OS CLASS ASSIGNMENT

REVISION ON DEADLOCKS

NAME:KOLA LOKESH

REG.NO:19BCS056

1.  
code:

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include<signal.h>

#define n 5

int compltedPhilo = 0,i;

struct Spoon

{

int taken;

}SpoonAvil[n];

struct philosp

{

int left;

int right;

}Philostatus[n];

enum

{

THINKING,

HUNGRY,

EATING

}state[5];

void goForDinner(int philID)

{

if(Philostatus[philID].left==10 && Philostatus[philID].right==10)

{

signal(philID,NULL);

printf("Philosopher %d completed his dinner\n",philID+1);

}

else if(Philostatus[philID].left==1 && Philostatus[philID].right==1)

{

state[i]= EATING;

signal(philID,NULL);

printf("Philosopher %d completed his dinner\n",philID+1);

Philostatus[philID].left = Philostatus[philID].right = 10;

int otherSpoon = philID-1;

if(otherSpoon== -1)

{

otherSpoon=(n-1);

}

SpoonAvil[philID].taken = SpoonAvil[otherSpoon].taken = 0;

printf("Philosopher %d released Spoon %d and Spoon %d\n",philID+1,philID+1,otherSpoon+1);

compltedPhilo++;

}

else if(Philostatus[philID].left==1 && Philostatus[philID].right==0)

{

if(philID==(n-1))

{

if(SpoonAvil[philID].taken==0)

{

SpoonAvil[philID].taken = Philostatus[philID].right = 1;

printf("Spoon %d taken by philosopher %d\n",philID+1,philID+1);

}

else

{

wait(NULL);

printf("Philosopher %d is waiting for Spoon %d\n",philID+1,philID+1);

}

}

else

{

int dupphilID = philID;

philID-=1;

if(philID== -1)

philID=(n-1);

if(SpoonAvil[philID].taken == 0)

{

SpoonAvil[philID].taken = Philostatus[dupphilID].right = 1;

printf("Spoon %d taken by Philosopher %d\n",philID+1,dupphilID+1);

}

else

{

wait(NULL);

printf("Philosopher %d is waiting for Spoon %d\n",dupphilID+1,philID+1);

}

}

}

else if(Philostatus[philID].left==0)

{

if(philID==(n-1))

{

if(SpoonAvil[philID-1].taken==0)

{

SpoonAvil[philID-1].taken = Philostatus[philID].left = 1;

printf("Spoon %d taken by philosopher %d\n",philID,philID+1);

}

else

{

wait(NULL);

printf("Philosopher %d is waiting for Spoon %d\n",philID+1,philID);

}

}

else

{

if(SpoonAvil[philID].taken == 0)

{

SpoonAvil[philID].taken = Philostatus[philID].left = 1;

printf("Spoon %d taken by Philosopher %d\n",philID+1,philID+1);

}

else

{

wait(NULL);

printf("Philosopher %d is waiting for Spoon %d\n",philID+1,philID+1);

}

}

}

else{}

}

int main()

{

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

{

state[i]=THINKING;

}

printf("All Philosophers are THINKING\n");

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

{

state[i]=HUNGRY;

}

printf("All Philosophers are HUNGRY\n");

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

{

SpoonAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;

}

while(compltedPhilo<n)

{

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

{

goForDinner(i);

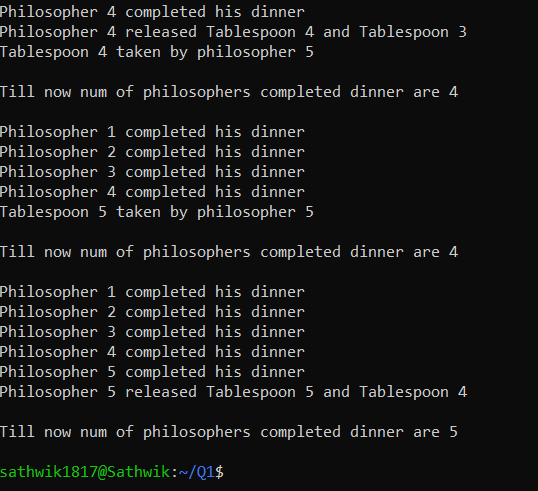
}

printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);

}

return 0

}



2.

Code:

#include <stdio.h>

int curr[5][5], maxclaim[5][5], avl[5];

int alloc[5] = {0, 0, 0, 0, 0};

int maxres[5], running[5], safe=0;

int count = 0, i, j, exec, r, p, k = 1;

int main()

{

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

scanf("%d", &p);

int safe\_sequence[p];

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

{

running[i] = 1;

count++;

}

printf("Enter the number of resources\n");

scanf("%d", &r);

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

{

printf("Enter the resources for instance %d : ", k++);

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

}

printf("Enter allocated resource table\n");

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

{

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

{

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

}

}

printf("Enter maximum resource table\n");

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

{

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

{

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

}

}

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

{

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

{

alloc[j] += curr[i][j];

}

}

printf("Allocated resources\n");

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

{

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

}

printf("\n");

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

{

avl[i] = maxres[i] - alloc[i];

}

printf("Available resources\n");

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

{

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

}

printf("\n");

while (count != 0)

{

safe = 0;

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

{

if (running[i])

{

exec = 1;

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

{

if (maxclaim[i][j] - curr[i][j] > avl[j])

{

exec = 0;

break;

}

}

if (exec)

{

printf("Process %d is executing\n", i + 1);

running[i] = 0;

count--;

safe = 1;

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

{

avl[j] += curr[i][j];

}

break;

}

}

}

if (!safe)

{

printf("\nThe processes are in unsafe state\n");

break;

}

else

{

printf("The process is in safe state\n");

printf("Available resources: ");

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

{

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

}

printf("\n\n");

}

}

return 0;

}