Tlemcen University
Faculty of Science
Computer Science Department
1st Year Engineer

Academic year: 2023-2024
Introduction to operating systems 2

Solution TP n°4: Management memory

Exercise n°1

First-fit is executed using the following algorithm:

```
#include <stdio.h>

int p[10],np,b[10],nb,alloc[10],flag[10],frag[10],fragm=0,i,j;

int main()
{
 printf("\n Enter the no of process:"); scanf("%d",&np);
 printf("\n Enter the no of blocks:"); scanf("%d",&nb);
 printf("\n Enter the size of each process:");
 for(i=0;i<np;i++)
 {
  printf("\nProcess %d:",i); scanf("%d",&p[i]);
 }
 printf("\n Enter the block sizes:");
 for(j=0;j<nb;j++)
   printf("\n Block %d:",j);scanf("%d",&b[j]);
 if(np<=nb)
 {
  printf("\n First Fit\n");
  for(i=0;i<np;i++)
  {
   for(j=0;j<nb;j++)
   {
    if(p[i]<=b[j])
    {
     alloc[j]=p[i];printf("\n\n Alloc[%d]",alloc[j]);
     printf("\n\n Process %d of size %d is allocated in block:%d of size:%d",i,p[i],j,b[j]);
     frag[j]=b[j]-p[j];
     printf("\n\n Fragmentation in %d :%d",j,frag[j]);
     flag[i]=0,b[j]=0;
     break;
    }
   else
    flag[i]=1;
  }
 }

 for(i=0;i<np;i++)
 {
  if(flag[i]!=0)
  printf("\n\n Process %d of size %d is not allocated",i,p[i]);
 }
 for(i=0;i<np;i++)
  fragm=fragm+frag[i];
 printf("\n\n Internal fragmentation is : %d",fragm);
}
return 0;
}
```

Perform a trial run on the following example to check your program:

INPUT

- no of process: 3
- no of blocks: 3
- Size of each process:
  - Process 0: 100
  - Process 1: 150
  - Process 2: 200
- block sizes:
  - Block 0: 300
  - Block 1: 350
  - Block 2: 200

Exercise n°2

1) In this exercise, you are asked to write a program in C that implements the concept of memory management using the "Best fit" algorithm and calculates internal fragmentation.

```
#include <stdio.h>

int p[10],np,b[10],nb,ch,c[10],d[10],alloc[10],flag[10],i,j;

int main()
{
 printf("\n Enter the no of process:"); scanf("%d",&np);
 printf("\nEnter the no of blocks:"); scanf("%d",&nb);
 printf("\nEnter the size of each process:");
 for(i=0;i<np;i++)
 {
  printf("\nProcess %d:",i); scanf("%d",&p[i]);
 }
 printf("\nEnter the block sizes:");
 for(j=0;j<nb;j++)
 {
  printf("\nBlock %d:",j);scanf("%d",&b[j]);
  c[j]=b[j];d[j]=b[j];
 }
 if(np<=nb)
 {
  printf("\n =============  Best Fit ===================\n");
  for(i=0;i<nb;i++)
  {
   for(j=i+1;j<nb;j++)
   {
    if(c[i]>c[j])
    {
     int temp=c[i];
         c[i]=c[j];
         c[j]=temp;
    }
   }
  }
  printf("\n After sorting block sizes:");
  for(i=0;i<nb;i++)
    printf("\n Block %d:%d",i,c[i]);
  for(i=0;i<np;i++)
  {
   for(j=0;j<nb;j++)
   {
    if(p[i]<=c[j])
    {
     alloc[j]=p[i];printf("\n\nAlloc[%d]",alloc[j]);
     printf("\n\nProcess %d of size %d is allocated in block %d of size %d",i,p[i],j,c[j]);
     flag[i]=0,c[j]=0;
     break;
    }
    else flag[i]=1;
  }
 }
 for(i=0;i<np;i++)
 {
  if(flag[i]!=0)
     printf("\n\n Process %d of size %d is not allocated",i,p[i]);
 }
}
 return 0;
}
```

Exercise n°3

```
#include <stdio.h>

int p[10],np,b[10],nb,ch,c[10],d[10],alloc[10],flag[10],i,j;

int main()
{
 printf("\n Enter the no of process:"); scanf("%d",&np);
 printf("\nEnter the no of blocks:"); scanf("%d",&nb);
 printf("\nEnter the size of each process:");
 for(i=0;i<np;i++)
 {
  printf("\nProcess %d:",i); scanf("%d",&p[i]);
 }
 printf("\nEnter the block sizes:");
 for(j=0;j<nb;j++)
 {
  printf("\nBlock %d:",j);scanf("%d",&b[j]);
  c[j]=b[j];d[j]=b[j];
 }
 if(np<=nb)
 {
  printf("\n =============  Worst Fit ===================\n");
  for(i=0;i<nb;i++)
  {
   for(j=i+1;j<nb;j++)
   {
    if(d[i]<d[j])
    {
     int temp=d[i]; d[i]=d[j]; d[j]=temp;
    }
   }
  }
  printf("\n After sorting block sizes:");
  for(i=0;i<nb;i++)
    printf("\n Block %d:%d",i,d[i]);
  for(i=0;i<np;i++)
  {
   for(j=0;j<nb;j++)
   {
    if(p[i]<=d[j])
    {
     alloc[j]=p[i];
     printf("\n\nAlloc[%d]",alloc[j]);
     printf("\n\nProcess %d of size %d is allocated in block %d of size %d",i,p[i],j,d[j]);
     flag[i]=0,d[j]=0;
     break;
    }
    else flag[i]=1;
   }
 }
 for(i=0;i<np;i++)
 {
  if(flag[i]!=0)
    printf("\n\n Process %d of size %d is not allocated",i,p[i]);
 }
 }
return 0;
}
```

Exercise n° 4

1) You are requested to implement paging concept for memory management in which you calculate the physical address from the logical address. Proceed as follows:

```
1. Start the program;
2. Enter the logical memory address;
3. Enter the page table which has offset and page frame;
4. The corresponding physical address can be calculated by:
         $@Physical = frame\ page\ no.* \ Size_{page} + offset$
5. Print the physical address for the corresponding logical address;
6. End
```

2) Perform a trial run on the following example to check your program:

INPUT:

- Enter the memory size: 1000
- Enter the page size :100
- Enter no. of pages required for a process: 6
- Enter pagetable for a process: 1, 4, 2, 7, 3, 0
- Enter logical address.

Dans cet exercice, je vous propose d'utiliser les deux possiblités (voir cours):

- **Adresse logique en format décimal**
  Pour calculer le n° de page, on utilise la fonction div(@logique,taille de page) et pour calculer l'offset on utilise la fonction mod(@logique, taille de page).

  Puis voir le n° de cadre qui correspond au n° de page.

  @physique = n° de cadre * taille de page + offset

- **Adresse logique en format binaire**
  Il faut scinder en deux parties : partie n° de page (les bits de poids le plus fort) et partie offset (bits le poids le plus faible).