|  |  |
| --- | --- |
| Exno: | PIPE PROCESSING |
| **Date:** |

**PROGRAM :( PIPE PROCESSING)**

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define MSG\_LEN 64 int main(){

int result;

int fd[2];

char message[MSG\_LEN]; char recvd\_msg[MSG\_LEN]; result = pipe (fd);

//Creating a pipe//fd[0] is for reading and fd[1] is for writing if (result < 0)

{

perror("pipe ");

exit(1);

}

strncpy(message,"Linux World!! ",MSG\_LEN); result=write(fd[1],message,strlen(message));

if (result < 0)

{perror("write");

exit(2);}

strncpy(message,"Understanding ",MSG\_LEN); result=write(fd[1],message,strlen(message));

if (result < 0)

{

error("write");

exit(2);

}

strncpy(message,"Concepts of ",MSG\_LEN); result=write(fd[1],message,strlen(message)); if (result < 0)

{

Perror ("write");

Exit (2);

}

Strncpy (message ,"Piping ", MSG\_LEN); result=write( fd [1], message , strlen (message)); if (result < 0)

{

perror("write");

exit(2);

}

result=read (fd[0],recvd\_msg,MSG\_LEN);

if (result < 0)

{

perror("read");

exit(3);

}

printf("%s\n",recvd\_msg); return 0;}

**OUTPUT:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Exno:** | **PRODUCER-CONSUMER PROBLEM USING SEMOPHERES** |
| **Date:** |

**PROGRAM: (PRODUCER-CONSUMER PROBLEM)**

#include<stdio.h>

void main()

{

int buffer[10], bufsize, in, out, produce, consume, choice=0;

in = 0; out = 0; bufsize = 10;

while(choice !=3)

{

printf("\n1. Produce \t 2. Consume \t3. Exit");

printf("\nEnter your choice: =");

scanf("%d", &choice);

switch(choice)

{

case 1: if((in+1)%bufsize==out)

printf("\nBuffer is Full");

else

{

printf("\nEnter the value: "); scanf("%d", &produce); buffer[in] = produce;

in = (in+1)%bufsize;

}

break;

case 2: if(in == out)

printf("\nBuffer is Empty");

else

{

consume = buffer[out];

printf("\nThe consumed value is %d", consume);

out = (out+1)%bufsize;

}

break;

} } }

**OUTPUT:**

|  |
| --- |
| 1. Produce 2. Consume 3. Exit  Enter your choice: 2  Buffer is Empty  1. Produce 2. Consume 3. Exit  Enter your choice: 1  Enter the value: 100  1. Produce 2. Consume 3. Exit  Enter your choice: 2  The consumed value is 100  1. Produce 2. Consume 3. Exit  Enter your choice: 3 |

**RESULT:**

|  |  |
| --- | --- |
| **Exno:** | **FIRST FIT&BEST FIT**  **MEMORY MANAGEMENT** |
| **Date:** |

**PROGRAM: (FIRST FIT MEMORY MANAGEMENT)**

#include<stdio.h>

int main()

{

static int block\_arr[10], file\_arr[10];

int fragments[10], blocks[10], files[10];

int m, n, number\_of\_blocks, number\_of\_files, temp;

printf("\nEnter the Total Number of Blocks:\t");

scanf("%d", &number\_of\_blocks);

printf("Enter the Total Number of Files:\t");

scanf("%d", &number\_of\_files);

printf("\nEnter the Size of the Blocks:\n");

for(m = 0; m < number\_of\_blocks; m++)

{

printf("Block No.[%d]:\t", m + 1);

scanf("%d", &blocks[m]);

}

printf("Enter the Size of the Files:\n");

for(m = 0; m < number\_of\_files; m++)

{

printf("File No.[%d]:\t", m + 1);

scanf("%d", &files[m]);

}

for(m = 0; m < number\_of\_files; m++)

{for(n = 0; n < number\_of\_blocks; n++)

{if(block\_arr[n] != 1)

{temp = blocks[n] - files[m];

if(temp >= 0)

{

file\_arr[m] = n;

break;

}}}

fragments[m] = temp;

block\_arr[file\_arr[m]] = 1;

}

printf("\nFile Number\tBlock Number\tFile Size\tBlock Size\tFragment");

for(m = 0; m < number\_of\_files; m++)

{

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", m, file\_arr[m], files[m], blocks[file\_arr[m]], fragments[m]);

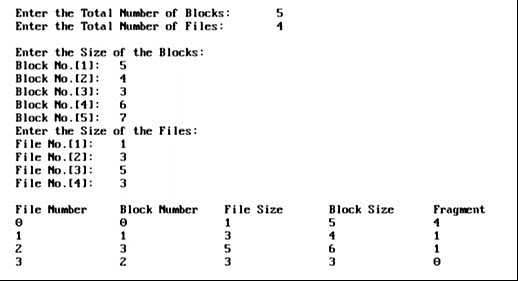
}

printf("\n");

return 0;

}

**Output:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Exno:** | **FILE MANIPULATION-I** |
| **Date:** |

**PROGRAM: (FILE MANIPULATION-I)**

#include <dirent.h>

#include <stdio.h>

int main(void)

{

DIR \*d;

struct dirent \*dir;

d = opendir(".");

if (d)

{

while ((dir = readdir(d)) != NULL)

{

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

}

closedir(d);

}

return(0);

}

**OUTPUT:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Exno:** | **FILE MANIPULATION-II** |
| **Date:** |

**PROGRAM :( FILE MANIPULATION-II)**

#include<stdio.h>

#include<sys/stat.h>

#include<time.h>

main(int ag,char\*arg[])

{

char buf[100]; struct stat s; int fd1,fd2,n;

fd1=open(arg[1],0); fd2=creat(arg[2],0777); stat(arg[2],&s); if(fd2==-1)

printf("ERROR IN CREATION");

while((n=read(fd1,buf,sizeof(buf)))>0)

{

if(write(fd2,buf,n)!=n)

{

close(fd1);

close(fd2);

}

}

printf("\t\n UID FOR FILE.......>%d \n FILE ACCESS TIME.....>%s \n FILE MODIFIED TIME........>%s \n FILE I-NODE NUMBER......>%d \n PERMISSION FOR FILE.....>%o\n\n",s.st\_uid,ctime(&s.st\_atime),ctime(&s.st\_mt ime),s.st\_mode);

close(fd1);

close(fd2);

}

**Output:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Exno:** | **IMPLEMENTATION OF TWO-LEVEL FILE ORGANIZATION TECHNIQUE** |
| **Date:** |

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<string.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;

typedef struct

{

char from[20];

char to[20];

}link;

link L[10];

int nofl;

node \*root;

void main()

{

int gd=DETECT,gm;

root=NULL;

clrscr();

create(&root,0,"root",0,639,320);

read\_links();

clrscr();

initgraph(&gd,&gm,"c:\\Turboc3\\BGI");

draw\_link\_lines();

display(root);

getch();

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("\nEnter name of dir/file(under %s):",dname);

fflush(stdin);

gets((\*root)->name);

printf("\nEnter 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("\nNo 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;

}

}

read\_links()

{

int i;

printf("\nHow many Links:");

scanf("%d",&nofl);

for(i=0;i<nofl;i++)

{

printf("\nFile/Dir:");

fflush(stdin);

gets(L[i].from);

printf("\nUser Name:");

fflush(stdin);

gets(L[i].to);

}

}

draw\_link\_lines()

{

int i,x1,y1,x2,y2;

for(i=0;i<nofl;i++)

{

search(root,L[i].from,&x1,&y1);

search(root,L[i].to,&x2,&y2);

setcolor(LIGHTGREEN);

setlinestyle(3,0,1);

line(x1,y1,x2,y2);

setcolor(YELLOW);

setlinestyle(0,0,1);

}

}

search(node \*root,char \*s,int \*x,int \*y)

{

int i;

if(root!=NULL)

{

if(strcmpi(root->name,s)==0)

{

\*x=root->x;

\*y=root->y;

return;

}

else

{

for(i=0;i<root->nc;i++)

search(root->link[i],s,x,y);

}}}

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);

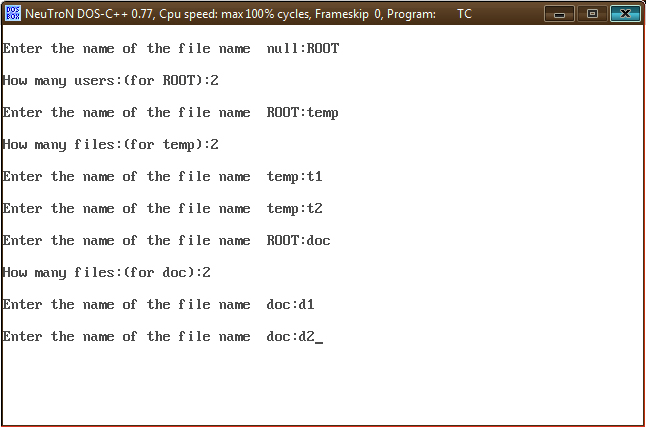
for(i=0;i<root->nc;i++)

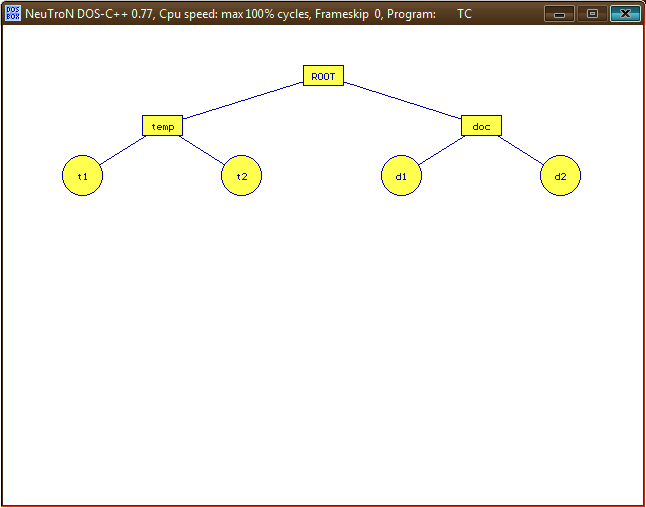
{

display(root->link[i]);

}}}

**OUTPUT:**

****

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex no:** | **IMPLEMENTATION OF DAG FILE ORGANIZATION TECHNIQUE** |
| **Date:** |

**PROGRAM:**

#include<stdio.h>

#include<conio.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;

clrscr();

create(&root,0,"null",0,639,320);

clrscr();

initgraph(&gd,&gm,"c:\\Turboc3\\BGI");

display(root);

getch();

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("\nEnter the name of the file name %s:",dname);

fflush(stdin);

gets((\*root)->name);

if(lev==0 || lev==1)

(\*root)-> ftype=1;

else

(\*root)->ftype=2;

(\*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)

{

if(lev==0 || lev==1)

{

if((\*root)->level==0)

printf("\nHow many users:");

else

printf("\nHow many files:");

printf("(for %s):",(\*root)->name);

scanf("%d",&(\*root)->nc);

}

else

(\*root)->nc=0;

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++)

{

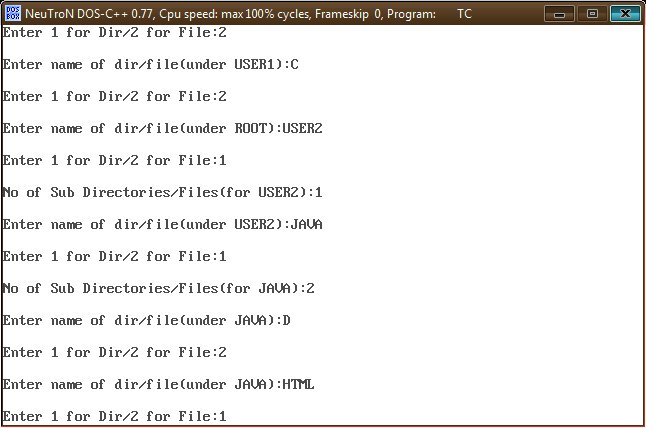
display(root->link[i]);

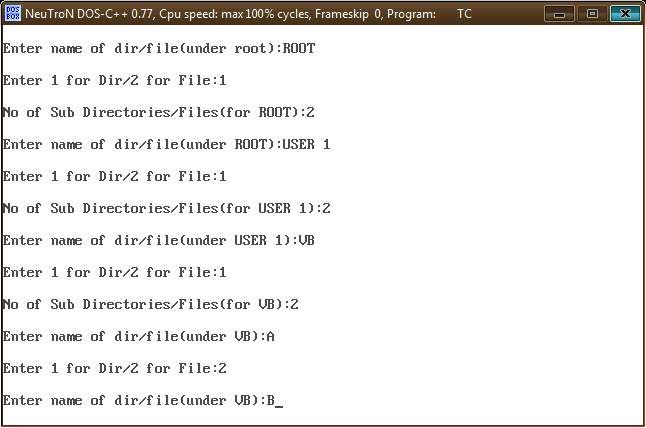
}

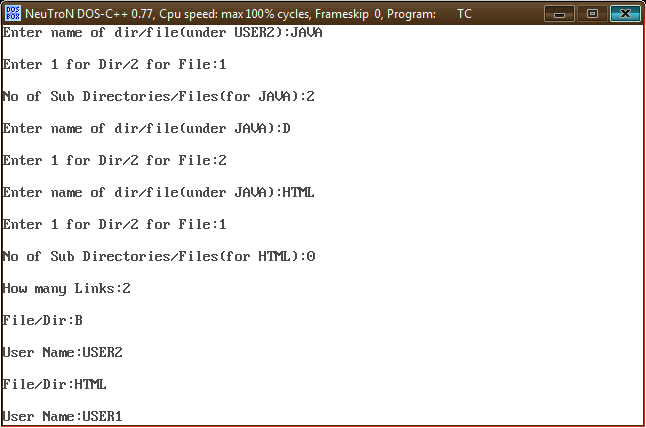
}

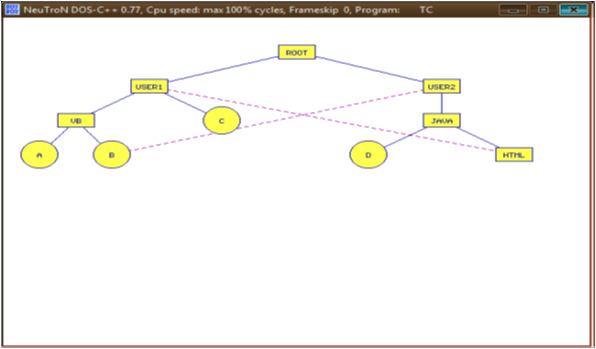
}

**OUTPUT:**

****

****

****

****

**RESULT:**

|  |  |
| --- | --- |
| **Exno:** | **Implementationof Stop and Wait Protocoland SlidingWindowProtocol** |
| **Date:** |

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm,count,i=0,j,mid,cir\_x;

char fname[10][20];

clrscr();

initgraph(&gd,&gm,"c:\\Turboc3\\BGI");

cleardevice();

setbkcolor(BLUE);

printf("enter number of files:");

scanf("%d",&count);

if(i<count)

{

cleardevice();

setbkcolor(6);

printf("enter %d file name:",i+1);

scanf("%s",fname[i]);

setfillstyle(1,MAGENTA);

mid=640/count;

cir\_x=mid/3;

bar3d(270,100,370,150,0,0);

settextstyle(2,0,4);

settextjustify(1,1);

outtextxy(320,125,"root directory");

setcolor(10);

i++;

for(j=0;j<i;j++,cir\_x+=mid)

{

line(320,150,cir\_x,250);

fillellipse(cir\_x,250,30,30);

outtextxy(cir\_x,250,fname[j]);

}

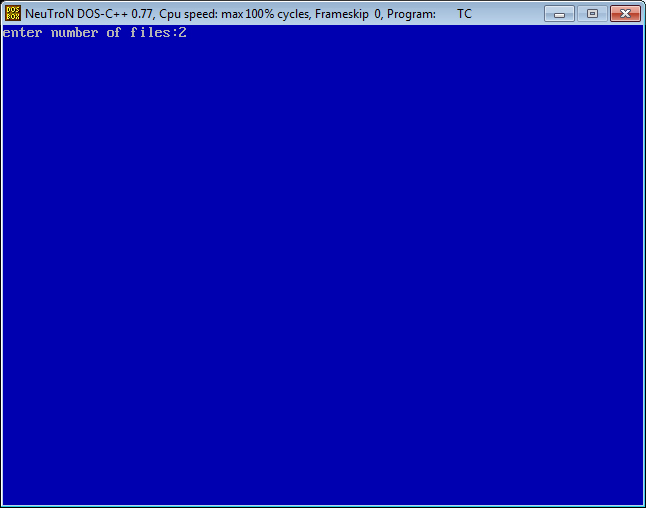
}

getch();

closegraph();

}

**OUTPUT:**

****

****

****

**RESULT:**