**Memory Allocation**

1. First Fit
2. Best Fit
3. Worst Fit
4. Next Fit

MMU

Non - Contiguous

Contiguous

Fixed Sized Partitioning Variable Size Partitioning

P1

P2

P3

CPU

Source Code:

#include <stdio.h>

void first\_fit(int blocks[], int block\_count, int processes[], int process\_count)

{

printf("\nFirst Fit Allocation:\n");

int allocated[50];

int i, j;

for (i = 0; i < block\_count; i++)

{

allocated[i] = 0;

}

for (i = 0; i < process\_count; i++)

{

int found = 0;

for (j = 0; j < block\_count; j++)

{

if (allocated[j] == 0 && blocks[j] >= processes[i])

{

blocks[j] -= processes[i];

allocated[j] = 1;

printf("Process %d allocated to Block %d\n", i + 1, j + 1);

found = 1;

break;

}

}

if (!found)

{

printf("Process %d could not be allocated\n", i + 1);

}

}

}

void best\_fit(int blocks[], int block\_count, int processes[], int process\_count)

{

printf("\nBest Fit Allocation:\n");

int allocated[50];

int i, j;

for (i = 0; i < block\_count; i++)

{

allocated[i] = 0;

}

for (i = 0; i < process\_count; i++)

{

int best\_idx = -1;

for (j = 0; j < block\_count; j++)

{

if (allocated[j] == 0 && blocks[j] >= processes[i])

{

if (best\_idx == -1 || blocks[j] < blocks[best\_idx])

{

best\_idx = j;

}

}

}

if (best\_idx != -1)

{

blocks[best\_idx] -= processes[i];

allocated[best\_idx] = 1;

printf("Process %d allocated to Block %d\n", i + 1, best\_idx + 1);

}

else

{

printf("Process %d could not be allocated\n", i + 1);

}

}

}

void worst\_fit(int blocks[], int block\_count, int processes[], int process\_count)

{

printf("\nWorst Fit Allocation:\n");

int allocated[50];

int i, j;

for (i = 0; i < block\_count; i++)

{

allocated[i] = 0;

}

for (i = 0; i < process\_count; i++)

{

int worst\_idx = -1;

for (j = 0; j < block\_count; j++)

{

if (allocated[j] == 0 && blocks[j] >= processes[i])

{

if (worst\_idx == -1 || blocks[j] > blocks[worst\_idx])

{

worst\_idx = j;

}

}

}

if (worst\_idx != -1)

{

blocks[worst\_idx] -= processes[i];

allocated[worst\_idx] = 1;

printf("Process %d allocated to Block %d\n", i + 1, worst\_idx + 1);

}

else

{

printf("Process %d could not be allocated\n", i + 1);

}

}

}

int main()

{

int block\_count, process\_count;

printf("Enter the number of memory blocks: ");

scanf("%d", &block\_count);

printf("Enter the number of processes: ");

scanf("%d", &process\_count);

int blocks[50];

int original\_blocks[50];

int processes[50];

int i;

printf("Enter the sizes of each block:\n");

for (i = 0; i < block\_count; i++)

{

printf("Block %d: ", i + 1);

scanf("%d", &blocks[i]);

original\_blocks[i] = blocks[i];

}

printf("Enter the sizes of each process:\n");

for (i = 0; i < process\_count; i++)

{

printf("Process %d: ", i + 1);

scanf("%d", &processes[i]);

}

first\_fit(blocks, block\_count, processes, process\_count);

for (i = 0; i < block\_count; i++)

{

blocks[i] = original\_blocks[i];

}

best\_fit(blocks, block\_count, processes, process\_count);

for (i = 0; i < block\_count; i++)

{

blocks[i] = original\_blocks[i];

}

worst\_fit(blocks, block\_count, processes, process\_count);

return 0;

}