Exp# 11a First Fit Allocation

Date:

Aim

To allocate memory requirements for processes using first fit allocation.

Memory Management

- The first-fit, best-fit, or worst-fit strategy is used to select a free hole from the set of available holes.

First fit

- Allocate the first hole that is big enough.
- Searching starts from the beginning of set of holes.

Algorithm

1. Declare structures *hole* and *process* to hold information about set of holes and processes respectively.
2. Get number of holes, say *nh*.
3. Get the size of each hole
4. Get number of processes, say *np*.
5. Get the memory requirements for each process.
6. Allocate processes to holes, by examining each hole as follows:
   a. If hole size > process size then
      i. Mark process as allocated to that hole.
      ii. Decrement hole size by process size.
   b. Otherwise check the next from the set of hole
7. Print the list of process and their allocated holes or unallocated status.
8. Print the list of holes, their actual and current availability.
9. Stop

Program

```
/* First fit allocation - ffit.c */

#include <stdio.h>

struct process
{
   int size;
   int flag;
   int holeid;
} p[10];
struct hole
{
   int size;
```

```
   int actual;
} h[10];

main()
{
   int i, np, nh, j;

   printf("Enter the number of Holes : ");
   scanf("%d", &nh);
   for(i=0; i<nh; i++)
   {
      printf("Enter size for hole H%d : ",i);
      scanf("%d", &h[i].size);
      h[i].actual =  h[i].size;
   }

   printf("\nEnter number of process : " );
   scanf("%d",&np);
   for(i=0;i<np;i++)
   {
      printf("enter the size of process P%d : ",i);
      scanf("%d", &p[i].size);
      p[i].flag = 0;
   }

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

```
Enter the number of Holes : 5
Enter size for hole H0 : 100
Enter size for hole H1 : 500
Enter size for hole H2 : 200
Enter size for hole H3 : 300
Enter size for hole H4 : 600

Enter number of process : 4
enter the size of process P0 : 212
enter the size of process P1 : 417
enter the size of process P2 : 112
enter the size of process P3 : 426

        First fit

Process PSize   Hole
P0      212     H1
P1      417     H4
P2      112     H1
P3      426     Not allocated

Hole    Actual  Available
H0      100     100
H1      500     176
H2      200     200
H3      300     300
H4      600     183
```

Result

Thus processes were allocated memory using first fit method.

Exp# 11b Best Fit Allocation

Date:

Aim

To allocate memory requirements for processes using best fit allocation.

Best fit

- Allocate the smallest hole that is big enough.
- The list of free holes is kept sorted according to size in ascending order.
- This strategy produces smallest leftover holes

Algorithm

1. Declare structures *hole* and *process* to hold information about set of holes and processes respectively.
2. Get number of holes, say *nh*.
3. Get the size of each hole
4. Get number of processes, say *np*.
5. Get the memory requirements for each process.
6. Allocate processes to holes, by examining each hole as follows:
    a. Sort the holes according to their sizes in ascending order
    b. If hole size > process size then
        i. Mark process as allocated to that hole.
        ii. Decrement hole size by process size.
    c. Otherwise check the next from the set of sorted hole
7. Print the list of process and their allocated holes or unallocated status.
8. Print the list of holes, their actual and current availability.
9. Stop

Program

```
#include <stdio.h>

struct process
{
   int size;
   int flag;
   int holeid;
} p[10];

struct hole
{
   int hid;
   int size;
   int actual;
} h[10];
```

```
main()
{
   int i, np, nh, j;
   void bsort(struct hole[], int);
   printf("Enter the number of Holes : ");
   scanf("%d", &nh);
   for(i=0; i<nh; i++)
   {
      printf("Enter size for hole H%d : ",i);
      scanf("%d", &h[i].size);
      h[i].actual =  h[i].size;
      h[i].hid = i;
   }
   printf("\nEnter number of process : " );
   scanf("%d",&np);
   for(i=0;i<np;i++)
   {
      printf("enter the size of 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

```
Enter the number of Holes : 5
Enter size for hole H0 : 100
Enter size for hole H1 : 500
Enter size for hole H2 : 200
Enter size for hole H3 : 300
Enter size for hole H4 : 600
Enter number of process : 4
enter the size of process P0 : 212
enter the size of process P1 : 417
enter the size of process P2 : 112
enter the size of process P3 : 426
        Best fit
Process PSize   Hole
P0      212     H3
P1      417     H1
P2      112     H2
P3      426     H4

Hole    Actual  Available
H1      500     83
H3      300     88
H2      200     88
H0      100     100
H4      600     174
```

Result

Thus processes were allocated memory using best fit method.