*Run with command : On linux shell.*

*🡺gcc pthread –o Banker BankersAlgorithm.c*

*🡺./Banker*

*#include*<pthread.h>

*#include*<sys/types.h>

*#include*<unistd.h>

*#include*<stdio.h>

*#include*<stdlib.h>

*/\* Varibles \*/*

int processV1=0,

    n,

    x=0,

    y=0,

    n2,

    np,

    na,

    r=0,

    ctr=0,*// Counter*

    cnt=0, *// Count*

    processRG=0,

    processRGG=0,

    flag=1,

    tr,

    Ucnt=0,

    Uflag=0;

int process[10][10][10],

    processAvail[10],

    processAvail2[10],

    processState[10],

    processFormat[1][10][10],

    processRR[1][0][10];

pthread\_mutex\_t lock;

int processCheck();

int processCheck1();

void processUI();

void process1();

void process2();

void process3();

void process4();

void processRReq();

void processMRReq();

void processTHF();

void processTHI();

void processTHUI();

void processDef();

void processDone();

void processBOTTOM();

void processULTIMATE();

void processTotalRecall();

void \*processTH1(); *// To ensure safe access to shared data, mutex locks are implemented  through these functions*

void \*processTH2(); *// To ensure safe access to shared data, mutex locks are implemented  through these functions*

void \*processTH3(); *// To ensure safe access to shared data, mutex locks are implemented  through these functions*

*/\*=================================\*/*

*/\*=================================\*/*

int main()

{

    system("clear");

    printf("\nBanker's  Algorithm\n");

    printf("CSE-316 Project\n");

    printf("PRESS (1) to CONTINUE\n");

    int n;

    printf("Enter your desired choice : ");

    scanf("%d",&n);

*switch*(n)

    {

*case* 1:

            processUI();

*break*;

*default*:

            printf("\nplease PRESS valid option\n");

    }

}

*/\*=================================\*/*

*/\*=================================\*/*

void processUI()

{

    system("clear");

    printf("\nBanker's  Algorithm\n");

    printf("CSE-316 Project\n");

    printf("---------------------------------------------\n");

    printf("\nPRESS (1) to OPTIONS for THREADS\n");

    printf("PRESS (2) to OPTIONS for PROCESS\n");

    printf("PRESS (3) to OPTIONS for DEFAULT\n");

    printf("\n--------------------------------------------\n");

    printf("Enter your desired choice:");

    scanf("%d",&tr);

*switch*(tr)

    {

*case* 1:

            processTHI();

*break*;

*case* 2:

            process1();

*break*;

*case* 3:

            processDef();

*break*;

*default*:

            printf("\nplease PRESS valid option\n");

    }

}

void process1()

{

    ctr=0;

    cnt=0;

    r=0;

    system("clear");

    printf("\nBanker's  Algorithm\n");

    printf("CSE-316 Project\n");

    printf("---------------------------------------\n");

    printf("\nEnter number of RESOURSES types\n");

    printf("Enter your desired choice:");

    scanf("%d",&np);

    printf("\nEnter number of PROCESSES\n");

    printf("Enter your desired choice:");

    scanf("%d",&n);

    printf("-----------------------------------------\n");

    printf("for each PROCESS (P1,P2,...)\n");

    printf("-----------------------------------------\n");

    printf("Enter MAXIMUM resources required by :\n");

*for*(int i=0;i<n;i++)

    {

        printf("PROCESS %d :",(++ctr));

        printf("\n");

*for*(int j=0;j<np;j++)

        {

            printf("R%d :",(++cnt));

            scanf("%d",&n2);

            process[0][r][j]=n2;

        }

        r++;

        cnt=0;

        printf("\n");

    }

    r=0;

    ctr=0;

    printf("Enter ALLOCATED no of resources to :\n");

*for*(int i=0;i<n;i++)

    {

        printf("PROCESS %d :",(++ctr));

        printf("\n");

*for*(int j=0;j<np;j++)

        {

            printf("R%d :",(++cnt));

            scanf("%d",&n2);

            process[1][r][j]=n2;

        }

        r++;

        cnt=0;

        printf("\n");

    }

    printf("Enter AVAILABLE no of resources\n");

*for*(int i=0;i<np;i++)

    {

        printf("R%d : ",(++cnt));

        scanf("%d",&na);

        processAvail[i]=na;

    }

    printf("-----------------------------------------\n");

    printf("PRESS (1) to check SAFE-STATE\n");

    printf("PRESS (2) to grant RESOURCES\n");

    printf("PRESS (3) to request RESOURCES\n");

    printf("\nEnter The  Number : ");

    int m;

    scanf("%d",&m);

*switch*(m)

    {

*case* 1:

            process2();

*break*;

*default*:

            printf("Please enter VALID option\n");

    }

}

*/\*=================================\*/*

*/\*=================================\*/*

void process2()

{

    r=0;

*for*(int j=0;j<n;j++)

    {

*for*(int i=0;i<np;i++)

        {

            process[2][r][i]=(process[0][r][i]-process[1][r][i]);

            processFormat[0][r][i]=process[2][r][i];

        }

        r++;

    }

    process3();

}

*/\*=================================\*/*

*/\*=================================\*/*

int processCheck1()

{

    r=0;

    int zz=0;

*for*(int j=0;j<n;j++)

    {

*for*(int i=0;i<np;i++)

        {

*if*((process[1][r][i]>=0)&&(process[0][r][i]>=0))

*if*(process[1][r][i]<=process[0][r][i])

                zz=1;

*else*

                zz=0;

        }

        r++;

    }

*if*(zz==1)

    {

        Uflag=1;

*return* 0;

    }

*else*

    {

        Uflag=0;

*return* 1;

    }

}

void process3()

{

    int fl=0;

*for*(int z=0;z<n;z++)

    {

        r=0;

*for*(int j=0;j<n;j++)

        {

*for*(int i=0;i<np;i++)

            {

*if*((process[2][r][i]>=0)&&(process[2][r][i]<=processAvail[i]))

                {x=1;++y;}

*else*

                    {x=0;}

            }

*if*(x==1 && y==np)

            {

                processState[fl]=(r+1);

                fl++;

*for*(int k=0;k<np;k++)

                {

                    processAvail[k]+=process[1][r][k];

                    process[2][r][k]=100;

                }

            }

            y=0;

            r++;

        }

    }

*for*(int i=0;i<n;i++)

    {

*if*(processCheck1() || processState[i]==0)

        {

            flag=0;

        }

*else*

            processRG=1;

    }

    processTotalRecall();

}

*/\*=================================\*/*

*/\*=================================\*/*

void processTotalRecall()

{

*if*(tr==1)

    {

        Ucnt++;

        processTHUI();

    }

*if*(tr==2)

        process4();

}

*/\*=================================\*/*

*/\*=================================\*/*

void process4()

{

    r=0;

    system("clear");

    printf("\nBanker's  Algorithm\n");

    printf("CSE-316 Project\n");

    printf("-----------------------------------------\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("PROCESS MAX  ALLOCATED NEED\n");

*for*(int i=0;i<n;i++)

    {

        printf("  P%d",(i+1));

*for*(int j=0;j<np;j++)

        {

            printf(" %d",process[0][r][j]);

        }

*for*(int j=0;j<np;j++)

        {

            printf(" %d",process[1][r][j]);

        }

*for*(int j=0;j<np;j++)

        {

            printf(" %d",processFormat[0][r][j]);

        }

        printf("\n");

        r++;

    }

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

*if*(flag==1)

    {

        printf("SAFE-STATE is : < ");

*for*(int i=0;i<n;i++)

        {

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

        }

        printf(">\n");

    }

*if*(flag==0)

    {

        printf("SAFE-STATE does't EXIST \n");

    }

    printf("-----------------------------------------\n");

    printf("PRESS (1) to go on MAIN-MENU\n");

    printf("PRESS (2) to EXIT\n");

    int non;

    printf("Enter your desired choice : ");

    scanf("%d",&non);

*switch*(non)

    {

*case* 1:

            processUI();

*break*;

*case* 2:

            exit(0);

*default*:

            printf("\nplease PRESS valid option\n");

    }

}

*/\*=================================\*/*

*/\*=================================\*/*

void processRReq()

{

    r=0;

    ctr=0;

    cnt=0;

    printf("Enter RESOURCE REQUEST for:\n");

*for*(int i=0;i<n;i++)

    {

        printf("THREADS %d :",(++ctr));

        printf("\n");

*for*(int j=0;j<np;j++)

        {

            printf("R%d :",(++cnt));

            scanf("%d",&n2);

            processRR[0][r][j]=n2;

        }

        r++;

        cnt=0;

        printf("\n");

    }

    pthread\_mutex\_init(&lock,NULL);

    pthread\_t a,b,c;

*if*(n==1)

    {

        pthread\_create(&a,NULL,processTH1,NULL);

        pthread\_join(a,NULL);

    }

*if*(n==2)

    {

        pthread\_create(&a,NULL,processTH1,NULL);

        pthread\_create(&b,NULL,processTH2,NULL);

        pthread\_join(a,NULL);

        pthread\_join(b,NULL);

    }

*if*(n==3)

    {

        pthread\_create(&a,NULL,processTH1,NULL);

        pthread\_create(&b,NULL,processTH2,NULL);

        pthread\_create(&c,NULL,processTH3,NULL);

        pthread\_join(a,NULL);

        pthread\_join(b,NULL);

        pthread\_join(c,NULL);

    }

}

*/\*=================================\*/*

*/\*=================================\*/*

int processCheck(int wow)

{

    int zz=0;

*for*(int i=0;i<np;i++)

    {

*if*(processRR[0][wow][i]<=process[0][wow][i])

            zz=1;

*else*

            zz=0;

    }

*if*(zz==1)

*return* 1;

*else*

*return* 0;

}

*/\*=================================\*/*

*/\*=================================\*/*

void processMRReq(int wow)

{

    processRGG=1;

*for*(int j=0;j<np;j++)

    {

        process[1][wow][j]+=processRR[0][wow][j];

    }

*for*(int j=0;j<np;j++)

    {

        processAvail[j]=processAvail2[j];

    }

*for*(int j=0;j<np;j++)

    {

        processAvail[j]-=processRR[0][wow][j];

    }

*if*(processCheck(wow))

    {

        Uflag=1;

        process2();

    }

*else*

        processTHF();

}

*/\*=================================\*/*

*/\*=================================\*/*

void \*processTH1()

{

    pthread\_mutex\_lock(&lock);

    processMRReq(0);

    pthread\_mutex\_unlock(&lock);

}

void \*processTH2()

{

    pthread\_mutex\_lock(&lock);

    processMRReq(1);

    pthread\_mutex\_unlock(&lock);

}

void \*processTH3()

{

    pthread\_mutex\_lock(&lock);

    processMRReq(2);

    pthread\_mutex\_unlock(&lock);

}

*/\*=================================\*/*

*/\*=================================\*/*

void processTHI()

{

    ctr=0;

    r=0;

    cnt=0;

    system("clear");

    printf("\nBanker's  Algorithm\n");

    printf("CSE-316 Project\n");

    printf("-----------------------------------------\n");

    printf("\nEnter number of RESOURSES types\n");

    printf("Enter your desired choice : ");

    scanf("%d",&np);

    printf("\nEnter number of THREADS to create (MAX : 3)\n");

    printf("Enter your desired choice : ");

    scanf("%d",&n);

*if*(n<=3)

    {

        printf("-----------------------------------------\n");

        printf("for each THREADS (TH1,TH2,...)\n");

        printf("-----------------------------------------\n");

        printf("Enter MAXIMUM resources required by :\n");

*for*(int i=0;i<n;i++)

        {

            printf("THREADS %d :",(++ctr));

            printf("\n");

*for*(int j=0;j<np;j++)

            {

                printf("R%d :",(++cnt));

                scanf("%d",&n2);

                process[0][r][j]=n2;

            }

            r++;

            cnt=0;

            printf("\n");

        }

        r=0;

        ctr=0;

        printf("Enter ALLOCATED no of resources to :\n");

*for*(int i=0;i<n;i++)

        {

            printf("THREADS %d :",(++ctr));

            printf("\n");

*for*(int j=0;j<np;j++)

            {

                printf("R%d :",(++cnt));

                scanf("%d",&n2);

                process[1][r][j]=n2;

            }

            r++;

            cnt=0;

            printf("\n");

        }

        printf("Enter AVAILABLE no of resources\n");

*for*(int i=0;i<np;i++)

        {

            printf("R%d : ",(++cnt));

            scanf("%d",&na);

            processAvail[i]=na;

            processAvail2[i]=na;

        }

        printf("-----------------------------------------\n");

        printf("PRESS (1) to check SAFE-STATE\n");

        printf("PRESS (2) to request RESOURCES\n");

        printf("\nEnter The  Number : ");

        int m;

        scanf("%d",&m);

*switch*(m)

        {

*case* 1:

                process2();

*break*;

*default*:

                printf("Please enter VALID option\n");

        }

    }

*else*

        printf("!!! OVERFLOW !!! cannot create more than 3 THREADS\n");

}

*/\*=================================\*/*

*/\*=================================\*/*

void processDone()

{

*//if(Ucnt==1)*

    printf("\nBanker's  Algorithm\n");

    printf("CSE-316 Project\n");

    printf("-----------------------------------------\n");

}

void processTHUI()

{

    r=0;

    system("clear");

*if*(Ucnt==1)

        system("clear");

    processDone();

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("PROCESS MAX  ALLOCATED NEED\n");

*for*(int i=0;i<n;i++)

    {

        printf("  TH%d",(i+1));

*for*(int j=0;j<np;j++)

        {

            printf(" %d",process[0][r][j]);

        }

*for*(int j=0;j<np;j++)

        {

            printf(" %d",process[1][r][j]);

        }

*for*(int j=0;j<np;j++)

        {

            printf(" %d",processFormat[0][r][j]);

        }

        printf("\n");

        r++;

    }

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

*if*(flag==1)

    {

        printf("SAFE-STATE is : < ");

*for*(int i=0;i<n;i++)

        {

            printf("TH%d ",processState[i]);

        }

        printf(">\n");

    }

*if*(flag==0)

    {

        printf("SAFE STATE doesn't EXIST \n");

    }

    processULTIMATE();

*if*(processRGG!=1)

        processBOTTOM();

}

*/\*=================================\*/*

*/\*=================================\*/*

void processTHF()

{

    r=0;

    system("clear");

    printf("\nBanker's  Algorithm\n");

    printf("CSE-316 Project\n");

    printf("-----------------------------------------\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("PROCESS MAX  ALLOCATED NEED\n");

*for*(int i=0;i<n;i++)

    {

        printf("  TH%d",(i+1));

*for*(int j=0;j<np;j++)

        {

            printf(" %d",process[0][r][j]);

        }

*for*(int j=0;j<np;j++)

        {

            printf(" %d",process[1][r][j]);

        }

*for*(int j=0;j<np;j++)

        {

            printf(" %d",processFormat[0][r][j]);

        }

        printf("\n");

        r++;

    }

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("SAFE STATE doesn't EXIST \n");

*if*(processRGG!=1)

        processBOTTOM();

}

*/\*=================================\*/*

*/\*=================================\*/*

void processBOTTOM()

{

    printf("-----------------------------------------\n");

    printf("PRESS (1) to REQUEST RESOURCES\n");

    printf("PRESS (2) to EXIT\n");

    printf("Enter your desired choice : ");

    int m;

    scanf("%d",&m);

*switch*(m)

    {

*case* 1:

            processRReq();

*break*;

*case* 2:

            exit(0);

*default*:

            printf("Please enter VALID option\n");

    }

}

*/\*=================================\*/*

*/\*=================================\*/*

void processULTIMATE()

{

*if*(processRG==1 && processRGG==1 && Uflag==1)

        printf("REQUEST granted SUCCESSFULLY\n");

*if*(processRG==0 && processRGG==1)

        printf("REQUEST cannot be granted\n");

}

*/\*=================================\*/*

*/\*=================================\*/*

void processDef()

{

    n=3;

    np=3;

    process[0][0][0]=5;

    process[0][0][1]=5;

    process[0][0][2]=6;

    process[0][1][0]=6;

    process[0][1][1]=6;

    process[0][1][2]=2;

    process[0][2][0]=8;

    process[0][2][1]=8;

    process[0][2][2]=7;

    process[1][0][0]=1;

    process[1][0][1]=1;

    process[1][0][2]=4;

    process[1][1][0]=1;

    process[1][1][1]=1;

    process[1][1][2]=0;

    process[1][2][0]=7;

    process[1][2][1]=7;

    process[1][2][2]=6;

    processAvail[0]=1;

    processAvail[1]=1;

    processAvail[2]=1;

    tr=2;

    process2();

}