

Academic Year: 2023-24 Sem: III

Sub: Operating Systems Laboratory SAP ID: 60003220045

Name: Anish Sharma

**EXPERIMENT NO. 05**

**(1) First Fit:**

#include<stdio.h>

void firstFit(int blockSize[], int m, int processSize[], int n) {

int i, j;

int allocation[n]; for(i = 0; i < n; i++) {

allocation[i] = -1; }

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

for (j = 0; j < m; j++) {

if (blockSize[j] >= processSize[i]) {

allocation[i] = j; blockSize[j] -= processSize[i];

break; }

} }

printf("\nProcess No.\tProcess Size\tBlock no.\n"); for (int i = 0; i < n; i++)

{

printf(" %i\t\t\t", i+1); printf("%i\t\t\t\t", processSize[i]); if (allocation[i] != -1)

printf("%i", allocation[i] + 1); else

printf("Not Allocated"); printf("\n");

}

}

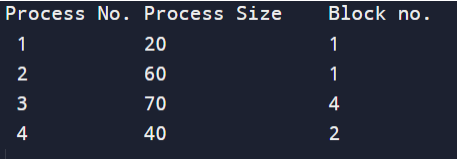
int main() {

int m; int n;

int blockSize[] = {100, 50, 30, 120, 35}; int processSize[] = {20,60,70,40};

m = sizeof(blockSize) / sizeof(blockSize[0]);

n = sizeof(processSize) / sizeof(processSize[0]); firstFit(blockSize, m, processSize, n);

return 0 ; }

**(2) Best Fit** #include <stdio.h>

void implimentBestFit(int blockSize[], int blocks, int processSize[], int proccesses) {

int allocation[proccesses]; int occupied[blocks];

for(int i = 0; i < proccesses; i++){ allocation[i] = -1;

}

for(int i = 0; i < blocks; i++){ occupied[i] = 0;

}

for (int i = 0; i < proccesses; i++) {

int indexPlaced = -1;

for (int j = 0; j < blocks; j++) {

if (blockSize[j] >= processSize[i] && !occupied[j]) {

if (indexPlaced == -1) indexPlaced = j;

else if (blockSize[j] < blockSize[indexPlaced]) indexPlaced = j;

} }

if (indexPlaced != -1) {

allocation[i] = indexPlaced; occupied[indexPlaced] = 1;

} }

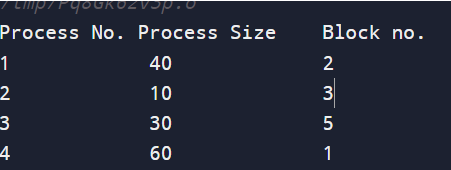
printf("\nProcess No.\tProcess Size\tBlock no.\n"); for (int i = 0; i < proccesses; i++)

{

printf("%d \t\t\t %d \t\t\t", i+1, processSize[i]); if (allocation[i] != -1)

printf("%d\n",allocation[i] + 1); else

printf("Not Allocated\n");



} }

int main() {

int blockSize[] = {100, 50, 30, 120, 35}; int processSize[] = {40, 10, 30, 60};

int blocks = sizeof(blockSize)/sizeof(blockSize[0]);

int proccesses = sizeof(processSize)/sizeof(processSize[0]); implimentBestFit(blockSize, blocks, processSize, proccesses); return 0 ;

}

**(3) Worst Fit:** #include <stdio.h>

void implimentWorstFit(int blockSize[], int blocks, int processSize[], int processes) {

int allocation[processes]; int occupied[blocks];

for(int i = 0; i < processes; i++){ allocation[i] = -1;

}

for(int i = 0; i < blocks; i++){ occupied[i] = 0;

}

for (int i=0; i < processes; i++) {

int indexPlaced = -1; for(int j = 0; j < blocks; j++) {

if(blockSize[j] >= processSize[i] && !occupied[j]) {

if (indexPlaced == -1) indexPlaced = j;

else if (blockSize[indexPlaced] < blockSize[j]) indexPlaced = j;

} }

if (indexPlaced != -1) {

allocation[i] = indexPlaced; occupied[indexPlaced] = 1; blockSize[indexPlaced] -= processSize[i];

} }

printf("\nProcess No.\tProcess Size\tBlock no.\n"); for (int i = 0; i < processes; i++)

{

printf("%d \t\t\t %d \t\t\t", i+1, processSize[i]); if (allocation[i] != -1)

printf("%d\n",allocation[i] + 1); else

printf("Not Allocated\n"); }

}

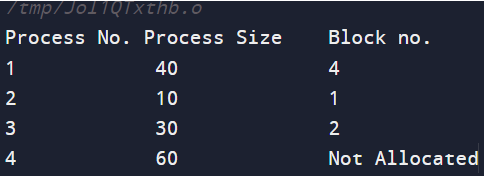
int main()

{

int blockSize[] = {100, 50, 30, 120, 35}; int processSize[] = {40, 10, 30, 60};

int blocks = sizeof(blockSize)/sizeof(blockSize[0]);

int processes = sizeof(processSize)/sizeof(processSize[0]); implimentWorstFit(blockSize, blocks, processSize, processes); return 0;

}