# Ex.No: 11 Memory allocation methods for fixed partition using

# Date : 24.04.2021 First fit,Best fit and Worst fit

**Aim:**

# To write a program for memory allocation methods for fixed partition using first fit,best fit and worst fit and to execute it.

# 1.Write a program to execute memory allocation method for fixed partition using first fit

**Algorithm:**

* Start
* Get the number of holes available and get the size of each hole from the user and also get the number of processes and get the size of each process to be allocated.
* A process will be allocated in the first available hole of size which is greater than or equal to that of the process’ size
* If a process’ size is equal to that of the hole’s size then that process completely occupies the hole
* If a process’ size is less than the hole’s size then the process will occupy the needed space and sets the remaining space free so that any other processes could occupy that space.
* So check the above said condition and allocate space for each and every process using a for loop.
* After completion display the output as which are all the processes been allocated space and not allocated space and if allocated display in which hole a particular process is allocated.And display the remaining space in each hole if available.

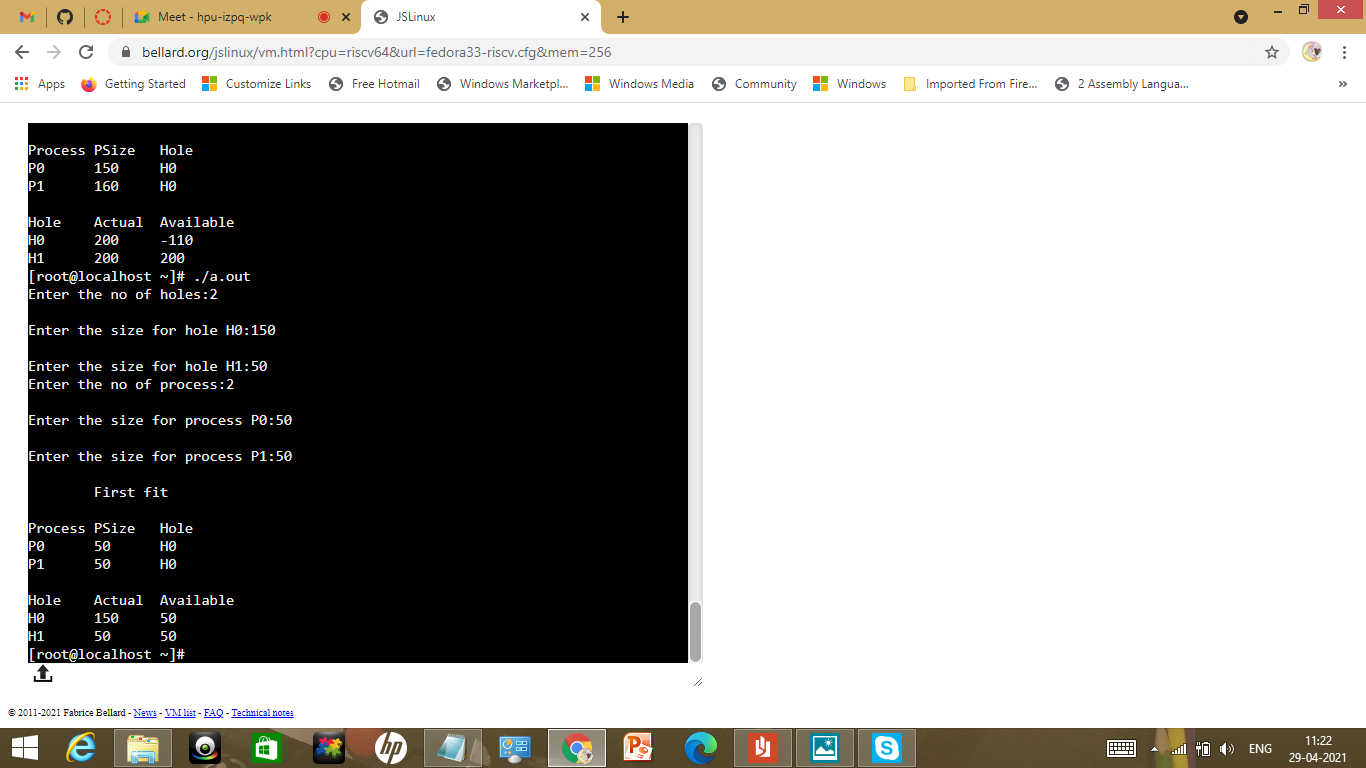
**Program:**

#include<stdio.h>  
struct process  
{  
int size;  
int flag;  
int holeid;  
} p[10];  
struct hole  
{  
int size;  
int actual;  
} h[10];  
void main()  
{  
int i,np,nh,j;  
printf("Enter the no of holes:");  
scanf("%d",&nh);  
for(i=0;i<nh;i++)  
{  
printf("\nEnter the size for hole H%d:",i);  
scanf("%d",&h[i].size);  
h[i].actual=h[i].size;  
}  
printf("Enter the no of process:");  
scanf("%d",&np);  
for(i=0;i<np;i++)  
{  
printf("\nEnter the size for process P%d:",i);  
scanf("%d",&p[i].size); p[i].flag=0;

}

or(i=0;i<np;i++)  
{  
for(j=0;j<nh;j++)  
{  
if(p[i].flag!=1  
{  
if(p[i].size <= h[i].size)  
{  
p[i].flag=1;  
p[i].holeid=j;  
h[j].size -=p[i].size;  
}  
}  
}  
}  
printf("\n\tFirst fit\n");  
printf("\nProcess\tPSize\tHole");  
for(i=0;i<np;i++)  
{  
if(p[i].flag!=1)  
printf("\nP%d\t%d\tNot allocated",i,p[i].size);  
else  
printf("\nP%d\t%d\tH%d",i,p[i].size,p[i].holeid);  
}  
printf("\n\nHole\tActual\tAvailable");  
for(i=0;i<nh;i++)  
printf("\nH%d\t%d\t%d",i,h[i].actual,h[i].size);  
printf("\n");  
}

**Output:**

****

# 2.Write a program to execute memory allocation method for fixed partition using Best fit

**Algorithm:**

* Start
* Get the number of holes available and get the size of each hole from the user and also get the number of processes and get the size of each process to be allocated.
* A process will be allocated in the first available smallest hole of size which is greater than or equal to that of the process’ size
* If a process’ size is equal to that of the hole’s size then that process completely occupies the hole
* If a process’ size is less than the hole’s size then the process will occupy the needed space and sets the remaining space free so that any other processes could occupy that space.
* So check the above said condition and allocate space for each and every process using a for loop.
* After completion display the output as which are all the processes been allocated space and not allocated space and if allocated display in which hole a particular process is allocated.And display the remaining space in each hole if available.

**Program:**

#include<stdio.h>  
struct process  
{  
int size;  
int flag;  
int holeid;  
} p[10];  
struct hole  
{  
int size;  
int actual;

int hid;  
} h[10];  
void main()  
{  
int i,np,nh,j;  
printf("Enter the no of holes:");  
scanf("%d",&nh);  
for(i=0;i<nh;i++)  
{  
printf("\nEnter the size for hole H%d:",i);  
scanf("%d",&h[i].size);  
h[i].actual=h[i].size;  
}  
printf("Enter the no of process:");  
scanf("%d",&np);  
for(i=0;i<np;i++)  
{  
printf("\nEnter the size for process P%d:",i);  
scanf("%d",&p[i].size);

p[i].flag=0;

}

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

{

bsort(h, nh);

for(j=0; j<nh; j++)

{

if(p[i].flag != 1)

{

if(p[i].size <= h[j].size)

{

p[i].flag = 1;

p[i].holeid=h[j].hid;

h[j].size -= p[i].size;

}

}

}

}

printf("\n\tBest fit\n");

printf("\nProcess\tPSize\tHole");

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

{

if(p[i].flag != 1)

printf("\nP%d\t%d\tNot allocated", i, p[i].size);

else

printf("\nP%d\t%d\tH%d", i, p[i].size, p[i].holeid);

}

printf("\n\nHole\tActual\tAvailable");

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

printf("\nH%d\t%d\t%d", h[i].hid, h[i].actual, h[i].size);

printf("\n");

}

void bsort(struct hole bh[], int n)

{

struct hole temp;

int i,j;

for(i=0; i<n-1; i++)

{

for(j=i+1; j<n; j++)

{

if(bh[i].size > bh[j].size)

{

temp = bh[i];

bh[i] = bh[j];

bh[j] = temp;

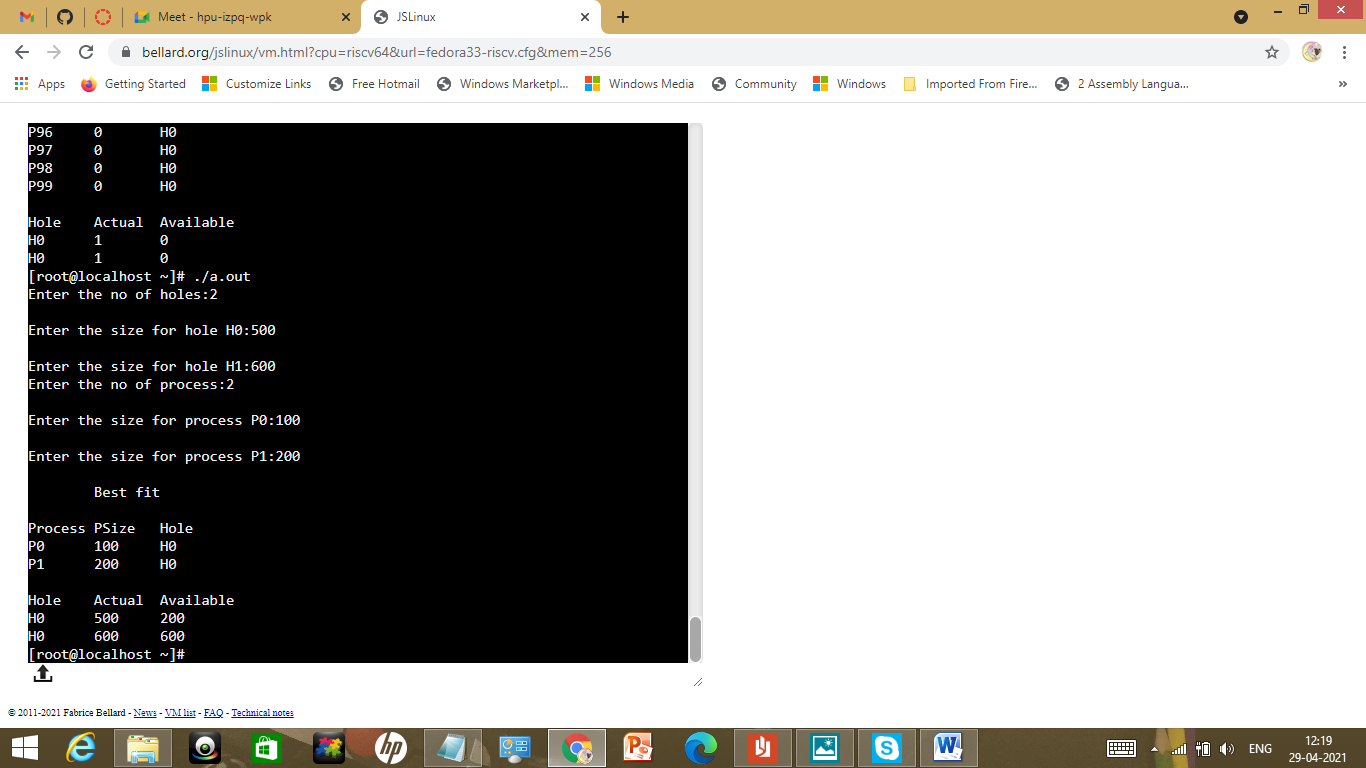
}

}

}

}

**Output:**



# 3.Write a program to execute memory allocation method for fixed partition using Worst fit

**Algorithm:**

* Start
* Get the number of holes available and get the size of each hole from the user and also get the number of processes and get the size of each process to be allocated.
* A process will be allocated in the first available largest hole of size which is greater than or equal to that of the process’ size
* If a process’ size is equal to that of the hole’s size then that process completely occupies the hole
* If a process’ size is less than the hole’s size then the process will occupy the needed space and sets the remaining space free so that any other processes could occupy that space.
* So check the above said condition and allocate space for each and every process using a for loop.
* After completion display the output as which are all the processes been allocated space and not allocated space and if allocated display in which hole a particular process is allocated.And display the remaining space in each hole if available.

**Program:**

#include<stdio.h>  
struct process  
{  
int size;  
int flag;  
int holeid;  
} p[10];  
struct hole  
{  
int size;  
int actual;

int hid;  
} h[10];  
void main()  
{  
int i,np,nh,j;  
printf("Enter the no of holes:");  
scanf("%d",&nh);  
for(i=0;i<nh;i++)  
{  
printf("\nEnter the size for hole H%d:",i);  
scanf("%d",&h[i].size);  
h[i].actual=h[i].size;  
}  
printf("Enter the no of process:");  
scanf("%d",&np);  
for(i=0;i<np;i++)  
{  
printf("\nEnter the size for process P%d:",i);  
scanf("%d",&p[i].size);

p[i].flag=0;

}

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

{

bsort(h, nh);

for(j=0; j<nh; j++)

{

if(p[i].flag != 1)

{

if(p[i].size <= h[j].size)

{

p[i].flag = 1;

p[i].holeid=h[j].hid;

h[j].size -= p[i].size;

}

}

}

}

printf("\n\tBest fit\n");

printf("\nProcess\tPSize\tHole");

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

{

if(p[i].flag != 1)

printf("\nP%d\t%d\tNot allocated", i, p[i].size);

else

printf("\nP%d\t%d\tH%d", i, p[i].size, p[i].holeid);

}

printf("\n\nHole\tActual\tAvailable");

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

printf("\nH%d\t%d\t%d", h[i].hid, h[i].actual, h[i].size);

printf("\n");

}

void bsort(struct hole bh[], int n)

{

struct hole temp;

int i,j;

for(i=0; i<n-1; i++)

{

for(j=i+1; j<n; j++)

{

if(bh[i].size < bh[j].size)

{

temp = bh[i];

bh[i] = bh[j];

bh[j] = temp;

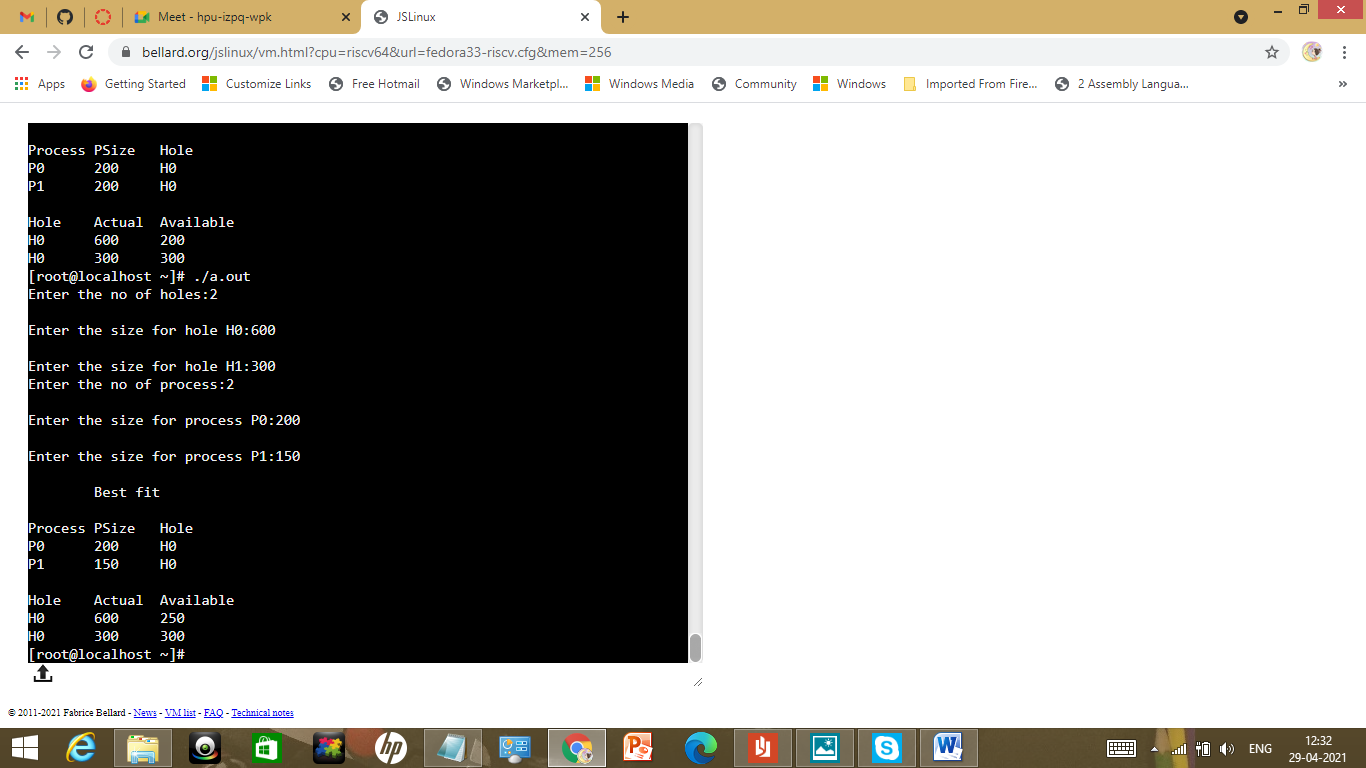
}

}

}

}

**Output:**



|  |  |
| --- | --- |
| Observation(20) |  |
| Record(5) |  |
| Total(25) |  |
| Initial |  |

**Result:**

Thus the memory allocation methods for fixed partition using first fit,worst fit and best fit were executed and outputs were noted.