LOVELYPROFESSIONALUNIVERSITY

CA-2

CourseTitle:OperatingSystem

Term:Max.Marks:30 DateofSubmission:2 April,2020

By Ch Nikhil Chowdary

RegistrationNo:11710918

RollNo:B-42

Section:EE028

Submittedto:Manjit mam



# SchoolofElecronicsCommunicationandEngineering

LovelyProfessionalUniversity Phagwara,Punjab StudentName:Ch Nikhil Chowdary

GitHubRepository:https://github.com/NikhilChowdary49/nikhil.git

Banker'salgorithm:

TheBanker'salgorithmisaresourceallocation&deadlockavoidancealgorithm developedbyEdsgerDijkstrathattestsforsafetybysimulatingtheallocationofpredeterminedmaximumpossibleamountsofallresources,andthenmakesa"safe-state" checktotestforpossibledeadlockconditionsforallotherpendingactivities,before decidingwhetherallocationshouldbeallowedtocontinue.Thealgorithmwas developedinthedesignprocessfortheoperatingsystemandoriginallydescribed(in

Dutch)inEWD108.Thenameisbyanalogywiththewaythatbankersaccountfor liquidityconstraints.

AlgorithmTheBanker'salgorithmisrunbytheoperatingsystemwheneveraprocess requestsresources.Thealgorithmpreventsdeadlockbydenyingorpostponingthe requestifitdeterminesthatacceptingtherequestcouldputthesysteminanunsafe state(onewheredeadlockcouldoccur).Whenanewprocessentersasystem,itmust declarethemaximumnumberofinstancesofeachresourcetypethatmaynotexceed thetotalnumberofresourcesinthesystem.

Letusassumethattherearenprocessesandmresourcetypes.Somedatastructuresthat areusedtoimplementthebanker'salgorithm

SafeandUnsafeStates

Astateisconsideredsafeifitispossibleforallprocessestofinishexecuting (terminate).Sincethesystemcannotknowwhenaprocesswillterminate,orhowmany resourcesitwillhaverequestedbythen,thesystemassumesthatallprocesseswill eventuallyattempttoacquiretheirstatedmaximumresourcesandterminatesoon afterward.Thisisareasonableassumptioninmostcasessincethesystemisnot particularlyconcernedwithhowlongeachprocessruns(atleastnotfromadeadlock avoidanceperspective).Also,ifaprocessterminateswithoutacquiringitsmaximum resources,itonlymakesiteasieronthesystem.Giventhatassumption,thealgorithm determinesifastateissafebytryingtofindahypotheticalsetofrequestsbythe processesthatwouldalloweachtoacquireitsmaximumresourcesandthenterminate (returningitsresourcestothesystem).Anystatewherenosuchsetexistsisanunsafe state.

Banker’sAlgorithm:

Definequantities:

* AVAILABLE:array[1..m]ofinteger;--itspecifiesforeachresourcehowmany copiesofitareavailable
* ALLOCATION:array[1..n,1..m]ofinteger;--ALLOCATION[i,j]specifiesthe numberofcopiesofresourcejthatareallocatedtoprocessi.
* MAXIM:array[1..n,1..m]ofinteger;--MAXIM[i,j]specifiesthemaximumnumber ofcopiesofresourcejthatprocessiwilluse.
* NEED;array[1..n,1..m]ofinteger;--NEED[i,j]specifiesthenumberofcopiesof resourcejthatprocessistillrequires.ItisequaltoMAXIM[i,j]-ALLOCATION[i,j]

andthefollowingnotation

A<B,whereAandBarem-aryvectors,istrueiffforalli,A[i]<B[i] IfAisarectangularmatrix,Aiisitsithrow.

procedureBANKER(REQUEST\_I:array[1..m]ofinteger; i:1..n)is

{

ifREQUEST\_I>NEEDithen

ERROR; --Theuserisaskingmorethantheagreedmaximum repeat while(REQUEST\_i>AVAILABLE) yield;--Resourcesarenotavailableatthistime

ALLOCATION\_i=ALLOCATION\_i+ REQUEST\_i; AVAILABLE=AVAILABLE-REQUEST\_I; ifSAFE\_STATEthen

RETURN;--Therequestisapproved

ALLOCATION\_i=ALLOCATION\_I- REQUEST\_i;

AVAILABLE=AVAILABLE+REQUEST\_i;

YIELD; --Therequestcannotsafelybesatisfiedatthistime forever;

}

BOOLEANfunctionSAFESTATEis--Determinesifcurrentstateissafe

{NOCHANGE:boolean;

WORK:array[1..m]ofINTEGER=AVAILABLE;

FINISH:array[1..n]ofboolean=[false,..,false]; I:integer;

repeat

NOCHANGE=TRUE; forI=1toNdo

if((notFINISH[i])and

NEEDi<=WORK)then{

WORK=WORK+ALLOCATION\_i;

FINISH[i]=true;

NOCHANGE=false;

}

untilNOCHANGE;

return(FINISH==(true,..,true));

}

ThetimecomplexityoftheBanker'salgorithmasafunctionofthenumbernof processesandmofresourcesiso(n\*n\*m).

Banker’s(Safety)Algorithm:

findasafesequence,

i.e.isthesysteminasafestate?

1.LetWorkandFinishbevectorslengthmandnrespectively. InitializeWork=Available,andFinish[i]=falsefori=0,...,n

-1

2.Findaprocessisuchthatboth

•Finish[i]==false,and

•Needi≤Work

3.Work=Work+

AllociFinish[i]= trueGotostep

2.

4.IfFinish[i]==trueforalli,thenthesystemisinasafestate

GivenProblem:(Code)

Reena’soperatingsystemusesanalgorithmfordeadlockavoidancetomanagethe allocationofresourcessaythreenamelyA,B,andCtothreeprocessesP0,P1,andP2. Considerthefollowingscenarioasreference.usermustenterthecurrentstateof systemasgiveninthisexample:SupposeP0has0,0,1instances,P1ishaving3,2,0 instancesandP2occupies2,1,1instancesofA,B,Cresourcerespectively.Alsothe maximumnumberofinstancesrequiredforP0is8,4,3andforp1is6,2,0andfinallyfor P2thereare3,3,3instancesofresourcesA,B,Crespectively.Thereare3instancesof resourceA,2instancesofresourceBand2instancesofresourceCavailable.

WriteaprogramtocheckwhetherReena’soperatingsystemisinasafestateor notinthefollowingindependentrequestsforadditionalresourcesinthecurrent state:

1.Request1:P0requests0instancesofAand0instancesofBand2instancesofC.

2.Request2:P1requestsfor2instancesofA,0instancesofBand0instancesofC.

ProposedSolution:

#include<stdio.h>

#include<stdlib.h> #include<unistd.h>

voidprint(intx[][10],intn,intm){ inti,j; for(i=0;i<n;i++){ printf("\n"); for(j=0;j<m;j++){ printf("%d\t",x[i][j]);

}

}

}

//ResourceRequestalgorithm

voidres\_request(intA[10][10],intN[10][10],intAV[10][10],intpid,intm)

{

intreqmat[1][10]; inti;

printf("\nEnteradditionalrequest:-\n"); for(i=0;i<m;i++){ printf("Requestforresource%d:",i+1);

scanf("%d",&reqmat[0][i]);

}

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

if(reqmat[0][i]>N[pid][i]){

printf("\nErrorencountered.\n"); exit(0);

}

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

if(reqmat[0][i]>AV[0][i]){

printf("\nResourcesunavailable.\n"); exit(0);

}

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

AV[0][i]-=reqmat[0][i];

A[pid][i]+=reqmat[0][i];

N[pid][i]-=reqmat[0][i];

}

}

//Safetyalgorithm

intsafety(intA[][10],intN[][10],intAV[1][10],intn,intm,inta[]){

inti,j,k,x=0; intF[10],W[1][10]; intpflag=0,flag=0; for(i=0;i<n;i++)

F[i]=0; for(i=0;i<m;i++)

W[0][i]=AV[0][i];

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

if(F[i]==0){

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

if(N[i][j]>W[0][j]) flag=1;

}

if(flag==0&&F[i]==0){

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

W[0][j]+=A[i][j]; F[i]=1; pflag++; a[x++]=i;

}

}

}

if(pflag==n)

return1;

}

return0;

}

//Banker'sAlgorithm

voidaccept(intA[][10],intN[][10],intM[10][10],intW[1][10],int\*n,int

\*m){inti,j;

printf("\nEntertotalno.ofprocesses:"); scanf("%d",n);

printf("\nEntertotalno.ofresources:"); scanf("%d",m); for(i=0;i<\*n;i++){ printf("\nProcess%d\n",i+1); for(j=0;j<\*m;j++){

printf("Allocationforresource%d:",j+1); scanf("%d",&A[i][j]);

printf("Maximumforresource%d:",j+1);

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

}

}

printf("\nAvailableresources:\n"); for(i=0;i<\*m;i++){

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

scanf("%d",&W[0][i]);

}

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

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

N[i][j]=M[i][j]-A[i][j];

printf("\nAllocationMatrix"); print(A,\*n,\*m);

printf("\nMaximumRequirementMatrix"); print(M,\*n,\*m); printf("\nNeedMatrix"); print(N,\*n,\*m);

}

intbanker(intA[][10],intN[][10],intW[1][10],intn,int

m){intj,i,a[10]; j=safety(A,N,W,n,m,a); if(j!=0){

printf("\n\n"); for(i=0;i<n;i++)

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

printf("\nAsafetysequencehasbeen detected.\n");return1;

}else{ printf("\nDeadlockhasoccured.\n"); return0;

}

}

intmain(){

intret; intA[10][10]; intM[10][10]; intN[10][10]; intW[1][10];

intn,m,pid,ch;

printf("\nDEADLOCKAVOIDANCEUSINGBANKER'SALGORITHM\n"); accept(A,N,M,W,&n,&m); ret=banker(A,N,W,n,m); if(ret!=0){

printf("\nDoyouwantmakeanadditionalrequest?(1=Yes|0=No)"); scanf("%d",&ch); if(ch==1){

printf("\nEnterprocessno.:"); scanf("%d",&pid); res\_request(A,N,W,pid-1,m); ret=banker(A,N,W,n,m); if(ret==0)

exit(0);

}

}else exit(0);

return0;

}

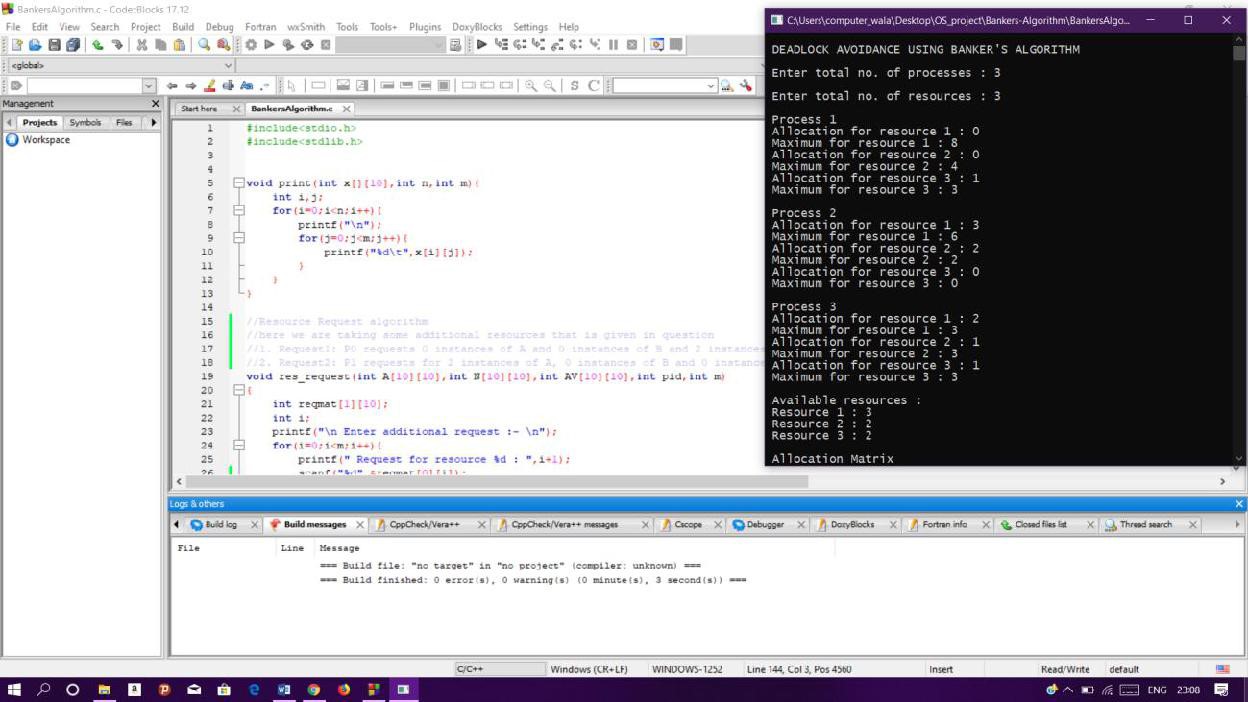
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process | Allocatio  n | |  | Max | |  | | Available |
|  | A B | | C | A B | | C | | A B C |
| P0 | 0 0 | | 1 | 8 4 | | 3 | | 3 2 2 |
| P1 | 3 2 | | 0 | 6 2 | | 0 | |  |
| P2 | 2 1 | | 1 | 3 3 | | 3 | |  |
| Need | | |  | |
| A B | | | C | |
| 8 4 | | | 2 | |
| 3 0 | | | 1 | |
| 1 2 | | | 2 | |

 

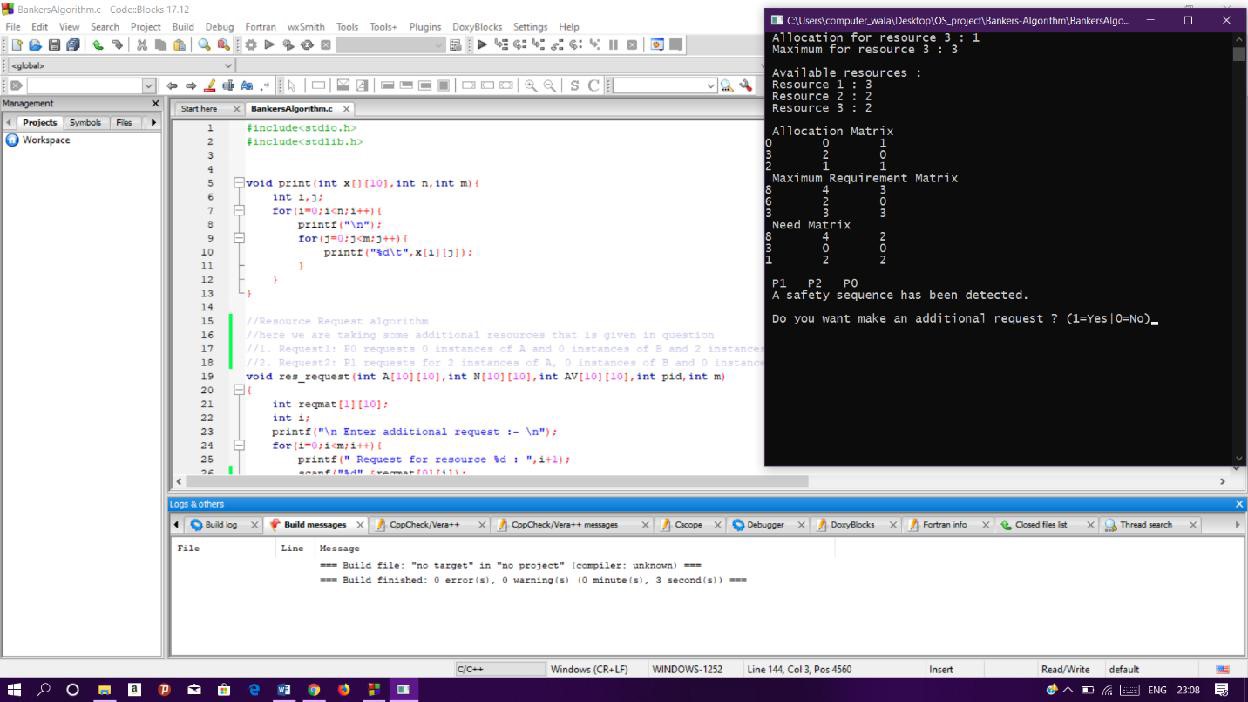
P1----------P2---------- P0

Theprocesswillbeinsafestate

OutputofProgram:Part1askusertoentervaluesofprocessandresource



Part2:Itwillassignvaluesintomatrix,findneedmatrixandusingsafetyalgorithmit willmakesequenceoftheprocessandprintweatherprocessisinsafestateornot



Part3:afterfinishingallprocessitwillaskforadditionalresourceandaskuserto inputthemaccordingtothatitwilldosequencingofprocessandprinttheoutput

