);
430
431         if(checkDirectory(r->root_name,file_name)==1){
432             printf("\nError: File already exists in %s. \n",r
->root_name);
433             return 0;
434         }
435         else{
436             strcat(path_name,"/");
437             strcat(path_name,file_name);
438             int file_fd=open(path_name,O_CREAT|O_RDWR,0664);
439             if(file_fd==-1){
440                 printf("\nError: Unable to create file. \n");
441                 perror("open error");
442                 return 0;
443             }
444             else{
445                 strcpy(r->file_list[r->no_of_files].file_name
,file_name);
446                 printf("\n File created! \n");
447                 close(file_fd);
448                 return 1;
449             }
450         }
451     }
452     else if(dest>1 && dest-1<=r->no_of_dir){
453         strcat(path_name,r->root_name);strcat(path_name,"/");
454         return createFile(r->dir_list[dest-2],file_name,
path_name);
455     }
456     else if(dest){
457         printf("\nInvalid destination path. \n");
458     }
459     else;
460     return 0;
461 }
462

```

```

463 //Search for a file in the directory structure, giving it's
    path
464 int searchFile(Root *r,char *file_name,char *path_name){
465     int flag=0;
466     strcat(path_name,r->root_name);
467     if(checkDirectory(path_name,file_name)==1){
468         printf("\nFound in %s\n",r->root_name);
469         flag=1;
470     }
471     else{
472         strcat(path_name,"/");
473         for(int i=0;i<r->no_of_dir;i++)
474             flag+= searchFile(r->dir_list[i],file_name,
path_name);
475     }
476     return (flag>0)?1:0;
477 }
478
479 //Tree Structure
480 /*
481 Logic:
482 1. Create a directory to act as root. Set its start address
    as 0.
483 2. Create directories inside this directory, count the number
    of directories,
484 set start address allocating 50 units of memory to each
    directory created.
485 3. Create files inside any of the directories, use random
    number generation for file location,
486 adding it to directory's start address.
487 */
488 void TreeStructure(){
489     Root *r=(Root*)malloc(sizeof(Root));
490     initialiseRoot(r);
491
492     char *dir_name=(char*)calloc(100,sizeof(char));
493     strcat(dir_name,"TreeStructure");
494     if(mkdir(dir_name,0777) ==-1){
495         printf("\nError:Unable to find directory. \n");
496     }
497     else{
498         strcpy(r->root_name,dir_name);
499         int action;
500         do{
501             printf("\n Choose action: \n 1.Create directory \

```

```

n 2.Create File ");
502     printf("\n 3.Search Files \n 0.Back ");
503     printf("\n Your choice: ");scanf("%d",&action);
504     if(action==1){
505         char *name_dir=(char*)calloc(100,sizeof(char)
);
506         printf("\n Enter name of directory: ");
507         scanf(" %[^\\n]",name_dir);
508
509         char *path_name=(char*)calloc(100,sizeof(char
));
510
511         createDirectory(r,name_dir,path_name);
512     }
513     else if(action==2){
514         char *file_name=(char*)calloc(100,sizeof(char
));
515         printf("\n Enter name of file: ");
516         scanf(" %[^\\n]",file_name);
517
518         char *path_name=(char*)calloc(100,sizeof(char
));
519
520         createFile(r,file_name,path_name);
521     }
522     else if(action==3){
523         char *file_name=(char*)calloc(100,sizeof(char
));
524         printf("\n Enter name of file: ");
525         scanf(" %[^\\n]",file_name);
526
527         char *path_name=(char*)calloc(100,sizeof(char
));
528
529         if(!searchFile(r,file_name,path_name))
530             printf("\nFile not found. ");
531     }
532     else if(action){
533         printf("\n Invalid action. \n");
534     }
535     else;
536 }while(action);
537 }
538 }
539

```

```

540 //Recursive function to set the destination path for a link
    to be created,
541 //moving through the directories using link_path_name
542 int chooseLinkDest(Root *r,char* link_name,char*
    link_path_name){
543
544     int dest;
545     if(link_path_name==NULL){
546         printf("\nCurrent path: %s \n",r->root_name);
547     }
548     else{
549         printf("\nCurrent path: %s \n",link_path_name);
550     }
551
552     printf("\n Choose destination path of file: \n 1.%s ",r->
    root_name);
553     for(int i=0;i<r->no_of_dir;i++){
554         printf("\n %d.%s ",i+2,r->dir_list[i]->root_name);
555     }
556     printf("\n 0.back \n Your choice: ");scanf("%d",&dest);
557
558     if(dest==1){
559
560         strcat(link_path_name,r->root_name);
561
562         if(checkDirectory(r->root_name,link_name)==1){
563             printf("\nError: File already exists in %s. \n",r
    ->root_name);
564             return 0;
565         }
566         else{
567             return 1;
568         }
569     }
570     else if(dest>1 && dest-1<=r->no_of_dir){
571
572         strcat(link_path_name,r->root_name);strcat(
    link_path_name,"/");
573         return chooseLinkDest(r->dir_list[dest-2],link_name,
    link_path_name);
574     }
575     else if(dest){
576         printf("\nInvalid destination path. \n");
577     }
578     else;

```



```

579
580     return 0;
581 }
582
583 //DAG
584 /*
585 Logic:
586 1. Create a directory to act as root. Set its start address
    as 0.
587 2. Create directories inside this directory, count the number
    of directories,
588 set start address allocating 50 units of memory to each
    directory created.
589 3. Create files inside any of the directories, use random
    number generation for file location,
590 adding it to directory's start address.
591 */
592 void DAG(){
593     Root *r=(Root*)malloc(sizeof(Root));
594     initialiseRoot(r);
595
596     char *dir_name=(char*)calloc(100,sizeof(char));
597     strcat(dir_name,"DAG");
598     if(mkdir(dir_name,0777) ==-1){
599         printf("\nError:Unable to find directory. \n");
600     }
601     else{
602         strcpy(r->root_name,dir_name);
603         int action;
604         do{
605             printf("\n Choose action: \n 1.Create directory \
n 2.Create File ");
606             printf("\n 3.Search Files \n 4.Create Link \n 0.
Back ");
607             printf("\n Your choice: ");scanf("%d",&action);
608             if(action==1){
609                 char *name_dir=(char*)calloc(100,sizeof(char)
);
610
611                 printf("\n Enter name of directory: ");
612                 scanf(" %[^\n]",name_dir);
613
614                 char *path_name=(char*)calloc(100,sizeof(char
));
615
616                 createDirectory(r,name_dir,path_name);

```

```

616     }
617     else if(action==2){
618         char *file_name=(char*)calloc(100,sizeof(char
));
619         printf("\n Enter name of file: ");
620         scanf(" %[^\\n]",file_name);
621
622         char *path_name=(char*)calloc(100,sizeof(char
));
623
624         createFile(r,file_name,path_name);
625     }
626     else if(action==3){
627         char *file_name=(char*)calloc(100,sizeof(char
));
628         printf("\n Enter name of file: ");
629         scanf(" %[^\\n]",file_name);
630
631         char *path_name=(char*)calloc(100,sizeof(char
));
632
633         if(!searchFile(r,file_name,path_name))
634             printf("\nFile not found. ");
635     }
636     else if(action==4){
637         char *file_name=(char*)calloc(100,sizeof(char
));
638         printf("\n Enter name of file to be linked: "
);
639         scanf(" %[^\\n]",file_name);
640
641         char *link_name=(char*)calloc(100,sizeof(char
));
642         printf("\n Enter name of link: ");
643         scanf(" %[^\\n]",link_name);
644
645         char *file_path_name=(char*)calloc(100,sizeof
(char));
646         char *link_path_name=(char*)calloc(100,sizeof
(char));
647
648         strcat(file_path_name,"./");
649         strcat(link_path_name,"./");
650
651

```

```

652         if(chooseLinkDest(r,link_name,link_path_name)
){
653         if(searchFile(r,file_name,file_path_name)
){
654
655             strcat(file_path_name,"/");
656             strcat(file_path_name,file_name);
657             strcat(link_path_name,"/");
658             strcat(link_path_name,link_name);
659
660             if(symlink(file_path_name ,
link_path_name)!=0){
661                 perror("link() error");
662                 unlink(file_path_name);
663             }
664             else{
665                 printf("\nLink created! \n");
666             }
667
668             }
669             else{
670                 printf("\nError: File not found. \n")
;
671             }
672         }
673     }
674     else if(action){
675         printf("\n Invalid action. \n");
676     }
677     else;
678 }while(action);
679 }
680 }
681
682 void main(){
683
684     int option;
685     do{
686         printf("\n Choose option: \n 1.Single Level Directory
\n 2.Two Level Directory ");
687         printf("\n 3.Tree Structure \n 4.DAG \n 0.Exit \n
Your choice: ");scanf("%d",&option);
688
689         if(option==1){
690             SingleLevelDirectory();

```

```

691     }
692     else if(option==2){
693         TwoLevelDirectory();
694     }
695     else if(option==3){
696         TreeStructure();
697     }
698     else if(option==4){
699         DAG();
700     }
701     else if(option){
702         printf("\n Invalid option. \n");
703     }
704     else;
705 }while(option);
706 }

```

Output:

```

1
2 Choose option:
3 1.Single Level Directory
4 2.Two Level Directory
5 3.Tree Structure
6 4.DAG
7 0.Exit
8 Your choice: 1
9
10 Choose action:
11 1.Create File
12 2.List Files
13 0.Back
14 Your choice: 1
15
16 Enter name of file: sample
17
18
19 File created!
20
21 Choose action:
22 1.Create File
23 2.List Files
24 0.Back
25 Your choice: 1
26

```

```

27 Enter name of file: sample
28
29 Error: File name already exists.
30
31 Choose action:
32 1.Create File
33 2.List Files
34 0.Back
35 Your choice: 1
36
37 Enter name of file: sample1
38
39
40 File created!
41
42 Choose action:
43 1.Create File
44 2.List Files
45 0.Back
46 Your choice: 2
47
48 File name      Location
49
50     sample           33
51
52     sample1          26
53
54 Choose action:
55 1.Create File
56 2.List Files
57 0.Back
58 Your choice: 0
59
60 Choose option:
61 1.Single Level Directory
62 2.Two Level Directory
63 3.Tree Structure
64 4.DAG
65 0.Exit
66 Your choice: 2
67
68 Choose action:
69 1.Create directory
70 2.Create File
71 3.Search Files

```

```
72 4.Display files
73 0.Back
74 Your choice: 1
75
76 Enter name of directory: D1
77
78
79 Directory created!
80
81 Choose action:
82 1.Create directory
83 2.Create File
84 3.Search Files
85 4.Display files
86 0.Back
87 Your choice: 2
88
89 Enter name of file: sample
90
91 Choose destination of file:
92 1.Root
93 2.D1
94 0.back
95 Your choice: 1
96
97
98 File created!
99
100 Choose action:
101 1.Create directory
102 2.Create File
103 3.Search Files
104 4.Display files
105 0.Back
106 Your choice: 2
107
108 Enter name of file: sample1
109
110 Choose destination of file:
111 1.Root
112 2.D1
113 0.back
114 Your choice: 2
115
116
```

```

117 File created!
118
119 Choose action:
120 1.Create directory
121 2.Create File
122 3.Search Files
123 4.Display files
124 0.Back
125 Your choice: 3
126
127 Enter name of file: sample
128
129 Found in TwoLevelDirectory
130
131 Choose action:
132 1.Create directory
133 2.Create File
134 3.Search Files
135 4.Display files
136 0.Back
137 Your choice: 3
138
139 Enter name of file: sample1
140
141
142 Found in D1
143
144 Choose action:
145 1.Create directory
146 2.Create File
147 3.Search Files
148 4.Display files
149 0.Back
150 Your choice: 4
151
152 Root:
153
154 File name      Location
155
156           D1           50
157
158       sample           17
159
160 D1:
161

```

```

162 File name      Location
163
164     sample1          73
165
166
167 Choose action:
168 1.Create directory
169 2.Create File
170 3.Search Files
171 4.Display files
172 0.Back
173 Your choice: 0
174
175 Choose option:
176 1.Single Level Directory
177 2.Two Level Directory
178 3.Tree Structure
179 4.DAG
180 0.Exit
181 Your choice: 3
182
183 Choose action:
184 1.Create directory
185 2.Create File
186 3.Search Files
187 0.Back
188 Your choice: 1
189
190 Enter name of directory: D1
191
192 Current path:
193
194 Choose destination path of directory:
195 1.TreeStructure
196 0.back
197 Your choice: 1
198
199
200 Directory created!
201
202 Choose action:
203 1.Create directory
204 2.Create File
205 3.Search Files
206 0.Back

```



```

207 Your choice: 1
208
209 Enter name of directory: D2
210
211 Current path:
212
213 Choose destination path of directory:
214 1.TreeStructure
215 2.D1
216 0.back
217 Your choice: 2
218
219 Current path: TreeStructure/
220
221 Choose destination path of directory:
222 1.D1
223 0.back
224 Your choice: 1
225
226
227 Directory created!
228
229 Choose action:
230 1.Create directory
231 2.Create File
232 3.Search Files
233 0.Back
234 Your choice: 2
235
236 Enter name of file: sample
237
238 Current path:
239
240 Choose destination path of file:
241 1.TreeStructure
242 2.D1
243 0.back
244 Your choice: 1
245
246
247 File created!
248
249 Choose action:
250 1.Create directory
251 2.Create File

```

```

252 3.Search Files
253 0.Back
254 Your choice: 2
255
256 Enter name of file: sample1
257
258 Current path:
259
260 Choose destination path of file:
261 1.TreeStructure
262 2.D1
263 0.back
264 Your choice: 2
265
266 Current path: TreeStructure/
267
268 Choose destination path of file:
269 1.D1
270 2.D2
271 0.back
272 Your choice: 1
273
274
275 File created!
276
277 Choose action:
278 1.Create directory
279 2.Create File
280 3.Search Files
281 0.Back
282 Your choice: 2
283
284 Enter name of file: sample2
285
286 Current path:
287
288 Choose destination path of file:
289 1.TreeStructure
290 2.D1
291 0.back
292 Your choice: 2
293
294 Current path: TreeStructure/
295
296 Choose destination path of file:

```

```
297 1.D1
298 2.D2
299 0.back
300 Your choice: 2
301
302 Current path: TreeStructure/D1/
303
304 Choose destination path of file:
305 1.D2
306 0.back
307 Your choice: 1
308
309
310 File created!
311
312 Choose action:
313 1.Create directory
314 2.Create File
315 3.Search Files
316 0.Back
317 Your choice: 3
318
319 Enter name of file: sample
320
321 Found in TreeStructure
322
323 Choose action:
324 1.Create directory
325 2.Create File
326 3.Search Files
327 0.Back
328 Your choice: 3
329
330 Enter name of file: sample2
331
332
333
334 Found in D2
335
336 Choose action:
337 1.Create directory
338 2.Create File
339 3.Search Files
340 0.Back
341 Your choice: 3
```

```

342
343 Enter name of file: sample1
344
345
346 Found in D1
347
348 Choose action:
349 1.Create directory
350 2.Create File
351 3.Search Files
352 0.Back
353 Your choice: 0
354
355 Choose option:
356 1.Single Level Directory
357 2.Two Level Directory
358 3.Tree Structure
359 4.DAG
360 0.Exit
361 Your choice: 4
362
363 Choose action:
364 1.Create directory
365 2.Create File
366 3.Search Files
367 4.Create Link
368 0.Back
369 Your choice: 1
370
371 Enter name of directory: D1
372
373 Current path:
374
375 Choose destination path of directory:
376 1.DAG
377 0.back
378 Your choice: 1
379
380
381 Directory created!
382
383 Choose action:
384 1.Create directory
385 2.Create File
386 3.Search Files

```

```

387 4.Create Link
388 0.Back
389 Your choice: 2
390
391 Enter name of file: sample
392
393 Current path:
394
395 Choose destination path of file:
396 1.DAG
397 2.D1
398 0.back
399 Your choice: 2
400
401 Current path: DAG/
402
403 Choose destination path of file:
404 1.D1
405 0.back
406 Your choice: 1
407
408
409 File created!
410
411 Choose action:
412 1.Create directory
413 2.Create File
414 3.Search Files
415 4.Create Link
416 0.Back
417 Your choice: 3
418
419 Enter name of file: sample
420
421
422 Found in D1
423
424 Choose action:
425 1.Create directory
426 2.Create File
427 3.Search Files
428 4.Create Link
429 0.Back
430 Your choice: 4
431

```

```
432 Enter name of file to be linked: sample
433
434 Enter name of link: link
435
436 Current path: ./
437
438 Choose destination path of file:
439 1.DAG
440 2.D1
441 0.back
442 Your choice: 1
443
444
445
446 Found in D1
447
448 Link created!
449
450 Choose action:
451 1.Create directory
452 2.Create File
453 3.Search Files
454 4.Create Link
455 0.Back
456 Your choice: 0
457
458 Choose option:
459 1.Single Level Directory
460 2.Two Level Directory
461 3.Tree Structure
462 4.DAG
463 0.Exit
464 Your choice: 0
```
