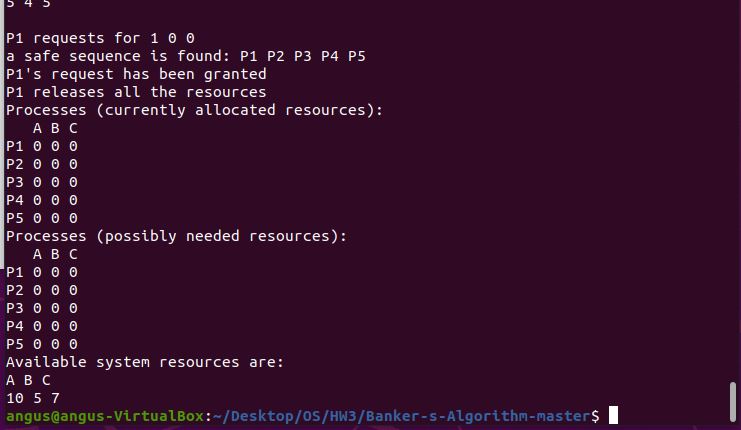
Steps：

1. gcc –pthread main.c
2. ./a.out 10 5 7



Source code：

/\*

team memeber：

0. printState輸出介面設計：戴陽

1. request and release resource 函式實作：江宥潔

2. Banker's algorithm 實作：易頡

3. thread programming support：辛愷庭

\*/

#include <stdio.h>

#include<stdlib.h>

#include <pthread.h>

#define NUMBER\_OF\_CUSTOMERS 5

#define NUMBER\_OF\_RESOURCES 3

int total[NUMBER\_OF\_RESOURCES]={0};

int available[NUMBER\_OF\_RESOURCES];

int maximum[NUMBER\_OF\_CUSTOMERS][NUMBER\_OF\_RESOURCES];

int allocation[NUMBER\_OF\_CUSTOMERS][NUMBER\_OF\_RESOURCES]={0};

int need[NUMBER\_OF\_CUSTOMERS][NUMBER\_OF\_RESOURCES];

int Bavailable[NUMBER\_OF\_RESOURCES];

int Bmaximum[NUMBER\_OF\_CUSTOMERS][NUMBER\_OF\_RESOURCES];

int Ballocation[NUMBER\_OF\_CUSTOMERS][NUMBER\_OF\_RESOURCES];

int Bneed[NUMBER\_OF\_CUSTOMERS][NUMBER\_OF\_RESOURCES];

int Finish[NUMBER\_OF\_CUSTOMERS]={0};

int release\_resources(int customer\_num);

int request\_resources(int customer\_num, int request[]);

void \*thread\_func(void\* customer\_numt);

int bankerAlgorithm(int customer\_num,int request[]);

void printState();

pthread\_mutex\_t mutex;

char string[NUMBER\_OF\_RESOURCES\*2]={0};

int safeSequence[NUMBER\_OF\_CUSTOMERS]={0};

int main(int argc, const char \* argv[]) {

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* initialize the matrices \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

available[i]=atoi(argv[i+1]);

total[i]=available[i]; // in the begining available resources equal to total resources

}

for(int i=0;i< NUMBER\_OF\_RESOURCES;i++){

for(int j=0;j<NUMBER\_OF\_CUSTOMERS;j++){

maximum[j][i]=rand()%(total[i]+1); //maximum should less than total

need[j][i]=maximum[j][i]; // need=maximum-allocation (allocation=0)

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* initialize the matrices \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* print the state of the process \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for (int i=0; i<NUMBER\_OF\_RESOURCES; i++) {

string[i\*2]=i+'A';

string[i\*2+1]=' ';

}

printf("Total system resources are:\n");

printf("%s\n",string);

for (int i=0; i<NUMBER\_OF\_RESOURCES; i++) {

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

}

printf("\n\nProcesses (maximum resources):\n");

printf(" %s\n",string);

for(int i=0; i<NUMBER\_OF\_CUSTOMERS;i++){

printf("P%d ",i+1);

for(int j=0;j<NUMBER\_OF\_RESOURCES;j++){

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

}

printf("\n");

}

printState();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* print the state of the process \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

pthread\_t p1,p2,p3,p4,p5;

int a=0,b=1,c=2,d=3,e=4;

pthread\_create(&p1,NULL,thread\_func,&a);

pthread\_create(&p2,NULL,thread\_func,&b);

pthread\_create(&p3,NULL,thread\_func,&c);

pthread\_create(&p4,NULL,thread\_func,&d);

pthread\_create(&p5,NULL,thread\_func,&e); //create 5 threads

char \*returnV;

pthread\_join(p1,(void\*\*)&returnV);

pthread\_join(p2,(void\*\*)&returnV);

pthread\_join(p3,(void\*\*)&returnV);

pthread\_join(p4,(void\*\*)&returnV);

pthread\_join(p5,(void\*\*)&returnV); // wait for all the 5 threads to terminate

return 0;

}

void \*thread\_func(void\* Tcustomer\_num){

int \*c=(int\*)Tcustomer\_num;

int customer\_num= \*c;

int requestSum=0;

while(!Finish[customer\_num]){ // the whilie loop stops when the thread has finished and its need becomes zero

requestSum=0;

int request[NUMBER\_OF\_RESOURCES]={0};

for(int i=0;i<NUMBER\_OF\_RESOURCES;i++){

request[i]=rand()%(need[customer\_num][i]+1); // generate a request below its need randomly

requestSum=requestSum+request[i];

}

if(requestSum!=0) // to make sure it doesn't requst for 0 reaources

while(request\_resources(customer\_num,request)==-1); // only when the request has been granted succesfully

// will the loop terminates, otherwise it will keep

// making the same request.

}

return 0;

}

int request\_resources(int customer\_num, int request[]){

int returnValue=-1;

// since i want the process to print the result continuously and also run without race condition, i make the part

// involing printing and accesing globale variable critical.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* critical section\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

pthread\_mutex\_lock(&mutex);

printf("\nP%d requests for ",customer\_num+1);

for(int i=0;i<NUMBER\_OF\_RESOURCES;i++){

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

}

printf("\n");

for(int i=0;i<NUMBER\_OF\_RESOURCES;i++){ // to check whether request<= availabe

// if it is not, then it will return -1

if(request[i]>available[i]){

printf("P%d is waiting for the reaources...\n",customer\_num+1);

pthread\_mutex\_unlock(&mutex); // before the thread return, it has to unlock the mutex

return -1;

}

}

returnValue=bankerAlgorithm(customer\_num,request); // execute banker's algorithm

if(returnValue==0){

int needIsZero=1;

printf("a safe sequence is found: ");

for(int i=0;i<NUMBER\_OF\_CUSTOMERS;i++){

printf("P%d ",safeSequence[i]+1 );

}

printf("\nP%d's request has been granted\n",customer\_num+1);

for(int j=0;j<NUMBER\_OF\_RESOURCES;j++){ // give the resources to the theread

allocation[customer\_num][j]=allocation[customer\_num][j]+request[j];

available[j]=available[j]-request[j];

need[customer\_num][j]=need[customer\_num][j]-request[j];

if(need[customer\_num][j]!=0){// to check if need is zero

needIsZero=0;

}

}

if(needIsZero){

Finish[customer\_num]=1; // if need is zero, mark the thread "finish"

release\_resources(customer\_num); // release resources when a thread finishes

}

printState();

}

else{

printf("cannot find a safe sequence\n");

}

pthread\_mutex\_unlock(&mutex); // unlock the mutex

return returnValue;

}

int release\_resources(int customer\_num){

printf("P%d releases all the resources\n",customer\_num+1);

for(int j=0;j<NUMBER\_OF\_RESOURCES;j++){

available[j]=available[j]+allocation[customer\_num][j]; // release the resources

allocation[customer\_num][j]=0;

}

return 0;

}

int bankerAlgorithm(int customer\_num,int request[]){

int finish[NUMBER\_OF\_CUSTOMERS]={0};

for(int i=0;i<NUMBER\_OF\_RESOURCES;i++){ // copy the matrices

Bavailable[i]=available[i];

for(int j=0;j<NUMBER\_OF\_CUSTOMERS;j++){

Ballocation[j][i]=allocation[j][i];

Bmaximum[j][i]=maximum[j][i];

Bneed[j][i]=need[j][i];

}

}

for(int i=0;i<NUMBER\_OF\_RESOURCES;i++){ // pretend to give the resource to the thread

Bavailable[i]=Bavailable[i]-request[i];

Ballocation[customer\_num][i]=Ballocation[customer\_num][i]+request[i];

Bneed[customer\_num][i]=Bneed[customer\_num][i]-request[i];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*safety Algorithm\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int count=0;

while(1){

int I=-1;

for(int i=0;i<NUMBER\_OF\_CUSTOMERS;i++){ // to find a thread that its need is less than or equal to available.

int nLessThanA=1;

for(int j=0; j<NUMBER\_OF\_RESOURCES;j++){

if(Bneed[i][j]>Bavailable[j] || finish[i]==1){

nLessThanA=0;

break;

}

}

if(nLessThanA){ // if the thread is found, record its thread number

I=i;

break;

}

}

if(I!=-1){

safeSequence[count]=I; // record the sequence

count++;

finish[I]=1; // mark the thread "finish"

for(int k=0;k<NUMBER\_OF\_RESOURCES;k++){ // pretend to give the reaource to thread

Bavailable[k]=Bavailable[k]+Ballocation[I][k];

}

}

else{ // if can not find any thread that its need is less than or equal to available.

for(int i=0;i<NUMBER\_OF\_CUSTOMERS;i++){

if(finish[i]==0){ // if there is any thread hasn't been found, that means it can't find a safe sequence

return -1;

}

}

return 0; // all the threads have been found

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*safety Algorithm\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

void printState(){

printf("Processes (currently allocated resources):\n");

printf(" %s\n",string);

for(int i=0; i<NUMBER\_OF\_CUSTOMERS;i++){

printf("P%d ",i+1);

for(int j=0;j<NUMBER\_OF\_RESOURCES;j++){

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

}

printf("\n");

}

printf("Processes (possibly needed resources):\n");

printf(" %s\n",string);

for(int i=0; i<NUMBER\_OF\_CUSTOMERS;i++){

printf("P%d ",i+1);

for(int j=0;j<NUMBER\_OF\_RESOURCES;j++){

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

}

printf("\n");

}

printf("Available system resources are:\n");

printf("%s\n",string);

for (int i=0; i<NUMBER\_OF\_RESOURCES; i++) {

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

}

printf("\n");

}