22f-3241

22f-3291

Os project

Start:

#include<iostream>

#include<string>

#include<unistd.h>

#include<stdlib.h>

#include<pthread.h>

#include <bits/stdc++.h>

#include<algorithm>

#include<string.h>

#include<sys/shm.h>

#include<stdio.h>

#include<queue>

using namespace std;

queue<int> wqueue;

queue<int> rqueue;

void showq(queue<int> wqueue)

{

    queue<int> g = wqueue;

    while (!g.empty()) {

        cout << '\t' << g.front();

        g.pop();

    }

    cout << '\n';

}

bool flag=true;

int i;

pthread\_t thrTask[15];///thread pool for our 2o tasks

int n=15;//to keep the value up to which our loops have to operate

int \*currentIDArr = new int [15];///creating ids for each process

//array for setting the priority for each process

int \*currentPriorityArr = new int[15];

int IDPosition = 0;//process index track

int positionPriority = 0;//priority array index track

int RAM;//variable in which we store the ram that we will store the ram given by the user

int \*shared\_memory;

struct Task

{//struct for kepping the values that are to identify and stored each process

  int id;

  string name;

  int ramUse; //in MBs

  int priority; //Higher the number ,higher the  priority

};

Task t[15];

//the scheduling algorithm that we will be using for the process

void priorityScheduling(){

     for(int i=1;i<n;i++)

     {

      for(int j=1;j<n;j++)

       {

        if(t[i].priority > t[j].priority)//higher priority comes first

        {

         swap(t[i].id,t[j].id);

         swap(t[i].name,t[j].name);

         swap(t[i].ramUse,t[j].ramUse);

         swap(t[i].priority,t[j].priority);

        }

      }

     }

  }

  ///function for showing each process with all its properties

void showAll(){

    for(int i=0;i<n;i++) {

        cout<<t[i].id<<"\t"<<t[i].name<<"\t\t"<<t[i].ramUse<<"\t"<<t[i].priority<<endl;

    }

}

//function that we will use if a process is stoped and the next is choosen

//this is simply for contexting switching

int ContextSwitch(int ID, Task t[]){

    int pri;

///this checks basiclly if the process choosen by the user has a higher or lower priority than the process that is before it in the array.

    for(int i = 0;i<n;i++){

        if(ID == t[i].id){

            pri = t[i].priority;

            break;

        }

    }

    return pri;

}

void board()

{

   cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

   for(int i=0;i<9;i++)

   {

     cout<<"\*\t\t\t\t\t\t\t\t\t\t\t\t\*"<<endl;

   }

        cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

    cout<<"\t\t\tLoading operating system"<<endl;

    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

   for(int i=0;i<9;i++)

   {

     cout<<"\*\t\t\t\t\t\t\t\t\t\t\t\t\*"<<endl;

   }

   cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

   sleep(5);

   system("clear");

   cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

   for(int i=0;i<9;i++)

   {

     cout<<"\*\t\t\t\t\t\t\t\t\t\t\t\t\*"<<endl;

   }

    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

    cout << "\t\t\tWELCOME TO OS WORLD!"<<endl;

    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

     for(int i=0;i<9;i++)

   {

     cout<<"\*\t\t\t\t\t\t\t\t\t\t\t\t\*"<<endl;

   }

   cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

   sleep(5);

}

/////proccess1////////

/////calculator/////////

void \*calculator(void \*args)

{

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ calculator.cpp && ./a.out\"'");

return  NULL;

}

/////proccess2////////

/////Tic Tac Toe/////////

void \*tictactoe(void \*args){

sleep(5);

 system("gnome-terminal -e  'sh -c \"g++ tictactoe.cpp && ./a.out\"'");

 return NULL;

}

/////proccess3////////

/////binarysearch/////////

void \*binar\_search(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ binarysearch.cpp && ./a.out\"'");

 return NULL;

}

/////proccess4////////

/////Banking System/////////

void \*banking(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ banking.cpp && ./a.out\"'");

 return NULL;

}

/////proccess5////////

/////Guessing Game/////////

void \*Guessing\_game(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ guessgame.cpp && ./a.out\"'");

 return NULL;

}

/////proccess6////////

/////Message Box/////////

void \*message(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ messagebox.cpp && ./a.out\"'");

 return NULL;

}

/////proccess7////////

/////Create File/////////

void \*createfile(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ createfile.cpp && ./a.out\"'");

return  NULL;

}

/////proccess8////////

/////Delete File/////////

void \*delete\_file(void \*args){

 sleep(5);

system("gnome-terminal -e  'sh -c \"g++ deletefile.cpp && ./a.out\"'");

 return NULL;

}

/////proccess9////////

/////calender/////////

void \*calender(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ calender.cpp && ./a.out\"'");

return  NULL;

}

/////proccess10////////

/////time/////////

void \*factorial(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ factorial.cpp && ./a.out\"'");

 return NULL;

}

/////proccess11////////

/////time/////////

void \*time(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ time.cpp && ./a.out\"'");

 return NULL;

}

/////proccess12////////

/////Stop Watch/////////

void \*stopwatch(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ stopwatch.cpp && ./a.out\"'");

 return NULL;

}

/////proccess13////////

/////Length of String/////////

void \*length(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ stringlen.cpp && ./a.out\"'");

 return NULL;

}

/////proccess14////////

/////Find Prime/////////

void \*prime(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ findprime.cpp && ./a.out\"'");

 return NULL;

}

/////proccess15////////

/////Hangman Game/////////

void \*hangman\_game(void \*args){

sleep(5);

system("gnome-terminal -e  'sh -c \"g++ hangman.cpp && ./a.out\"'");

 return NULL;

}

///this function is used for telling which process or task with how much prority and ram useage is runnig.

//////23///////

void \*TaskManager(void \*args){

cout << "ID\tName\t\tRam"<<endl;

    int ID;

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

        ID = currentIDArr[i];

        cout << ID << "\t" << t[ID].name << "\t" << t[ID].ramUse<<endl;

    }

sleep(8);

 return NULL;

}

void userFunc()

{

    bool cond1=1;

    int choice1,pri;

    while(cond1)

    {

        system("clear");

        cout<<"\t-\*-\*-\*-\*-\*-\*-\*-\*-"<<"Total Ram is " << \*shared\_memory<<"\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

        cout<<"\t\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

            cout << "\t\t\t\tMAIN MENU"<<endl;

            cout<<"\t\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

        cout<<"1. Calculator.\n";

        cout<<"2. Tic Tac Toe.\n";

        cout<<"3. Binary Search\n";

        cout<<"4. Banking System \n";

        cout<<"5. Guessing Game.\n";

        cout<<"6. Message Box.\n";

        cout<<"7. Create file.\n";

        cout<<"8. Delete file.\n";

        cout<<"9. Calendar.\n";

        cout<<"10. Time.\n";

        cout<<"11. Find Factorial.\n";

        cout<<"13. length of string.\n";

        cout<<"14. Find Prime.\n";

        cout<<"15. Hangman.\n";

        cout<<"12. Stop Watch.\n";

        cout<<"16.Shutdown.\n";

        cout<<"Enter choice";

        cin>>choice1;

        if(t[choice1-1].ramUse>\*shared\_memory)

        {

        wqueue.push(t[choice1-1].id);

        }

        if( !wqueue.empty())

        {

         int tempid=wqueue.front();

         wqueue.pop();

           for(i=0;i<15&&flag!=false;i++)

            {

               if(t[i].id==tempid)

               {

                  flag=false;

              }

            }

            if(t[i].ramUse>=\*shared\_memory)

            {

                rqueue.push(tempid);

        currentIDArr[IDPosition] = t[choice1-1].id;

        IDPosition++;

        if(RAM>t[choice1-1].ramUse)

          {

            RAM = RAM - t[choice1-1].ramUse;

            //add currnet position in arr

            currentPriorityArr[positionPriority] = t[choice1-1].priority;

            pri = ContextSwitch(choice1, t);

            if(pri > currentPriorityArr[positionPriority] )

            {

               RAM = RAM + 2;

            }

            positionPriority++;

        cout<<"-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

        if( !rqueue.empty())

        {

            int tempid=rqueue.front();

            rqueue.pop();

            for(i=0;i<15&&flag!=false;i++)

            {

               if(t[i].id==tempid)

               {

                  flag=false;

              }

            }

            flag=true;

            if(t[i].name=="Calculator")

            {

                  cout<<"Opening the calculator"<<endl;

                  pthread\_create(&thrTask[0],NULL,&calculator,NULL);

                      pthread\_join(thrTask[0],NULL);

                      cout<<"The process for calculaotr is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

            else if(t[i].name=="TIC TAC TOE")

            {

                 cout<<"Opening the TIC TAC TOE"<<endl;

                  pthread\_create(&thrTask[1],NULL,&tictactoe,NULL);

                      pthread\_join(thrTask[1],NULL);

                      cout<<"The process for tic tac toe is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Binary Search")

            {

                 cout<<"Opening the binary search"<<endl;

                  pthread\_create(&thrTask[2],NULL,&binar\_search,NULL);

                      pthread\_join(thrTask[2],NULL);

                      cout<<"The process for binary search is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Banking System")

            {

                 cout<<"Opening the banking "<<endl;

                  pthread\_create(&thrTask[3],NULL,&banking,NULL);

                      pthread\_join(thrTask[3],NULL);

                      cout<<"The process for banking is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Guessing Game")

            {

                 cout<<"Opening the guessing game"<<endl;

                  pthread\_create(&thrTask[4],NULL,&Guessing\_game,NULL);

                      pthread\_join(thrTask[4],NULL);

                      cout<<"The process for guessing game is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Message Box")

            {

                 cout<<"Opening the message box"<<endl;

                  pthread\_create(&thrTask[5],NULL,&message,NULL);

                      pthread\_join(thrTask[5],NULL);

                      cout<<"The process for message box is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Create File")

            {

                 cout<<"Opening the create file"<<endl;

                  pthread\_create(&thrTask[6],NULL,&createfile,NULL);

                      pthread\_join(thrTask[6],NULL);

                      cout<<"The process for create file is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Delete File")

            {

                 cout<<"Opening the delete file"<<endl;

                  pthread\_create(&thrTask[7],NULL,&delete\_file,NULL);

                      pthread\_join(thrTask[7],NULL);

                      cout<<"The process for delete file is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Calender")

            {

                 cout<<"Opening the calender"<<endl;

                  pthread\_create(&thrTask[8],NULL,&calender,NULL);

                      pthread\_join(thrTask[8],NULL);

                      cout<<"The process for calender is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

              else if(t[i].name=="Time Task")

            {

                 cout<<"Opening the clock"<<endl;

                  pthread\_create(&thrTask[9],NULL,&time,NULL);

                      pthread\_join(thrTask[9],NULL);

                      cout<<"The process for clock is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Find Factorial")

            {

                 cout<<"Opening the find factorial proccess"<<endl;

                  pthread\_create(&thrTask[10],NULL,&factorial,NULL);

                      pthread\_join(thrTask[10],NULL);

                      cout<<"The process for finding factorial is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

              else if(t[i].name=="String Length")

            {

                 cout<<"Opening the string length proccess"<<endl;

                  pthread\_create(&thrTask[11],NULL,&length,NULL);

                      pthread\_join(thrTask[11],NULL);

                      cout<<"The process for find string length is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

              else if(t[i].name=="Find prime")

            {

                 cout<<"Opening the find prime proccess"<<endl;

                  pthread\_create(&thrTask[12],NULL,&prime,NULL);

                      pthread\_join(thrTask[12],NULL);

                      cout<<"The process for finding prime is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Hangman Task")

            {

                 cout<<"Opening the hangman game"<<endl;

                  pthread\_create(&thrTask[13],NULL,&hangman\_game,NULL);

                      pthread\_join(thrTask[13],NULL);

                      cout<<"The process for hangman game is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

             else if(t[i].name=="Stop Watch")

            {

                 cout<<"Opening the stop watch"<<endl;

                  pthread\_create(&thrTask[14],NULL,&stopwatch,NULL);

                      pthread\_join(thrTask[14],NULL);

                      cout<<"The process for stop watch is executed"<<endl;

                      RAM = RAM + t[i].ramUse;

                      sleep(2);

            }

        }

        }

          }

          }

          else

          {

              if(choice1==1)

               {

                  cout<<"Opening the calculator"<<endl;

                  pthread\_create(&thrTask[0],NULL,&calculator,NULL);

                      pthread\_join(thrTask[0],NULL);

                      cout<<"The process for calculaotr is executed"<<endl;

                      RAM = RAM + t[choice1-1].ramUse;

                      sleep(2);

                   }

            else if(choice1==2)

              {

                cout<<"Opening TIC TAC TOE "<<endl;

                pthread\_create(&thrTask[1],NULL,&tictactoe,NULL);

                pthread\_join(thrTask[1],NULL);

                cout<<"The process for Tic tac toe is executed"<<endl;

                RAM = RAM + t[choice1-1].ramUse;

                    sleep(2);

                  }

          else if(choice1==3)

            {

               cout<<"Proccess for binary search is opening"<<endl;

               pthread\_create(&thrTask[2],NULL,&binar\_search,NULL);

               pthread\_join(thrTask[2],NULL);

               cout<<"The process for binary search is executed"<<endl;

               RAM = RAM + t[choice1-1].ramUse;

                   sleep(2);

                }

        else if(choice1==4)

           {

              cout<<"opening banking system "<<endl;

              pthread\_create(&thrTask[3],NULL,&banking,NULL);

              pthread\_join(thrTask[3],NULL);

              cout<<"The process for banking system is executed"<<endl;

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

           }

        else if(choice1==5)

           {

              cout<<"Opening guessing game "<<endl;

              pthread\_create(&thrTask[4],NULL,&Guessing\_game,NULL);

              pthread\_join(thrTask[4],NULL);

              cout<<"The proccess for guessing game is executed"<<endl;

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

           }

        else if(choice1==6)

           {

              cout<<"Opening Message Box"<<endl;

              pthread\_create(&thrTask[5],NULL,&message,NULL);

              pthread\_join(thrTask[5],NULL);

              cout<<"The proccess for Message Box is executed"<<endl;

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

        else if(choice1==7)

          {

              cout<<"Opening create file"<<endl;

              pthread\_create(&thrTask[6],NULL,&createfile,NULL);

              pthread\_join(thrTask[6],NULL);

              cout<<"The proccess for creating file is executed"<<endl;

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

        else if(choice1==8)

          {

              cout<<"Opening Delete file process"<<endl;

              pthread\_create(&thrTask[7],NULL,&delete\_file,NULL);

              pthread\_join(thrTask[7],NULL);

              cout<<"The proccess for deleting file is executed"<<endl;

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

        else if(choice1==9)

          {

              cout<<"Opening calender process"<<endl;

              pthread\_create(&thrTask[8],NULL,&calender,NULL);

              pthread\_join(thrTask[8],NULL);

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

           else if(choice1==10)

          {

              cout<<"clock opening..."<<endl;

              pthread\_create(&thrTask[9],NULL,&time,NULL);

              pthread\_join(thrTask[9],NULL);

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

           else if(choice1==11)

          {

              cout<<"Factorial Proccess opening..."<<endl;

              pthread\_create(&thrTask[10],NULL,&factorial,NULL);

              pthread\_join(thrTask[10],NULL);

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

           else if(choice1==12)

          {

             cout<<"Length of string proccess opening..."<<endl;

             pthread\_create(&thrTask[11],NULL,&length,NULL);

             pthread\_join(thrTask[11],NULL);

             RAM = RAM + t[choice1-1].ramUse;

                 sleep(2);

          }

           else if(choice1==13)

          {

              cout<<"Find Prime Proccess opening..."<<endl;

              pthread\_create(&thrTask[12],NULL,&prime,NULL);

              pthread\_join(thrTask[12],NULL);

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

            else if(choice1==14)

         {

             cout<<"Hangman game opening..."<<endl;

             pthread\_create(&thrTask[13],NULL,&hangman\_game,NULL);

             pthread\_join(thrTask[13],NULL);

             RAM = RAM + t[choice1-1].ramUse;

                 sleep(2);

         }

           else if(choice1==15)

          {

              cout<<"Stop watch opening..."<<endl;

              pthread\_create(&thrTask[14],NULL,&stopwatch,NULL);

              pthread\_join(thrTask[14],NULL);

              RAM = RAM + t[choice1-1].ramUse;

                  sleep(2);

          }

          else if(choice1==16)

          {

                cout<<"SHUTTING DOWN"<<endl;

                sleep(5);

                system("clear");

                return;

          }

          else

         {

             cout<<"Invalid task no!"<<endl;

         }

    }

        }

    }

void \*initialize(void \*args){

   t[0].id=1; t[0].name="Calculator";

   t[0].ramUse=10;t[0].priority=1;

   t[1].id=2; t[1].name="TIC TAC TOE";

   t[1].ramUse=30;t[1].priority=2;

   t[2].id=3; t[2].name="Binary Search";

   t[2].ramUse=40;t[2].priority=4;

   t[3].id=4; t[3].name="Banking System";

   t[3].ramUse=2; t[3].priority=3;

   t[4].id=5; t[4].name="Guessing Game";

   t[4].ramUse=2; t[4].priority=3;

   t[5].id=6; t[5].name="Message Box";

   t[5].ramUse=30;t[5].priority=6;

   t[6].id=7; t[6].name="Create File";

   t[6].ramUse=6; t[6].priority=8;

   t[7].id=8; t[7].name="Delete File";

   t[7].ramUse=3; t[7].priority=7;

   t[8].id=9; t[8].name="Calender";

   t[8].ramUse=10;t[8].priority=9;

   t[9].id=10;t[9].name="Time Task";

   t[9].ramUse=30;t[9].priority=10;

   t[10].id=11;t[10].name="Find Factorial";

   t[10].ramUse=11;t[10].priority=11;

   t[11].id=12;t[11].name="String Length";

   t[11].ramUse=11;t[11].priority=12;

   t[12].id=13;t[12].name="Find prime";

   t[12].ramUse=20;t[12].priority=9;

   t[13].id=14;t[13].name="Hangman Task";

   t[13].ramUse=15;t[13].priority=19;

   t[14].id=15;t[14].name="Stop Watch";

   t[14].ramUse=11;t[14].priority=15;

    pthread\_exit(NULL);

}

int main(int argc, char\*\* num)

{

    string r = string(num[1]);

    RAM = stoi(r);

    string c = string(num[2]);

    int Cores = stoi(c);

    string d = string(num[3]);

    int Disk = stoi(d);

    bool cond1=1;

    int choice1;

        pthread\_t thr;

        pthread\_create(&thr,NULL,&initialize,NULL);  //loading tasks

        pthread\_join(thr,NULL);

    string pass;

    string kernalpass="1122";

    system("clear");

int shmid;

shmid=shmget((key\_t)1122,sizeof(int),0666|IPC\_CREAT);

shared\_memory=(int\*)shmat(shmid,NULL,0);

int data=RAM;

\*shared\_memory=data;

board();

    system("gnome-terminal -e  'sh -c \"g++ calender.cpp && ./a.out\"'");

    system("gnome-terminal -e  'sh -c \"g++ time.cpp && ./a.out\"'");

    system("clear");

    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

    cout << "\t\t\tWELCOME TO LOGIN PAGE!"<<endl;

    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

    cout<<"Welcome! How do you want to Log In? \nPlease Select from the following options:\n1. User.\n2. Kernal.\nEnter the number of selected option:";

    cin>>choice1;

    system("clear");

    switch(choice1)

    {

        case 1:

            userFunc();

            break;

        case 2:

                Task t1[50];

                for(int i=0;i<24;++i)

                {

                    t1[i].name=t[i].name;

                    t1[i].priority=t[i].priority;

                    t1[i].ramUse=t[i].ramUse;

                }

                t[50];

                for(int i=0;i<50;++i)

                {

                    t[i].name="123";

                }

                for(int i=0;i<15;++i)

                {

                    t[i].name=t1[i].name;

                    t[i].priority=t1[i].priority;

                    t[i].ramUse=t1[i].ramUse;

                }

                int count=15;

                 cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

                     cout << "\t\t\tPLEASE ENTER PASSWORD TO LOGIN!"<<endl;

                     cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

                cout<<"Enter password of kernal: ";

                cin>>pass;

                if(pass=="1122")

                {

                  cout<<"yahoo!!!password is correct:))\n";

                }

                while(pass!=kernalpass)

                {

                    cout<<"Try again OR press 0 to shutdown"<<endl;

                    cin>>pass;

                    if(pass=="0")

                       break;

                }

                int kChoice;

            do

            {

                    cout<<"1. To Add Task"<<endl;

                    cout<<"2. To Delete Task"<<endl;

                    cout<<"3. To Shutdown"<<endl;

                    cin>>kChoice;

                    if(kChoice == 1)

                    {

                    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

                                    cout << "\t\t\tWELCOME TO THE PAGE TO ADD TASK!"<<endl;

                                    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

                      for(int i = 0;i<50;i++)

                      {

                        if(t[count].name!="")

                        {

                        cout<<"Enter the name of Task:";

                        cin>>t[count].name;

                        cout<<"Enter the Priority of Task:";

                        cin>>t[count].priority;

                        cout<<"Enter the Ram usage of Task:";

                        cin>>t[count].ramUse;

                        count++;

                        break;

                        }

                      }//endFor

                         cout<<"TOTAL TASK"<<endl;

                      cout<<"Name\t\t\tPriority\t\t\tRAM"<<endl;

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

                      {

                          if(t[i].name!="")

                          {

                         cout<<t[i].name<<"\t\t\t"<<t[i].priority<<"\t\t\t"<<t[i].ramUse<<endl;

                         }

                      }

                         sleep(5);

                    }

                    else if(kChoice == 2)

                    {

                    string inp;

                    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

                                    cout << "\t\t\tWELCOME TO THE PAGE TO DELETE TASK!"<<endl;

                                    cout<<"\t\t-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"<<endl;

                    cout<<"Please enter the name of task you want to delete:";

                    cin>>inp;

                        for(int i = 0;i<50;i++)

                        {

                            if(inp==t[i].name)

                            {

                                t[i].name="";

                                t[i].id=0;

                                t[i].priority=0;

                                t[i].ramUse=0;

                                cout<<"Task deleted succesfully"<<endl;

                                break;

                            }

                        }

                             cout<<"TOTAL TASK"<<endl;

                             cout<<endl;

                             cout<<"Name\t\t\tPriority\t\t\tRAM"<<endl;

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

                      {

                          if(t[i].name!="")

                          {

                         cout<<t[i].name<<"\t\t\t"<<t[i].priority<<"\t\t\t"<<t[i].ramUse<<endl;

                         }

                      }

                       sleep(5);

                    }

            }while((kChoice==1)||(kChoice==2));

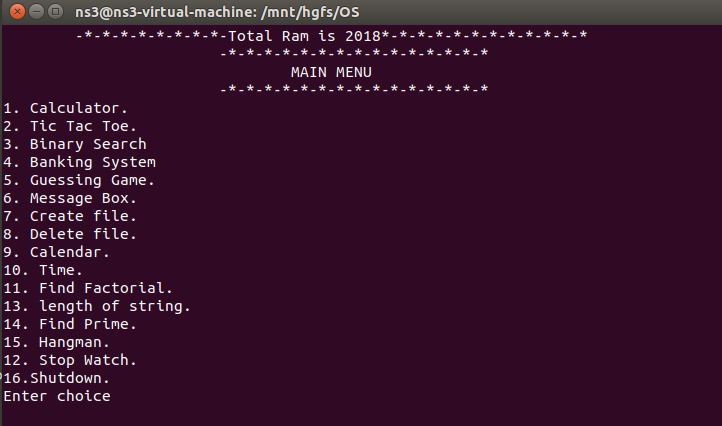
            system("clear");

            break;

    }

    return 0;

}

S

