Pritee Parwekar

MEMORY MANAGEMENT ALGORITHMS

(FIRST FIT ; BEST FIT ; WORST FIT)

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

struct process
{
        int pno,psize;
        char pstatus;
};
struct memory
{
        int mno,msize;
        char mstatus,min;
};
int main()
{
        int n,i,j,mint,mext,flag,flag2,min[6],mini,pos,pos1,k,flag1,ch,used;
        printf("\nenter no of processess : ");
        scanf("%d",&n);
        struct process p[n];
        for(i=0;i<n;i++)
        {
                printf("\nprocess %d",i);
                printf("\nenter process no ; process size  :");
                scanf("%d %d",&p[i].pno,&p[i].psize);
        }
        while(1)
        {
                        printf("\n\n1.first fit\n2.best fit\n3.worst fit\n4.exit\n\n");
                        printf("\nenter your choice:");
                        scanf("%d",&ch);
                        mint=0,mext=0,flag=0,used=0,flag2=0;
                        for(i=0;i<n;i++)
                                p[i].pstatus='w';
                        struct memory
m[6]={{1,100,'f','e'},{2,300,'u','e'},{3,500,'f','e'},{4,200,'f','e'},{5,300,'f','e'},{6,600,'f','e'}};
                        switch(ch)
                        {
                                case 1:
                                                printf("\n\t\t***FIRST FIT***");
                                                for(i=0;i<n;i++)
                                                {
                                                        for(j=0;j<6;j++)
                                                        {
                                                                if(p[i].pstatus=='a')break;


                                                        if(m[j].mstatus=='f'&&m[j].msize>=p[i].psize)
                                                                {
```

```
                                        m[j].msize=(m[j].msize)-
(p[i].psize);
                                        m[j].min='i';
                                        p[i].pstatus='a';
                                }
                        }
                }
                break;

        case 2:
                printf("\n\t\t***BEST FIT***");
                for(i=0;i<n;i++)
                        {

                                for(k=0;k<6;k++)
                                        min[k]=-1;
                                flag1=0;
                                for(j=0;j<6;j++)
                                {
                if((m[j].mstatus=='f')&&(m[j].msize>=p[i].psize))
                                        {
                                                min[j]=(m[j].msize)-
(p[i].psize);
                                                pos1=j;pos=pos1;
                                                flag1=1;
                                        }
                                }
                                if(flag1)
                                {
                                        mini=min[pos1];
                                        for(k=pos1;k>=0;k--)
                                        {
                                        if(min[k]<=mini&&min[k]>=0)
                                                {
                                                        mini=min[k];
                                                        pos=k;
                                                }
                                        }
                                        m[pos].msize=mini;
                                        m[pos].min='i';
                                        p[i].pstatus='a';
                                }
                        }
                break;
        case 3:
                printf("\n\t\t***WORST FIT***");
                for(i=0;i<n;i++)
                {
                        for(k=0;k<6;k++)
                                min[k]=-1;
                        flag1=0;
```

```
                        for(j=0;j<6;j++)
                        {
                    if((m[j].mstatus=='f')&&(m[j].msize>=p[i].psize))
                                {
                                        min[j]=(m[j].msize)-(p[i].psize);
                                                pos1=j;pos=pos1;
                                                flag1=1;
                                }
                        }
                        if(flag1)
                        {
                                mini=min[pos1];
                                for(k=pos1;k>=0;k--)
                                {
                                        if(min[k]>=mini&&min[k]>=0)
                                        {
                                                mini=min[k];
                                                pos=k;
                                        }
                                }
                                m[pos].msize=mini;
                                m[pos].min='i';
                                p[i].pstatus='a';
                        }
                    }
                        break;
                case 4:printf("\n\t\t***THANK YOU***\n");exit(0);
                default:printf("\n\nwrong choice");break;
        }
        for(j=0;j<6;j++)
        {
                if(m[j].mstatus=='f')
                {
                        if(m[j].min=='i')
                                mint=mint+m[j].msize;
                        else
                                mext=mext+m[j].msize;
                }
                else
                        used=used+m[j].msize;
        }
        printf("\n\nfree space:");
        printf("\n\tinternal fragmentation %d \n\texternal fragmentation
%d",mint,mext+mint);
        printf("\n\nused space: %d",used);
        printf("\n\nallocated processess: \n");
        for(i=0;i<n;i++)
        {
                if(p[i].pstatus=='a')
                {
                        printf("\tp[%d] with process no %d\n",i,p[i].pno);flag2=1;
                }
```

```
                }
                if(flag2==0)
                        printf("none");
                printf("\n\nwaiting processess: \n");
                for(i=0;i<n;i++)
                {
                        if(p[i].pstatus=='w')
                        {
                                printf("\tp[%d] with process no %d\n",i,p[i].pno);flag=1;
                        }
                }
                if(flag==0)
                        printf("none");
                }
        return 0;
}
```

OUTPUT:

```
frames :        3       1

frames :        2       1

page faults : 5
page replacements : 3
kavya@kavya-Inspiron-3521:~/kavya$ geany fifo.c
kavya@kavya-Inspiron-3521:~/kavya$ geany memfits.c
kavya@kavya-Inspiron-3521:~/kavya$ gcc memfits.c
.kavya@kavya-Inspiron-3521:~/kavya$ ./a.out

enter no of processess : 4

process 0
enter process no ; process size  :1 212

process 1
enter process no ; process size  :2 417

process 2
enter process no ; process size  :3 112

process 3
enter process no ; process size  :4 426


1.first fit
2.best fit
3.worst fit
4.exit


enter your choice:1

               ***FIRST FIT***

free space:
        internal fragmentation 359
        external fragmentation 959

used space: 300
```

```
used space: 300

allocated processess:
        p[0] with process no 1
        p[1] with process no 2
        p[2] with process no 3


waiting processess:
        p[3] with process no 4


1.first fit
2.best fit
3.worst fit
4.exit


enter your choice:2

                ***BEST FIT***

free space:
        internal fragmentation 433
        external fragmentation 533

used space: 300

allocated processess:
        p[0] with process no 1
        p[1] with process no 2
        p[2] with process no 3
        p[3] with process no 4


waiting processess:
none

1.first fit
2.best fit
3.worst fit
4.exit
```

```
        p[3] with process no 4


waiting processess:
none

1.first fit
2.best fit
3.worst fit
4.exit


enter your choice:3

                ***WORST FIT***

free space:
        internal fragmentation 359
        external fragmentation 959

used space: 300

allocated processess:
        p[0] with process no 1
        p[1] with process no 2
        p[2] with process no 3


waiting processess:
        p[3] with process no 4


1.first fit
2.best fit
3.worst fit
4.exit


enter your choice:4

                ***THANK YOU***
kavya@kavya-Inspiron-3521:~/kavya$
```