SSN COLLEGE OF ENGINEERING, KALAVAKKAM (An Autonomous Institution, Affiliated to Anna University, Chennai) Department of Computer Science and Engineering UCS1411 – Operating Systems Laboratory II Year CSE - B Section ( IV Semester) Academic Year 2019-20

Exercise 12-File Allocation Techniques

Name:Prathyush.S

Class:CSE-B

Register No.:185001112

***Program:-***

#include<stdio.h>

#include<stdlib.h>

typedef struct file {

char n[10];

int s, b;

int i;

}File;

typedef struct indexblock {

int blockid;

int blocks[100];

} IndexBlock;

IndexBlock index1[100];

File files[100];

int freeb[100], mem, bsize, n, nf, c, r, f;

void print(int i)

{

printf("Index block no.: %d\nBlock table:\n", index1[i].blockid);

for(int j = 0; j < files[i].b; j++) {

printf("%d ", index1[i].blocks[j]);

}

}

int main()

{

printf("Enter size of memory in KB: ");

scanf("%d", &mem);

printf("Enter size of block in KB: ");

scanf("%d", &bsize);

printf("No. of blocks = %d\n", mem/bsize);

n = mem/bsize;

nf = n;

for(int i = 0; i <= n/3; i++) {

r = random()%n;

if(freeb[r] == 1) {

i--;

}

else {

freeb[r] = 1;

nf--;

}

}

printf("Free blocks:\n");

for(int i = 0; i < n; i++) {

if(freeb[i] == 0) printf("%d ", i);

}

printf("\nEnter no. of files: ");

scanf("%d", &f);

for(int i = 0; i < f; i++) {

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

scanf("%s", files[c].n);

printf("Enter size in KB: ");

scanf("%d", &files[c].s);

files[c].b = files[c].s/bsize;

if(files[c].s\*1.0/bsize > files[c].b) (files[c].b)++;

if(files[c].b + 1 > nf) {

printf("Can't allocate!\n");

i--;

}

else {

do {

r = random()%n;

}while(freeb[r] == 1);

index1[c].blockid = r;

files[c].i = r;

freeb[r] = 1;

nf--;

for(int j = 0; j < files[c].b; j++) {

r = random()%n;

if(freeb[r] == 0) {

freeb[r] = 1;

nf--;

index1[c].blocks[j] = r;

}

else j--;

}

c++;

}

if(nf == 0) {

printf("Memory over!\n");

f = c;

break;

}

}

printf("\nFile Allocation:\n");

for(int i = 0; i < f; i++) {

printf("File %s:\n",files[i].n);

print(i);

printf("\n");

}

return 0;

}

***Output:-***

prathyush@ubuntu$./a.out

Enter size of memory in KB: 200

Enter size of block in KB: 4

No. of blocks = 50

Free blocks:

0 1 2 3 4 5 6 7 8 10 14 16 17 19 20 23 24 25 28 29 30 31 32 34 37 38 39 41 44 45

46 47 48

Enter no. of files: 5

Enter name of file 1: file1.txt

Enter size in KB: 7

Enter name of file 2: file2.txt

Enter size in KB: 24

Enter name of file 3: file3.txt

Enter size in KB: 50

Enter name of file 4: file4.txt

Enter size in KB: 2

Enter name of file 5: file5.txt

Enter size in KB: 30

Can't allocate!

Enter name of file 5: file5,# #.txt

Enter size in KB: 10

File Allocation:

File file1.txt:

Index block no.: 17

Block table:

29 32

File file2.txt:

Index block no.: 30

Block table:

23 2 8 19 6 34

File file3.txt:

Index block no.: 37

Block table:

48 24 20 41 46 31 5 25 7 45 14 0 28

File file4.txt:

Index block no.: 38

Block table:

3

File file5.txt:

Index block no.: 1

Block table:

4 10 39

***Program:-***

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct memory {

int block;

int isfree;

char f[50];

};

typedef struct memory memory;

memory disc[100];

int blocksize=4;

void initialise()

{

for(int i=0;i<100;i++)

{

disc[i].block=i;

disc[i].isfree=1;

strcpy(disc[i].f,"---");

}

}

struct element{

int block;

int size;

char f[50];

struct element \*next;

};

typedef struct element element;

element \*table=NULL;

void insertfile(char f[],int size,int block)

{

element \*newnode = (element\*)malloc(sizeof(element));

newnode->next=NULL;

newnode->block = block;

newnode->size = size;

strcpy(newnode->f,f);

if (table == NULL)

table = newnode;

else

{

newnode->next = table;

table = newnode;

}

}

void display()

{

element \*temp = table;

printf("FILE\tBLOCK\tSIZE\n");

while(temp!=NULL)

{

printf("%s\t%d\t%d\n",temp->f,temp->block,temp->size);

temp=temp->next;

}

}

int checkfree(int size)

{

int ind=-1;

int reqblocks=size/blocksize;

if(size/(float)blocksize-reqblocks > 0)

reqblocks++;

for(int i=0;i<100;)

{

if(disc[i].isfree==1)

{

int flag=0;

int j=i;

while(disc[j].isfree==1)

{

if(j-i+1==reqblocks)

{

return i;

}

j++;

}

i+=(j+1);

}

else

i++;

}

return ind;

}

int check(int ind,int size)

{

if(disc[ind].isfree)

{

for(int i=ind;i<ind+size;i++)

{

if(!disc[i].isfree)

return 0;

}

}

return 1;

}

int allocate(int size,char name[])

{

int flag =0;

if(checkfree(size)>=0)

{

while(1)

{int possible[100];

int np=0;

int ind = rand()%100;

if(check(ind,size)){

flag=1;

int b= size/blocksize;

if(size/(float)blocksize-b>0)

b++;

for (int i=ind;i<=b+ind;i++)

{

disc[i].isfree=0;

strcpy(disc[i].f,name);

}

insertfile(name,b,ind);

break;

}

}

}

else

printf("Memory not available for the file!!!!!!! \n");

return flag==0?0:1;

}

int main()

{

initialise();

int ch;

char f[50];

int size;

int block;

while(1)

{

printf("1.Allocate\n2.Display\n3.Exit\nEnter your choice: ");

scanf("%d",&ch);

if(ch==1)

{

printf("Enter the file name: ");

scanf("%s",f);

printf("Enter the size of the file: ");

scanf("%d",&size);

allocate(size,f);

}

else if(ch==2)

{

display();

}

else

break;

}

}

***Output:-***

[prathyush@ubuntu $./a.out](mailto:prathyush@ubuntu$./a.out)

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file1.txt

Enter the size of the file: 300# #

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file2.txt

Enter the size of the file: 100

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file3.txt

Enter the size of the file: 50

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file4.txt

Enter the size of the file: 100

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file5.txt

Enter the size of the file: 50

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file6.txt

Enter the size of the file: 100

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file7.txt

Enter the size of the file: 150

1.Allocate

2.Display

3.Exit

Enter your choice: 1

Enter the file name: file8.txt

Enter the size of the file: 3000

Memory not available for the file!!!!!!!

1.Allocate

2.Display

3.Exit

Enter your choice: 2

FILE BLOCK SIZE

file7.txt 36 38

file6.txt 26 25

file5.txt 40 13

file4.txt 26 25

file3.txt 27 13

file2.txt 21 25

file1.txt 15 8

1.Allocate

2.Display

3.Exit

Enter your choice: 3

***Program:-***

#include<stdio.h>

#include<stdlib.h>

typedef struct list {

int block;

struct list \*next;

}List;

typedef struct file {

char n[10];

int s, b;

List\* head;

}File;

File files[100];

int freeb[100], mem, bsize, n, nf, c, r, f;

List\* newnode(int n)

{

List\* p = (List\*)malloc(sizeof(List));

p->block = n;

p->next = NULL;

return p;

}

void print(List\* head)

{

for(List\* t = head; t != NULL; t = t->next)

{

printf("%d",t->block);

if(t->next == NULL) printf("\n");

else printf("->");

}

}

int main()

{

printf("Enter size of memory in KB: ");

scanf("%d", &mem);

printf("Enter size of block in KB: ");

scanf("%d", &bsize);

printf("No. of blocks = %d\n", mem/bsize);

n = mem/bsize;

nf = n;

for(int i = 0; i <= n/3; i++) {

r = random()%n;

if(freeb[r] == 1) {

i--;

}

else {

freeb[r] = 1;

nf--;

}

}

printf("Free blocks:\n");

for(int i = 0; i < n; i++) {

if(freeb[i] == 0) printf("%d ", i);

}

printf("\nEnter no. of files: ");

scanf("%d", &f);

for(int i = 0; i < f; i++) {

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

scanf("%s", (files[c].n));

printf("Enter size in KB: ");

scanf("%d", &files[c].s);

files[c].b = files[c].s/bsize;

if(files[c].s\*1.0/bsize > files[c].b) (files[c].b)++;

if(files[c].b > nf) {

printf("Can't allocate!\n");

i--;

}

else {

List \*t, \*p;

for(int j = 0; j < files[c].b; j++) {

r = random()%n;

if(freeb[r] == 0) {

freeb[r] = 1;

nf--;

t = newnode(r);

if(j == 0) {

files[c].head = t;

p = files[c].head;

}

else {

p->next = t;

p = t;

}

}

else j--;

}

c++;

}

if(nf == 0) {

printf("Memory over!\n");

f = c;

break;

}

}

printf("\nFile allocation:\n");

for(int i = 0; i < f; i++) {

printf("File %s:\n",files[i].n);

print(files[i].head);

printf("\n");

}

return 0;

}

***Output:-***

[prathyush@ubuntu $./a.out](mailto:prathyush@ubuntu$./a.out)

Enter size of memory in KB: 500

Enter size of block in KB: 10

No. of blocks = 50

Free blocks:

0 1 2 3 4 5 6 7 8 10 14 16 17 19 20 23 24 25 28 29 30 31 32 34 37 38 39 41 44 45

46 47 48

Enter no. of files: 4

Enter name of file 1: fie# #le1.txt

Enter size in KB: 250

Enter name of file 2: file2/# #.txt

Enter size in KB: 100

Can't allocate!

Enter name of file 2: file3.txt

Enter size in KB: 50

Enter name of file 3: fie# ## #ile2.txt

Enter size in KB: 50

Can't allocate!

Enter name of file 3: file2.txt

Enter size in KB: 25

Memory over!

File allocation:

File file1.txt:

17->29->32->30->23->2->8->19->6->34->37->48->24->20->41->46->31->5->25->7->45-

>14->0->28->38

File file3.txt:

3->1->4->10->39

File file2.txt:

44->47->16