FCFS

#include <stdio.h>

int main() {

int at[10], bt[10], wt[10], ft[10], tat[10], i, j, n, temp1, temp2, x = 0;

int twt = 0, ttat = 0;

float awt1, atat1;

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter arrival times:\n");

for(i = 0; i < n; i++) scanf("%d", &at[i]);

printf("Enter burst times:\n");

for(i = 0; i < n; i++) scanf("%d", &bt[i]);

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

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

if(at[i] > at[j]) {

temp1 = at[i]; at[i] = at[j]; at[j] = temp1;

temp2 = bt[i]; bt[i] = bt[j]; bt[j] = temp2;

}

printf("Arrival Time\tBurst Time\n");

for(i = 0; i < n; i++) printf("%d\t\t%d\n", at[i], bt[i]);

printf("Calculating Finishing Time, Waiting Time, Turnaround Time\n");

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

x += bt[i];

ft[i] = x;

wt[i] = ft[i] - at[i] - bt[i];

tat[i] = ft[i] - at[i];

twt += wt[i];

ttat += tat[i];

printf("Process %d: FT=%d, WT=%d, TAT=%d\n", i+1, ft[i], wt[i], tat[i]);

}

awt1 = (float)twt/n;

atat1 = (float)ttat/n;

printf("Average Waiting Time: %.2f\n", awt1);

printf("Average Turnaround Time: %.2f\n", atat1);

return 0;

}

Sjf

#include <stdio.h>

int main() {

int at[10], bt[10], wt[10], ft[10], tat[10];

int i, j, n, temp1, temp2, temp3, x = 0, completed = 0, min\_index;

int twt = 0, ttat = 0, min\_bt;

float awt1, atat1;

int is\_completed[10] = {0};

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter arrival times:\n");

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

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

}

printf("Enter burst times:\n");

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

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

}

printf("Arrival Time\tBurst Time\n");

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

printf("%d\t\t%d\n", at[i], bt[i]);

}

printf("Calculating Finishing Time, Waiting Time, Turnaround Time\n");

while (completed != n) {

min\_bt = 9999;

min\_index = -1;

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

if (at[i] <= x && is\_completed[i] == 0) {

if (bt[i] < min\_bt) {

min\_bt = bt[i];

min\_index = i;

} else if (bt[i] == min\_bt) {

if (at[i] < at[min\_index]) {

min\_index = i;

}

}

}

}

if (min\_index == -1) {

x++; // CPU idle

} else {

x += bt[min\_index];

ft[min\_index] = x;

tat[min\_index] = ft[min\_index] - at[min\_index];

wt[min\_index] = tat[min\_index] - bt[min\_index];

is\_completed[min\_index] = 1;

completed++;

twt += wt[min\_index];

ttat += tat[min\_index];

printf("Process %d: FT=%d, WT=%d, TAT=%d\n", min\_index + 1, ft[min\_index], wt[min\_index], tat[min\_index]);

}

}

awt1 = (float)twt / n;

atat1 = (float)ttat / n;

printf("Average Waiting Time: %.2f\n", awt1);

printf("Average Turnaround Time: %.2f\n", atat1);

return 0;

}

Priority

#include<stdio.h>

#define max 20

main()

{

int bt[max] ,at[max],ft[max],wt[max],tat[max],p[max];

int i,j,n,x=0,y=0,z=0,temp1,temp2,temp3,min=0,k,l;

float t,u;

printf("Enter the processes to be executed:\n");

scanf("%d",&n);

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

{

printf("Enter the burst time for process-%d:\n",i);

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

printf("Enter the arrival time for process-%d:\n",i);

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

printf("Enter the priority of process-%d:\n",i);

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

}

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

{

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

{

if(at[i]>at[j])

{

temp1=at[i];

at[i]=at[j];

at[j]=temp1;

temp2=bt[i];

bt[i]=bt[j];

bt[j]=temp2;

temp3=p[i];

p[i]=p[j];

p[j]=temp3;

}

}

}

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

{

min=min+bt[i];

for(j=i+1;at[j]<=min;j++)

{

for(k=j+1;at[k]<=min;k++)

{

if(p[k]<p[j])

{

temp1=bt[k];

bt[k]=bt[j];

bt[j]=temp1;

temp2=at[k];

at[k]=at[j];

at[j]=temp2;

temp3=p[i];

p[i]=p[j];

p[j]=temp3;

}

}

}

}

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

{

x=x+bt[i];

ft[i]=x;

if(i==1)

wt[i]=y;

else

wt[i]=ft[i-1]-at[i];

tat[i]=bt[i]+wt[i];

}

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

{

y=y+tat[i];

z=z+wt[i];

}

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

{

printf("\nProcess:%d-->at:%d\t bt:%d\t ft:%d\t wt:%d\t tat:%d\t",i,at[i],bt[i],ft[i],wt[i],tat[i]);

}

printf("\nAverage waiting time:%d",z/n);

printf("\nAverage turn around time:%d",y/n);}

Round robin

#include<stdio.h>

int main(){

int at[10], bt[10], wt[10], st[10], tat[10], n, tq;

int i, count=0, swt=0, stat=0, temp, sq=0, j, temp1, temp2;

float awt=0.0, atat=0.0;

printf("Enter number of processes:\n");

scanf("%d", &n);

printf("Enter burst times:\n");

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

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

}

printf("Enter arrival times:\n");

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

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

}

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

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

if(at[i] > at[j]) {

temp1 = at[i];

at[i] = at[j];

at[j] = temp1;

temp2 = bt[i];

bt[i] = bt[j];

bt[j] = temp2;

}

}

}

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

st[i] = bt[i];

printf("Arrival time: %d, Burst time: %d\n", at[i], bt[i]);

}

printf("\nEnter time quantum: ");

scanf("%d", &tq);

// Round Robin Scheduling

while(1) {

count = 0;

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

temp = tq;

if(st[i] == 0) {

count++;

continue;

}

if(st[i] > tq) {

st[i] -= tq;

} else {

temp = st[i];

st[i] = 0;

}

sq += temp;

tat[i] = sq - at[i];

}

if(count == n) {

break;

}

}

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

wt[i] = tat[i] - bt[i];

swt += wt[i];

stat += tat[i];

}

awt = (float)swt / n;

atat = (float)stat / n;

printf("\nProcess No\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, at[i], bt[i], wt[i], tat[i]);

}

printf("\nAverage waiting time: %f\n", awt);

printf("Average turn-around time: %f\n", atat);

return 0;

}

First fit

#include <stdio.h>

void firstFit(int b[], int m, int p[], int n) {

int alloc[10], i, j;

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

alloc[i] = -1;

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

if (b[j] >= p[i]) {

alloc[i] = j;

b[j] -= p[i];

break;

}

}

printf("Process No.\tProcess Size\tBlock No.\n");

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

printf("%d\t\t%d\t\t%s\n", i, p[i], alloc[i] != -1 ? (char[3]){alloc[i] + '0', '\0'} : "Not Allocated");

}

int main() {

int i, m, n, b[10], p[10];

printf("Enter no. of blocks: ");

scanf("%d", &m);

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

printf("Enter %d block size: ", i);

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

}

printf("Enter no. of processes: ");

scanf("%d", &n);

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

printf("Enter %d process size: ", i);

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

}

firstFit(b, m, p, n);

return 0;

}

Worst fit

#include <stdio.h>

void worstFit(int b[], int m, int p[], int n) {

int alloc[10], i, j, c;

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

alloc[i] = -1;

for (j = 0, c = -1; j < m; j++)

if (b[j] >= p[i] && (c == -1 || b[j] > b[c]))

c= j;

if (c != -1) {

alloc[i] = c;

b[c] -= p[i];

}

}

printf("Process No.\tProcess Size\tBlock No.\n");

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

printf("%d\t\t%d\t\t%s\n", i, p[i], alloc[i] != -1 ? (char[3]){alloc[i] + '0', '\0'} : "Not Allocated");

}

int main() {

int i, m, n, b[10], p[10];

printf("Enter no. of blocks: ");

scanf("%d", &m);

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

printf("Enter %d block size: ", i);

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

}

printf("Enter no. of processes: ");

scanf("%d", &n);

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

printf("Enter %d process size: ", i);

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

}

worstFit(b, m, p, n);

return 0;

}

Best fit

#include <stdio.h>

void bestFit(int b[], int m, int p[], int n) {

int alloc[10], i, j, bi;

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

alloc[i] = -1;

for (j = 0, bi = -1; j < m; j++)

if (b[j] >= p[i] && (bi == -1 || b[j] < b[bi]))

bi = j;

if (bi != -1) {

alloc[i] = bi;

b[bi] -= p[i];

}

}

printf("Process No.\tProcess Size\tBlock No.\n");

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

printf("%d\t\t%d\t\t%s\n", i, p[i], alloc[i] != -1 ? (char[3]){alloc[i] + '0', '\0'} : "Not Allocated");

}

int main() {

int i, m, n, b[10], p[10];

printf("Enter no. of blocks: ");

scanf("%d", &m);

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

printf("Enter %d block size: ", i);

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

}

printf("Enter no. of processes: ");

scanf("%d", &n);

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

printf("Enter %d process size: ", i);

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

}

bestFit(b, m, p, n);

return 0;

}

Fifo page replacement

#include <stdio.h>

int main() {

int n, a[50], b[10], m, i, k = 0, j, c = 0, hit;

printf("Enter number of pages: ");

scanf("%d", &n);

printf("Enter page numbers:\n");

for (i = 0; i < n; i++) scanf("%d", &a[i]);

printf("Enter number of frames: ");

scanf("%d", &m);

for (i = 0; i < m; i++) b[i] = -1;

printf("\nRef\tString\tPage\tFrames\n");

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

printf("%d\t\t\t", a[i]);

hit = 0;

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

if (b[j] == a[i]) hit = 1;

if (!hit) {

b[k] = a[i];

k = (k + 1) % m;

c++;

}

for (j = 0; j < m; j++) printf("%d\t", b[j]);

printf("\n");

}

printf("\nPage fault count is: %d\n", c);

return 0;

}

Bankers algorithm

#include <stdio.h>

int main() {

int total[3], available[3], work[3], max[5][3], alloc[5][3], need[5][3], safe[5];

char finish[5] = {'f', 'f', 'f', 'f', 'f'};

int i, j, k, count = 0;

printf("Enter instances of each resources:\n");

for (i = 0; i < 3; i++) {

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

}

printf("Enter maximum no of resources for each process:\n");

for (i = 0; i < 5; i++) {

for (j = 0; j < 3; j++) {

printf("Enter process %d resource %d: ", i, j + 1);

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

}

}

printf("Enter no of resources allocated to each process:\n");

for (i = 0; i < 5; i++) {

for (j = 0; j < 3; j++) {

printf("Enter the resource %d allocated to process %d: ", j + 1, i);

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

}

}

for (j = 0; j < 3; j++) {

int sum = 0;

for (i = 0; i < 5; i++) {

need[i][j] = max[i][j] - alloc[i][j];

sum += alloc[i][j];

}

available[j] = total[j] - sum;

work[j] = available[j];

}

printf("Allocated resources:\n");

for (i = 0; i < 5; i++) {

for (j = 0; j < 3; j++) {

printf("%d ", alloc[i][j]);

}

printf("\n");

}

printf("Maximum resources:\n");

for (i = 0; i < 5; i++) {

for (j = 0; j < 3; j++) {

printf("%d ", max[i][j]);

}

printf("\n");

}

printf("Available resources: ");

for (i = 0; i < 3; i++) {

printf("%d ", available[i]);

}

printf("\n");

while (count < 5) {

int found = 0;

for (i = 0; i < 5; i++) {

if (finish[i] == 'f') {

for (j = 0; j < 3; j++) {

if (need[i][j] > work[j]) {

break;

}

}

if (j == 3) {

for (k = 0; k < 3; k++) {

work[k] += alloc[i][k];

}

finish[i] = 't';

safe[count++] = i;

found = 1;

}

}

}

if (!found) {

break;

}

}

if (count == 5) {

printf("Safe sequence is:\n");

for (i = 0; i < 5; i++) {

printf("P%d ", safe[i]);

}

} else {

printf("System is not in a safe state.\n");

}

return 0;

}

Producer consumer problem

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define MAX 5

int buffer[MAX];

int in = 0, out = 0;

sem\_t empty, full;

pthread\_mutex\_t mutex;

void\* producer() {

for (int i = 0; i < 10; i++) {

int item = rand() % 100;

sem\_wait(&empty);

pthread\_mutex\_lock(&mutex);

buffer[in] = item;

printf("Produced: %d at index %d\n", item, in);

in = (in + 1) % MAX;

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

sleep(1);

}

return NULL;

}

void\* consumer() {

for (int i = 0; i < 10; i++) {

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

int item = buffer[out];

printf("Consumed: %d from index %d\n", item, out);

out = (out + 1) % MAX;

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty);

sleep(1);

}

return NULL;

}

int main() {

pthread\_t p, c;

sem\_init(&empty, 0, MAX);

sem\_init(&full, 0, 0);

pthread\_mutex\_init(&mutex, NULL);

pthread\_create(&p, NULL, producer, NULL);

pthread\_create(&c, NULL, consumer, NULL);

pthread\_join(p, NULL);

pthread\_join(c, NULL);

sem\_destroy(&empty);

sem\_destroy(&full);

pthread\_mutex\_destroy(&mutex);

return 0;

}