EXP 8 (first fit)

#include <stdio.h> #include <stdlib.h>

#define MAX\_PARTITIONS 10

#define MAX\_PROCESSES 5 typedef struct {

int size; int isFree;

} Partition; typedef struct {

int size;

int allocated;

} Process;

void firstFit(Partition partitions[], Process processes[], int pCount, int partCount) { printf("First Fit:\n");

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

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

if (partitions[j].isFree && partitions[j].size >= processes[i].size) { partitions[j].isFree = 0; // Mark partition as allocated processes[i].allocated = 1; // Mark process as allocated

printf("Process %d of size %d allocated to Partition %d of size %d\n", i, processes[i].size, j, partitions[j].size);

break;

}} }}

void printPartitions(Partition partitions[], int partCount) { printf("\nPartitions Status:\n");

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

printf("Partition %d: Size %d, %s\n", i, partitions[i].size, partitions[i].isFree ? "Free" : "Allocated");

}}

int main() {

Partition partitions[MAX\_PARTITIONS] = {

{100, 1}, {500, 1}, {200, 1}, {300, 1}, {600, 1}

};

Process processes[MAX\_PROCESSES] = {

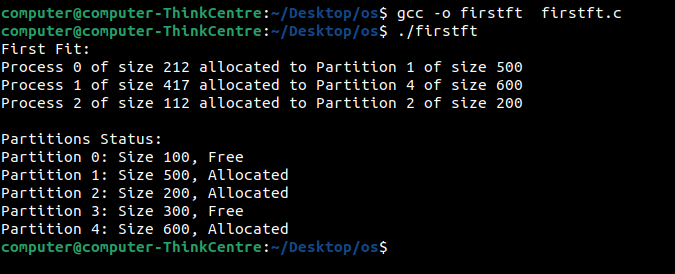
{212, 0}, {417, 0}, {112, 0}, {426, 0}, {500, 0}

};

int partCount = 5; int pCount = 5;

firstFit(partitions, processes, pCount, partCount); printPartitions(partitions, partCount);

return 0;}



(best fit)

#include <stdio.h> #include <stdlib.h>

#define MAX\_PARTITIONS 10

#define MAX\_PROCESSES 5 typedef struct {

int size; int isFree;

} Partition; typedef struct {

int size;

int allocated;

} Process;

void bestFit(Partition partitions[], Process processes[], int pCount, int partCount) { printf("Best Fit:\n");

for (int i = 0; i < pCount; i++) { int bestIndex = -1;

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

if (partitions[j].isFree && partitions[j].size >= processes[i].size) {

if (bestIndex == -1 || partitions[j].size < partitions[bestIndex].size) { bestIndex = j; }}}

if (bestIndex != -1) {

partitions[bestIndex].isFree = 0; // Mark partition as allocated processes[i].allocated = 1; // Mark process as allocated

printf("Process %d of size %d allocated to Partition %d of size %d\n", i, processes[i].size, bestIndex, partitions[bestIndex].size);

} }}

void printPartitions(Partition partitions[], int partCount) { printf("\nPartitions Status:\n");

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

printf("Partition %d: Size %d, %s\n", i, partitions[i].size, partitions[i].isFree ? "Free" : "Allocated");

}}

int main() {

Partition partitions[MAX\_PARTITIONS] = {

{100, 1}, {500, 1}, {200, 1}, {300, 1}, {600, 1}};

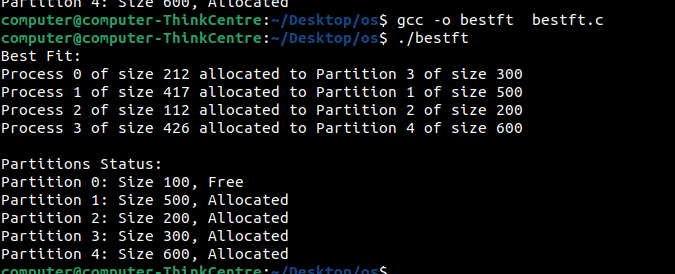
Process processes[MAX\_PROCESSES] = {

{212, 0}, {417, 0}, {112, 0}, {426, 0}, {500, 0} };

int partCount = 5; int pCount = 5;

bestFit(partitions, processes, pCount, partCount); printPartitions(partitions, partCount);

return 0;



(worst fit) #include <stdio.h>

#include <stdlib.h>

#define MAX\_PARTITIONS 10

#define MAX\_PROCESSES 5 typedef struct {

int size; int isFree;

} Partition; typedef struct {

int size;

int allocated;

} Process;

void worstFit(Partition partitions[], Process processes[], int pCount, int partCount) { printf("Worst Fit:\n");

for (int i = 0; i < pCount; i++) { int worstIndex = -1;

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

if (partitions[j].isFree && partitions[j].size >= processes[i].size) {

if (worstIndex == -1 || partitions[j].size > partitions[worstIndex].size) { worstIndex = j; // Find the worst fit partition

}}}

if (worstIndex != -1) {

partitions[worstIndex].isFree = 0; // Mark partition as allocated processes[i].allocated = 1; // Mark process as allocated

printf("Process %d of size %d allocated to Partition %d of size %d\n", i, processes[i].size, worstIndex, partitions[worstIndex].size);} }}

void printPartitions(Partition partitions[], int partCount) { printf("\nPartitions Status:\n");

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

printf("Partition %d: Size %d, %s\n", i, partitions[i].size, partitions[i].isFree ? "Free" : "Allocated"); }} int main() {

Partition partitions[MAX\_PARTITIONS] = {

{100, 1}, {500, 1}, {200, 1}, {300, 1}, {600, 1} };

Process processes[MAX\_PROCESSES] = {

{212, 0}, {417, 0}, {112, 0}, {426, 0}, {500, 0}};

int partCount = 5; int pCount = 5;

worstFit(partitions, processes, pCount, partCount); printPartitions(partitions, partCount);

return 0;}

