UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS

MANUAL TECNICO

2012-12487

Alan René Bautista Barahona
SISTEMAS OPERATIVOS 1

INDICE

Introducción

El siguiente manual contiene información técnica acerca del funcionamiento, así como las partes y lógica de programación que posee la aplicación. La aplicación básicamente consiste en un juego realizado en consola con una lógica similar a la de Space Invaders, este se desarrollara en lenguajes C.

El juego será desarrollado para dos jugares en simultaneo (modo versus). La comunicación entre las instancias que consumirán el juego realizará por medio de memoria compartida, así como la selección de bandos para cada usuario se implementara el uso de semáforos del sistema.

Objetivos

- ✓ Implementar el algoritmo de Dekker para hacer uso de memoria compartida.
- ✓ Implementar semáforos del sistema operativos GNU/Linux.
- ✓ Poner en práctica los conocimientos adquiridos en el curso y laboratorio de sistemas operativos 1.

ANÁLISIS

DESCRIPCIÓN DEL PROBLEMA

Se desea crear una aplicación por medio de la cual se desarrolle un juego juego similar a “Space Invaders” en lenguajes C. El juego será desarrollado para dos jugares en simultaneo (modo versus). Para ello se deben utilizar dos instancias diferentes del mismo programa. El cual se podrá jugar utilizando conexión SSH y usuario local.

El juego consiste en bando defensor y bando invasor y contara con un sistema de puntuación para definir el bando ganador. La comunicación entre ambas instancias del programa se realizará por medio de memoria compartida, deben implementar la versión 5 del algoritmo de Dekker. También se debe hacer uso de semáforos en ambas instancias del juego.

ANÁLISIS DE REQUERIMIENTOS DEL USUARIO.

- ✓ El usuario necesita una interfaz agradable y fácil de entender.
- ✓ Una manera intuitiva para poder movilizarse dentro de la aplicación.
- ✓ Un indicador que permita ver el progreso del mismo ademas de informar a los usuarios cuando el juego termine.

PLANTEO INICIAL DE LA SOLUCIÓN.

El juego se debe desarrollar en lenguajes C y debe implementar graficas basadas en texto que hagan del juego más llamativo. Se planea el uso de librería Ncurser para poder presentar una interfase agradable.

La aplicación funcionara con memoria compartida ya se busca mostrar la misma información en las instancias que estén conectadas. El juego se iniciara cuando los dos jugares estén listos, así mismo terminará cuando uno de los dos jugadores se quede sin vidas o el score del defensor sea suficiente.

El juego contara con dos bandos: defensor e invasor.

Defensor: es el encargado de derribar la línea de naves invasoras y comandante.

Invasor: es el comandante de las líneas de naves invasoras y es el encargado de indicar que nave debe disparar.

BUENAS PRÁCTICAS DE PROGRAMACIÓN A IMPLEMENTAR.

Realizar un análisis estructurado y lógico de lo que se de hacer siempre es una buena práctica para implementar de la forma más apropiada los distintos métodos para resolver un problema, además el hecho de planificar y revisar las acciones que realizara nuestro programa al momento de ejecutarse, siempre nos permite encontrar posibles erres de compilación o ejecución, errores que se pueden corregir con el simple hecho se ponerles más atención tratando de evitarlos.

Al momento de realizar cada uno de los algoritmos o implementar distintos métodos para resolver cada uno de los problemas, es bueno verificar que el software se desarrolle de la forma deseada o esperada, y que los métodos, clases, algoritmos, etc… que acabamos de implementar no interfieran con el buen funcionamiento de los que ya tenemos creados y funcionando.

Investigar siempre la mejor manera de resolver un problema, ya que existen muchas formas de resolverla misma problemática, pero unas se ajustan de mejor forma dependiendo las circunstancias en la que esta se esté requiriendo.

DISEÑO

MAIN.C

```
#include <stdio.h>
#include <stdlib.h>
#include <ncurses.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/shm.h>
#include <stdbool.h>

#define _GNU_SOURCE
#include <string.h>

#include <time.h>
#include <stdlib.h>
#include <sys/sem.h>

#define DELAY 30000

#ifdef MUTEX
/* mutex (EM) para sincronizar el acceso a buffer */
pthread_mutex_t mutexBuffer;
#endif

void izquierda();
void derecha();
void arriba();
void abajo();
void control();
void Reinicio();
void *funcionThread (void *parametro);
void *funcionThread2 (void *parametro);
void *escribir(void *parametro);
void *leer(void *parametro);
void *reloj(void *parametro);

int x=0,y=0;
int max_x=0,max_y=0;
FILE *archivocompartido;
key_t clavecompartida;
int mem = 0,error_hilo=0;
int *mcompartida = NULL;
pthread_t idHilo;
pthread_t rhilo;
pthread_t whilo;
pthread_t chilo;
bool p1_entra,p2_entra;
int turno,vida=0,ptos=0;
int jugador=0,aliens=0,rival=0,vidarival=5,ptorival=0;
int objetivos[20];
int minutos=0;
int segundos=0;
int vidarival2=5;
bool ledi=false;

struct sembuf procc;
struct sembuf procc2;
key_t llave;
int identificadorSEM;

#if defined(_GNU_LIBRARY_) && !defined(_SEM_SEMUN_UNDEFINED)
#endif
```

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

INICIO:

    archivocompartido = fopen("/tmp/acompartido","w+");
    clavecompartida = ftok ("/tmp/acompartido",33);
    mem = shmget(clavecompartida,sizeof(int *)*100,0777 | IPC_CREAT);
    mcompartida = (int *) shmat(mem,(char *)0,0);
    //informacion de memoria compartida
    initscr();                                  /* Start curses mode                        */
    curs_set(FALSE);
    getmaxyx(stdscr, max_y, max_x);
    clear();
    printw("                        SAPCE INVADERS\n");
    printw("\n\n");
    printw("                          Bienvenido");
    printw("\n\n\n");
    printw("            ----------------------------------------\n");
    printw("           |      Precione Enter para Continuar      |\n");
    printw("            ----------------------------------------\n");
    printw("\n\n\n\n");
    printw("             Alan Rene Bautista Barahona - 201212487\n");
    getch();
    getch();
    clear();

    //pal semaforo
    llave = ftok ("/bin/ls", 33);
    identificadorSEM = semget (llave, 10, 0600 | IPC_CREAT);


    printw("_____________________________________________________________________________\n");
    printw("\n\n\n\n");
    printw("            ----------------------------------------\n");
    printw("           |      Jugador 1 -> DEFENSOR <-           |\n");
    printw("           |            (precione a)                 |\n");
    printw("            ----------------------------------------\n");
    printw("\n\n\n\n");
    printw("            ----------------------------------------\n");
    printw("           |      Jugador 2 -> INVASOR <-            |\n");
    printw("           |            (precione b)                 |\n");
    printw("            ----------------------------------------\n");
    printw("\n\n\n\n");
    printw("_____________________________________________________________________________\n");

    jugador=getch();
    vida=5;
    ptos=0;
    mcompartida[8]=1;
    mcompartida[9]=1;
    mcompartida[10]=1;
    mcompartida[11]=1;
    mcompartida[12]=1;
    mcompartida[13]=1;
    mcompartida[14]=1;
    mcompartida[15]=1;
    mcompartida[16]=1;
    mcompartida[17]=1;
    mcompartida[18]=1;
    mcompartida[19]=1;
    mcompartida[20]=1;
    mcompartida[21]=1;
    mcompartida[22]=1;
    mcompartida[23]=1;
    mcompartida[24]=1;
```

```
mcompartida[25]=1;
mcompartida[26]=1;
mcompartida[27]=1;

//estado de las naves
objetivos[1] = 1;
objetivos[2] = 1;
objetivos[3] = 1;
objetivos[4] = 1;
objetivos[5] = 1;
objetivos[6] = 1;
objetivos[7] = 1;
objetivos[8] = 1;
objetivos[9] = 1;
objetivos[10] = 1;
objetivos[11] = 1;
objetivos[12] = 1;
objetivos[13] = 1;
objetivos[14] = 1;
objetivos[15] = 1;
objetivos[16] = 1;
objetivos[17] = 1;
objetivos[18] = 1;
objetivos[19] = 1;
objetivos[20] = 1;

turno=1;
p1_entra=false;
p2_entra=false;

switch(jugador)
{
        //asignacion de variabels para jugadores segun sea el caso
   case 'a':

            printw("          ------> LISTO!!! Esperando al despreciable invasor <------          \n");
            getch();
            semctl (identificadorSEM, 0, SETVAL, 0);

      procc.sem_num = 0;
      procc.sem_op = -1;
      procc.sem_flg = 0;

            semop (identificadorSEM, &procc, 1);

      jugador=1;
      mcompartida[0]=1;
      mcompartida[1]=x;
      mcompartida[2]=vida;
      mcompartida[3]=ptos;
      error_hilo= pthread_create (&rhilo, NULL, leer, NULL);
      error_hilo= pthread_create (&whilo, NULL, escribir, NULL);
      error_hilo= pthread_create (&chilo, NULL, reloj, NULL);
      break;

   default:

      procc2.sem_num = 0;
      procc2.sem_op = 1;
      procc2.sem_flg = 0;

      int i;
      for (i = 0; i<1; i++)
      {
         printf("AL ATAQUE!!! \n");
         semop (identificadorSEM, &procc2, 1);
         sleep (1);
```

```
            }

            jugador=2;
            mcompartida[4]=1;
            mcompartida[5]=x;
            mcompartida[6]=vida;
            mcompartida[7]=ptos;
            error_hilo= pthread_create (&rhilo, NULL, leer, NULL);
            error_hilo= pthread_create (&whilo, NULL, escribir, NULL);
            error_hilo= pthread_create (&chilo, NULL, reloj, NULL);
            break;
    }

//          error_hilo= pthread_create (&idHilo, NULL, funcionThread2, NULL);
#ifdef MUTEX
            /* Se inicia el mutex (EM) */
            pthread_mutex_init (&mutexBuffer, NULL);
#endif
    usleep(DELAY); // delay entre moviminetos

    control();
    shmdt ((char *)mcompartida);
    shmctl (mem, IPC_RMID,(struct shmid_ds *)NULL); //finaliza el recurso de memoria
    unlink ("\tmp\acompartido");
            return 0;
}

void control()
{

#ifdef MUTEX
                        /* Esperamos y bloqueamos el mutex (EM) */
                        pthread_mutex_lock (&mutexBuffer);
#endif
    clear();
    y=max_y-4;
    int rivaly=3;
    int ch;
            initscr();
            raw();
            keypad(stdscr, TRUE);                   /* get F1, F2 etc..             */
            noecho();                               /* que no se repita o se quede esperando miestras se hace el getch*/
    int iterador=0;
    getmaxyx(stdscr, max_y, max_x);

    if((vidarival==1) || (vida==1) || (ptos>=100))
    {
          printw("Finish Him!!!\n");
               char *puntos="Puntos: ";
          char ptos1[15];
          char ptos2[15];
          char *tiempo="Tiempo: ";
          char tiempov[15];
          sprintf(tiempov,"%s%d\n",tiempo,minutos);
          if(jugador==1)
          {
             sprintf(ptos1,"%s%d\n",puntos,ptos);
             sprintf(ptos2,"%s%d\n",puntos,ptorival);
          }else
          {
             sprintf(ptos2,"%s%d\n",puntos,ptos);
             sprintf(ptos1,"%s%d\n",puntos,ptorival);
          }
          if(jugador==1)
          {
             if(ptorival>ptos)
             {
```

```
                printw("Ganador Invasor \n");
                printw(ptos2);
                printw("Perdedor Defensor \n");
                printw(ptos1);
            }else
            {
                printw("Ganador Defensor \n");
                printw(ptos1);
                printw("Perdedor Invasor \n");
                printw(ptos2);
            }
        }else
        {
            if(ptorival>ptos)
            {
                printw("Ganador Defensor \n");
                printw(ptos1);
                printw("Perdedor Invasor \n");
                printw(ptos2);
            }
            else
            {
                printw("Ganador Invasor \n");
                printw(ptos2);
                printw("Perdedor Defensor \n");
                printw(ptos1);
            }
        }
        printw(tiempov);
        printw("\n");
        ch = getch();
             //goto INICIO;
             Reinicio();
}
clear();
for(iterador=0;iterador<max_x-4;iterador++)
         {
            mvprintw(2,iterador,"_");
            mvprintw(max_y-3,iterador,"_");
         }

for(iterador=0;iterador<max_y;iterador++)
         {
            mvprintw(iterador,max_x-10,"|");
         }

char *clock1="Tiempo: ";
char clock2[5];
sprintf(clock2,"%s%d",clock1,minutos);
mvprintw(max_y/2,max_x-10,clock2);
          char *info1="Puntos ";
          char info2[10];
          sprintf(info2,"%s%d",info1,ptos);
          char info3[10];
          sprintf(info3,"%s%d",info1,ptorival);
          char *nombre1="Defensor ";
          char *nombre2="Invasor ";
          char name1[9];
          char name2[9];
          if(jugador==1)
          {
             sprintf(name1,"%s%d",nombre1,vida);
             mvprintw(max_y-2,0,name1);
             mvprintw(max_y-4,max_x-10,info2);
             sprintf(name2,"%s%d",nombre2,vidarival);
             mvprintw(0,0,name2);
             mvprintw(0,max_x-10,info3);
```

```
            }
            else
            {
               sprintf(name1,"%s%d",nombre2,vida);
               mvprintw(max_y-2,0,name1);
               mvprintw(max_y-4,max_x-10,info2);
               sprintf(name2,"%s%d",nombre1,vidarival);
      mvprintw(0,0,name2);
               mvprintw(0,max_x-10,info3);
            }
            mvprintw(rivaly,max_x-15-rival,"<--->");
   int espaciado=max_x/5;
   int mitady=max_y/2;
   int enemigos = 1;



   srand(time(NULL));   // should only be called once
   int r1 = rand() % 19;
   int r2 = rand() % 19;
   int r3 = rand() % 19;
   int r4 = rand() % 19;
   int Cualuno = 1;

   for(iterador=0;iterador<20;iterador++)
   {

/*
      if(iterador<5)
      {
         if(objetivos[iterador]==1) mvprintw(mitady+2,(espaciado*iterador)+1,"\\-.-/");
      }
      else if(iterador>9)
      {
         if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(iterador-10))+1,"\\-.-/");
      }
      else
      {
         char cadena[128] = "/-";
         char texto[100];

         sprintf(texto, "%d", enemigos);
         strcat(cadena, texto);
         strcat(cadena, "-\\");

         if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,cadena);
         enemigos = enemigos +1;
      }
*/

///*



      if(iterador < 5 )
           {
                if(iterador==r1+1)
                {
                     if(Cualuno == 1 )
           {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(0))+1,"/-1-\\");
              Cualuno = 2;
           }
                else if(Cualuno == 2 )
           {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*1)+1,"/-1-\\");
              Cualuno = 3;
```

```
        }
            else if(Cualuno == 3 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(2))+1,"/-1-\\");
           Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(3))+1,"/-1-\\");
           Cualuno = 5;
        }
            else if(Cualuno == 5 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(4))+1,"/-1-\\");
           Cualuno = 1;
        }
        //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-1-\\");
        enemigos = enemigos +1;
}
else if(iterador==r2+1)
{
            if(Cualuno == 1 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(0))+1,"/-2-\\");
           Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*1)+1,"/-2-\\");
           Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(2))+1,"/-2-\\");
           Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(3))+1,"/-2-\\");
           Cualuno = 4;
        }
            else if(Cualuno == 5 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(4))+1,"/-2-\\");
           Cualuno = 1;
        }
            //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-2-\\");
        enemigos = enemigos +1;
}
else if(iterador==r3+1)
{
            if(Cualuno == 1 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(0))+1,"/-3-\\");
           Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*1)+1,"/-3-\\");
           Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(2))+1,"/-3-\\");
           Cualuno = 4;
        }
            else if(Cualuno == 4 )
```

```
        {
            if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(3))+1,"/-3-\\");
          Cualuno = 5;
        }
          else if(Cualuno == 5 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(4))+1,"/-3-\\");
          Cualuno = 1;
        }
        //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-3-\\");
        enemigos = enemigos +1;
}
else if(iterador==r4+1)
{
          if(Cualuno == 1 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(0))+1,"/-4-\\");
          Cualuno = 2;
        }
          else if(Cualuno == 2 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*1)+1,"/-4-\\");
          Cualuno = 3;
        }
          else if(Cualuno == 3 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(2))+1,"/-4-\\");
          Cualuno = 4;
        }
          else if(Cualuno == 4 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(3))+1,"/-4-\\");
          Cualuno = 5;
        }
          else if(Cualuno == 5 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(4))+1,"/-4-\\");
          Cualuno = 1;
        }
          //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-4-\\");
        enemigos = enemigos +1;
}
else
{
        if(Cualuno ==1 )
        {
          if(objetivos[iterador]==1) mvprintw(mitady-+1,(espaciado*(0))+1,"\\-.-/");
            //mvprintw(mitady+2,(espaciado*(iterador+5))+1,"\\-.-/");
          Cualuno = 2;
        }
          else if(Cualuno ==2 )
        {
          if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*1)+1,"\\-.-/");
            //mvprintw(mitady+1,(espaciado*iterador)+1,"\\-.-/");
          Cualuno = 3;
        }
          else if(Cualuno ==3 )
        {
          if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(2))+1,"\\-.-/");
            //mvprintw(mitady,(espaciado*(iterador-5))+1,"\\-.-/");
          Cualuno = 4;
        }
          else if(Cualuno == 4 )
        {
          if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(3))+1,"\\-.-/");
            //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-.-/");
          Cualuno = 5;
```

```
                }
                    else if(Cualuno == 5 )
                {
                   if(objetivos[iterador]==1) mvprintw(mitady+1,(espaciado*(4))+1,"\\-.-/");
                        //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-.-/");
                   Cualuno = 1;
                }
        }
}

if(iterador > 5 && iterador < 10)
{
        if(iterador==r1+1)
        {
                    if(Cualuno == 1 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(0))+1,"/-1-\\");
                   Cualuno = 2;
              }
                    else if(Cualuno == 2 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*1)+1,"/-1-\\");
                   Cualuno = 3;
              }
                    else if(Cualuno == 3 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(2))+1,"/-1-\\");
                   Cualuno = 4;
              }
                    else if(Cualuno == 4 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(3))+1,"/-1-\\");
                   Cualuno = 5;
              }
                    else if(Cualuno == 5 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(4))+1,"/-1-\\");
                   Cualuno = 1;
              }
              //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-1-\\");
              enemigos = enemigos +1;
        }
        else if(iterador==r2+1)
        {
                    if(Cualuno == 1 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(0))+1,"/-2-\\");
                   Cualuno = 2;
              }
                    else if(Cualuno == 2 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*1)+1,"/-2-\\");
                   Cualuno = 3;
              }
                    else if(Cualuno == 3 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(2))+1,"/-2-\\");
                   Cualuno = 4;
              }
                    else if(Cualuno == 4 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(3))+1,"/-2-\\");
                   Cualuno = 4;
              }
                    else if(Cualuno == 5 )
              {
                    if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(4))+1,"/-2-\\");
```

```
            Cualuno = 1;
        }
            //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-2-\\");
        enemigos = enemigos +1;
}
else if(iterador==r3+1)
{
            if(Cualuno == 1 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(0))+1,"/-3-\\");
            Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*1)+1,"/-3-\\");
            Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(2))+1,"/-3-\\");
            Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(3))+1,"/-3-\\");
            Cualuno = 5;
        }
            else if(Cualuno == 5 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(4))+1,"/-3-\\");
            Cualuno = 1;
        }
        //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-3-\\");
        enemigos = enemigos +1;
}
else if(iterador==r4+1)
{
            if(Cualuno == 1 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(0))+1,"/-4-\\");
            Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*1)+1,"/-4-\\");
            Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(2))+1,"/-4-\\");
            Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(3))+1,"/-4-\\");
            Cualuno = 5;
        }
            else if(Cualuno == 5 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(4))+1,"/-4-\\");
            Cualuno = 1;
        }
            //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-4-\\");
        enemigos = enemigos +1;
}
else
{
```

```
                    if(Cualuno ==1 )
                    {
                       if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(0))+1,"\\-./");
                            //mvprintw(mitady+2,(espaciado*(iterador+5))+1,"\\-./");
                       Cualuno = 2;
                    }
                      else if(Cualuno ==2 )
                    {
                       if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*1)+1,"\\-./");
                            //mvprintw(mitady+1,(espaciado*iterador)+1,"\\-./");
                       Cualuno = 3;
                    }
                      else if(Cualuno ==3 )
                    {
                       if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(2))+1,"\\-./");
                            //mvprintw(mitady,(espaciado*(iterador-5))+1,"\\-./");
                       Cualuno = 4;
                    }
                      else if(Cualuno == 4 )
                    {
                       if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(3))+1,"\\-./");
                            //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-./");
                       Cualuno = 5;
                    }
                      else if(Cualuno == 5 )
                    {
                       if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(4))+1,"\\-./");
                            //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-./");
                       Cualuno = 1;
                    }
               }

}

if(iterador > 10 && iterador < 15)
{
               if(iterador==r1+1)
               {
                        if(Cualuno == 1 )
                    {
                        if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(0))+1,"/-1-\\");
                       Cualuno = 2;
                    }
                      else if(Cualuno == 2 )
                    {
                        if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*1)+1,"/-1-\\");
                       Cualuno = 3;
                    }
                      else if(Cualuno == 3 )
                    {
                        if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(2))+1,"/-1-\\");
                       Cualuno = 4;
                    }
                      else if(Cualuno == 4 )
                    {
                        if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(3))+1,"/-1-\\");
                       Cualuno = 5;
                    }
                      else if(Cualuno == 5 )
                    {
                        if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(4))+1,"/-1-\\");
                       Cualuno = 1;
                    }
                    //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-1-\\");
                    enemigos = enemigos +1;
               }
               else if(iterador==r2+1)
```

```
{
            if(Cualuno == 1 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(0))+1,"/-2-\\");
            Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*1)+1,"/-2-\\");
            Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(2))+1,"/-2-\\");
            Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(3))+1,"/-2-\\");
            Cualuno = 4;
        }
            else if(Cualuno == 5 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(4))+1,"/-2-\\");
            Cualuno = 1;
        }
            //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-2-\\");
        enemigos = enemigos +1;
}
else if(iterador==r3+1)
{
            if(Cualuno == 1 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(0))+1,"/-3-\\");
            Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*1)+1,"/-3-\\");
            Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(2))+1,"/-3-\\");
            Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(3))+1,"/-3-\\");
            Cualuno = 5;
        }
            else if(Cualuno == 5 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(4))+1,"/-3-\\");
            Cualuno = 1;
        }
        //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-3-\\");
        enemigos = enemigos +1;
}
else if(iterador==r4+1)
{
            if(Cualuno == 1 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(0))+1,"/-4-\\");
            Cualuno = 2;
        }
            else if(Cualuno == 2 )
```

```
                {
                    if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*1)+1,"/-4-\\");
                Cualuno = 3;
                }
                  else if(Cualuno == 3 )
                {
                    if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(2))+1,"/-4-\\");
                Cualuno = 4;
                }
                  else if(Cualuno == 4 )
                {
                    if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(3))+1,"/-4-\\");
                Cualuno = 5;
                }
                  else if(Cualuno == 5 )
                {
                    if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(4))+1,"/-4-\\");
                Cualuno = 1;
                }
                  //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-4-\\");
               enemigos = enemigos +1;
        }
        else
        {
            if(Cualuno ==1 )
            {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(0))+1,"\\-./");
                    //mvprintw(mitady+2,(espaciado*(iterador+5))+1,"\\-./");
                Cualuno = 2;
            }
              else if(Cualuno ==2 )
            {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*1)+1,"\\-./");
                    //mvprintw(mitady+1,(espaciado*iterador)+1,"\\-./");
                Cualuno = 3;
            }
              else if(Cualuno ==3 )
            {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(2))+1,"\\-./");
                    //mvprintw(mitady,(espaciado*(iterador-5))+1,"\\-./");
                Cualuno = 4;
            }
              else if(Cualuno == 4 )
            {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(3))+1,"\\-./");
                    //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-./");
                Cualuno = 5;
            }
              else if(Cualuno == 5 )
            {
                if(objetivos[iterador]==1) mvprintw(mitady-1,(espaciado*(4))+1,"\\-./");
                    //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-./");
                Cualuno = 1;
            }
        }
}

if(iterador < 15 )
{
        if(iterador==r1+1)
        {
              if(Cualuno == 1 )
            {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(0))+1,"/-1-\\");
                Cualuno = 2;
            }
              else if(Cualuno == 2 )
```

```
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*1)+1,"/-1-\\");
        Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(2))+1,"/-1-\\");
        Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(3))+1,"/-1-\\");
        Cualuno = 5;
        }
            else if(Cualuno == 5 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(4))+1,"/-1-\\");
        Cualuno = 1;
        }
        //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-1-\\");
        enemigos = enemigos +1;
}
else if(iterador==r2+1)
{
            if(Cualuno == 1 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(0))+1,"/-2-\\");
        Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*1)+1,"/-2-\\");
        Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(2))+1,"/-2-\\");
        Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(3))+1,"/-2-\\");
        Cualuno = 4;
        }
            else if(Cualuno == 5 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(4))+1,"/-2-\\");
        Cualuno = 1;
        }
            //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-2-\\");
        enemigos = enemigos +1;
}
else if(iterador==r3+1)
{
            if(Cualuno == 1 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(0))+1,"/-3-\\");
        Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*1)+1,"/-3-\\");
        Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(2))+1,"/-3-\\");
```

```
            Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(3))+1,"/-3-\\");
            Cualuno = 5;
        }
            else if(Cualuno == 5 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(4))+1,"/-3-\\");
            Cualuno = 1;
        }
        //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-3-\\");
        enemigos = enemigos +1;
}
else if(iterador==r4+1)
{
            if(Cualuno == 1 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(0))+1,"/-4-\\");
            Cualuno = 2;
        }
            else if(Cualuno == 2 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*1)+1,"/-4-\\");
            Cualuno = 3;
        }
            else if(Cualuno == 3 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(2))+1,"/-4-\\");
            Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(3))+1,"/-4-\\");
            Cualuno = 5;
        }
            else if(Cualuno == 5 )
        {
                if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(4))+1,"/-4-\\");
            Cualuno = 1;
        }
            //if(objetivos[iterador]==1) mvprintw(mitady,(espaciado*(iterador-5))+1,"/-4-\\");
        enemigos = enemigos +1;
}
else
{
        if(Cualuno ==1 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(0))+1,"\\-./");
                //mvprintw(mitady+2,(espaciado*(iterador+5))+1,"\\-./");
            Cualuno = 2;
        }
            else if(Cualuno ==2 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*1)+1,"\\-./");
                //mvprintw(mitady+1,(espaciado*iterador)+1,"\\-./");
            Cualuno = 3;
        }
            else if(Cualuno ==3 )
        {
            if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(2))+1,"\\-./");
                //mvprintw(mitady,(espaciado*(iterador-5))+1,"\\-./");
            Cualuno = 4;
        }
            else if(Cualuno == 4 )
        {
```

```
                        if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(3))+1,"\\-./");
                            //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-./");
                        Cualuno = 5;
                    }
                      else if(Cualuno == 5 )
                    {
                        if(objetivos[iterador]==1) mvprintw(mitady-2,(espaciado*(4))+1,"\\-./");
                            //mvprintw(mitady-1,(espaciado*(iterador-10))+1,"\\-./");
                        Cualuno = 1;
                    }
                }
            }

//*/

        aliens=aliens+1;
    }
    mvprintw(y, x, "<-ô->"); // Print our "ball" at the current xy position
            ch = getch();
            switch(ch)
            {
                case 'w':
            error_hilo= pthread_create (&idHilo, NULL, funcionThread, NULL);
            break;
                case 's':
            clear();
            printw("Finish Him!!!\n");
            char *puntos="Puntos: ";
            char ptos1[15];
            char ptos2[15];
            char *tiempo="Tiempo: ";
            char tiempov[15];
            sprintf(tiempov,"%s%d\n",tiempo,minutos);
            if(jugador==1)
            {
                sprintf(ptos1,"%s%d\n",puntos,ptos);
                sprintf(ptos2,"%s%d\n",puntos,ptorival);
            }else
            {
                sprintf(ptos2,"%s%d\n",puntos,ptos);
                sprintf(ptos1,"%s%d\n",puntos,ptorival);
            }
            if(jugador==1)
            {
                if(ptorival>ptos)
                {
                    printw("Ganador Invasor \n");
                    printw(ptos2);
                    printw("Perdedor Defensor \n");
                    printw(ptos1);
                }else
                {
                    printw("Ganador Defensor \n");
                    printw(ptos1);
                    printw("Perdedor Invasor \n");
                    printw(ptos2);
                }
            }else
            {
                if(ptorival>ptos)
                {
                    printw("Ganador Defensor \n");
                    printw(ptos1);
                    printw("Perdedor Invasor \n");
                    printw(ptos2);
                }
                else
```

```
            {
               printw("Ganador Invasor \n");
               printw(ptos2);
               printw("Perdedor Defensor \n");
               printw(ptos1);
            }
         }
         printw(tiempov);
         ch = getch();
              Reinicio();
         break;
              case 'a':
         izquierda();
         break;
              case 'd':
         derecha();
         break;
              case 'q':
         shmdt ((char *)mcompartida);
         shmctl (mem, IPC_RMID,(struct shmid_ds *)NULL); //finaliza el recurso de memoria
         unlink ("\tmp\acompartido");
         endwin();
         return;
         break;
           }
           refresh();
#ifdef MUTEX
                     /* Desbloquemos el mutex (EM) */
                     pthread_mutex_unlock (&mutexBuffer);
#endif
          control();                                 /* End curses mode                        */
}

void izquierda()
{
      if(x!=1)
      {
         mvprintw(y, x, "<-ô->"); // Print our "ball" at the current xy position
         refresh();
         usleep(DELAY);
         x--; // mover a la izquierda una posicion
      }
      return;
}
void derecha()
{
      if(x!=max_x-15)
      {
         mvprintw(y, x, "<-ô->"); // Print our "ball" at the current xy position
         refresh();
         usleep(DELAY);
         x++; // moder ala derecha una posicion
      }
      return;
}
void *funcionThread (void *parametro)
{
   int copiax=x;
   int copiay=y;
   int iterador=0;
   bool choco=false;
   getmaxyx(stdscr, max_y, max_x);
   while(copiay>0){
#ifdef MUTEX
                     /* Esperamos y bloqueamos el mutex (EM) */
                     pthread_mutex_lock (&mutexBuffer);
#endif
```

```
clear();
for(iterador=0;iterador<max_x-4;iterador++)
{
   mvprintw(2,iterador,"-");
   mvprintw(max_y-3,iterador,"-");
}
for(iterador=0;iterador<max_y;iterador++)
{
   mvprintw(iterador,max_x-15,"|");
}
mvprintw(y, x, "<-ô->"); // dibujamos la nave en la posicion acutal
mvprintw(copiay, copiax, "o"); // dibujamos el proyectil
refresh();
usleep(DELAY);
copiay--;               // hacemos q el proyectil avance
int espaciado=max_x/5;
int mitady=max_y/2;
if(copiay==mitady+2)
{
   for(iterador=0;iterador<5;iterador++)
   {
      if((copiax==(espaciado*iterador)+1) &&(objetivos[iterador]==1))
      {
         choco=true;
         objetivos[iterador]=0;
                  mcompartida[iterador+8]=0;
         ptos=ptos+10;
      }
   }
}
else if(copiay==mitady)
{
   for(iterador=0;iterador<5;iterador++)
   {
      if((copiax==(espaciado*iterador)+1) &&(objetivos[iterador+5]==1))
      {
         choco=true;
         objetivos[iterador+5]=0;
                  mcompartida[iterador+5+8]=0;
         ptos=ptos+15;
      }
   }
}
else if(copiay==mitady-2)
{
   for(iterador=0;iterador<5;iterador++)
   {
      if((copiax==(espaciado*iterador)+1) &&(objetivos[iterador+10]==1))
      {
         choco=true;
         objetivos[iterador+10]=0;
                  mcompartida[iterador+10+8]=0;
         ptos=ptos+10;
      }
   }
}
else if(copiay==3)
{
   if(copiax==max_x-15-rival)
   {
      vidarival2=vidarival2-1;
      if(vidarival2==0)
      {
         vidarival2=1;
      }
      choco=true;
      ptos=ptos+10;
```

```
            ledi=true;
         }
      }
      if(choco==true)
      {
         refresh();
         mvprintw(copiay, copiax, "X"); // Le pego
         usleep(DELAY);
         break;
      }
#ifdef MUTEX
   /* Desbloquemos el mutex (EM) */
   pthread_mutex_unlock (&mutexBuffer);
#endif
   }
}

void *escribir(void *parametro)
{
   while( true )
   {
   p1_entra = true;
   while( p2_entra )
   {
      if( turno == 2 )
      {
      p1_entra = false;
      while( turno == 2 ){}
      p1_entra = true;
      }
   }
   int iterador;
   if(jugador==1)
   {
      if(ledi==true)
      {
         mcompartida[6]=vidarival2;
         ledi=false;
      }
      mcompartida[2]=vida;
      mcompartida[1]=x;
      mcompartida[3]=ptos;
      for(iterador=0;iterador<15;iterador++)
      {
         mcompartida[iterador+8]=objetivos[iterador];
      }
   }
   else
   {
      if(ledi==true)
      {
         mcompartida[2]=vidarival2;
         ledi=false;
      }
      mcompartida[5]=x;
      mcompartida[6]=vida;
      mcompartida[7]=ptos;
      for(iterador=0;iterador<15;iterador++)
      {
         mcompartida[iterador+8]=objetivos[14-iterador];
      }
   }
   turno = 2;
   p1_entra = false;
   }
}
void *leer(void *parametro)
```

```
{
   while( true )
   {
   p2_entra = true;
   while( p1_entra )
   {
      if( turno == 1 )
      {
      p2_entra = false;
      while( turno == 1 ){}
      p2_entra = true;
      }
   }
   int iterador=0;
   if(jugador==1)
   {
      vidarival=mcompartida[6];
      ptorival=mcompartida[7];
      rival=mcompartida[5];
      for(iterador=0;iterador<15;iterador++)
      {
         objetivos[iterador]=mcompartida[iterador+8];
      }
   }
   else
   {
      vidarival=mcompartida[2];
      ptorival=mcompartida[3];
      rival=mcompartida[1];
      for(iterador=0;iterador<15;iterador++)
      {
         objetivos[iterador]=mcompartida[22-iterador];
      }
   }
   turno = 1;
   p2_entra = false;
   }
}

void *reloj(void *parametro)
{
   while(1)
   {
#ifdef MUTEX
                        /* Esperamos y bloqueamos el mutex (EM) */
                        pthread_mutex_lock (&mutexBuffer);
#endif
      minutos=minutos+1;

#ifdef MUTEX
                        /* Desbloquemos el mutex (EM) */
                        pthread_mutex_unlock (&mutexBuffer);
#endif
      sleep(1);
   }
}

void Reinicio()
{
   //pthread_exit(&reloj);
   turno,vida=0,ptos=0;
   jugador=0,aliens=0,rival=0,vidarival=5,ptorival=0;
   objetivos[20];
   minutos=0;
   segundos=0;
   vidarival2=5;
   ledi=false;
```

```
clear();
llave = ftok ("/bin/ls", 33);
identificadorSEM = semget (llave, 10, 0600 | IPC_CREAT);


printw("_____\n");
printw("\n\n\n\n");
printw("            ----------------------------------------\n");
printw("           |     Jugador 1 -> DEFENSOR <-          |\n");
printw("           |           (precione a)               |\n");
printw("            ----------------------------------------\n");
printw("\n\n\n\n");
printw("            ----------------------------------------\n");
printw("           |     Jugador 2 -> INVASOR <-           |\n");
printw("           |           (precione b)               |\n");
printw("            ----------------------------------------\n");
printw("\n\n\n\n");
printw("_____\n");

jugador=getch();
vida=5;
ptos=0;
mcompartida[8]=1;
mcompartida[9]=1;
mcompartida[10]=1;
mcompartida[11]=1;
mcompartida[12]=1;
mcompartida[13]=1;
mcompartida[14]=1;
mcompartida[15]=1;
mcompartida[16]=1;
mcompartida[17]=1;
mcompartida[18]=1;
mcompartida[19]=1;
mcompartida[20]=1;
mcompartida[21]=1;
mcompartida[22]=1;
mcompartida[23]=1;
mcompartida[24]=1;
mcompartida[25]=1;
mcompartida[26]=1;
mcompartida[27]=1;

//estado de las naves
objetivos[1] = 1;
objetivos[2] = 1;
objetivos[3] = 1;
objetivos[4] = 1;
objetivos[5] = 1;
objetivos[6] = 1;
objetivos[7] = 1;
objetivos[8] = 1;
objetivos[9] = 1;
objetivos[10] = 1;
objetivos[11] = 1;
objetivos[12] = 1;
objetivos[13] = 1;
objetivos[14] = 1;
objetivos[15] = 1;
objetivos[16] = 1;
objetivos[17] = 1;
objetivos[18] = 1;
objetivos[19] = 1;
objetivos[20] = 1;

turno=1;
```

```
   p1_entra=false;
   p2_entra=false;

   switch(jugador)
   {
          //asignacion de variabels para jugadores segun sea el caso
      case 'a':

            printw("          ------> LISTO!!! Esperando al despreciable invasor <------          \n");
            getch();
            semctl (identificadorSEM, 0, SETVAL, 0);

         procc.sem_num = 0;
         procc.sem_op = -1;
         procc.sem_flg = 0;

            semop (identificadorSEM, &procc, 1);

         jugador=1;
         mcompartida[0]=1;
         mcompartida[1]=x;
         mcompartida[2]=vida;
         mcompartida[3]=ptos;
         error_hilo= pthread_create (&rhilo, NULL, leer, NULL);
         error_hilo= pthread_create (&whilo, NULL, escribir, NULL);
         error_hilo= pthread_create (&chilo, NULL, reloj, NULL);
         break;

      default:

         procc2.sem_num = 0;
         procc2.sem_op = 1;
         procc2.sem_flg = 0;

         int i;
         for (i = 0; i<1; i++)
         {
            printf("AL ATAQUE!!! \n");
            semop (identificadorSEM, &procc2, 1);
            sleep (1);
         }

         jugador=2;
         mcompartida[4]=1;
         mcompartida[5]=x;
         mcompartida[6]=vida;
         mcompartida[7]=ptos;
         error_hilo= pthread_create (&rhilo, NULL, leer, NULL);
         error_hilo= pthread_create (&whilo, NULL, escribir, NULL);
         error_hilo= pthread_create (&chilo, NULL, reloj, NULL);
         break;
   }
#ifdef MUTEX
          /* Se inicia el mutex (EM) */
          pthread_mutex_init (&mutexBuffer, NULL);
#endif
   usleep(DELAY); // delay entre moviminetos

   control();
   shmdt ((char *)mcompartida);
   shmctl (mem, IPC_RMID,(struct shmid_ds *)NULL); //finaliza el recurso de memoria
   unlink ("\tmp\acompartido");
}

void *funcionThread2 (void *parametro)
{
   if((vidarival==1) || (vida==1) || (ptos>=100))
```

```
    {
        printw("Finish Him!!!\n");
            char *puntos="Puntos: ";
        char ptos1[15];
        char ptos2[15];
        char *tiempo="Tiempo: ";
        char tiempov[15];
        sprintf(tiempov,"%s%d\n",tiempo,minutos);
        if(jugador==1)
        {
           sprintf(ptos1,"%s%d\n",puntos,ptos);
           sprintf(ptos2,"%s%d\n",puntos,ptorival);
        }else
        {
           sprintf(ptos2,"%s%d\n",puntos,ptos);
           sprintf(ptos1,"%s%d\n",puntos,ptorival);
        }
        if(jugador==1)
        {
           if(ptorival>ptos)
           {
              printw("Ganador Invasor \n");
              printw(ptos2);
              printw("Perdedor Defensor \n");
              printw(ptos1);
           }else
           {
              printw("Ganador Defensor \n");
              printw(ptos1);
              printw("Perdedor Invasor \n");
              printw(ptos2);
           }
        }else
        {
           if(ptorival>ptos)
           {
              printw("Ganador Defensor \n");
              printw(ptos1);
              printw("Perdedor Invasor \n");
              printw(ptos2);
           }
           else
           {
              printw("Ganador Invasor \n");
              printw(ptos2);
              printw("Perdedor Defensor \n");
              printw(ptos1);
           }
        }
        printw(tiempov);
        printw("\n");
        getch();
            //goto INICIO;
            Reinicio();
    }
}
```

ESPECIFICACIONES

A continuación se especificaran ciertas restricciones o contenido relevante para el funcionamiento correcto de la aplicación entre las cuales están:

- Cada vez que el defensor logre darle a una nave invasora se le sumaran 10 o 15 puntos, según el tipo de nave

- El tiempo se debe mostrar en ambos jugadores, y debe ser continuo, no se debe reiniciar durante la ejecución del juego.

- Las naves invasoras se estarán moviendo de forma horizontal, y en ambas instancias del juego deben moverse igual. El movimiento de los invasores es automático, es decir no es controlado por el jugador invasor.

- Cada vez que el defensor sea golpeado por una bala enemiga perderá un punto de vida.

- Cada vez que el comandante invasor sea golpeado por una bala del defensor perderá un punto de vida.

- El juego termina cuando algunos de los dos jugadores tengan 0 vidas. O cuando el jugador defensor logre un score de 100 puntos.

SOPORTE

Autor	ALAN BAUTISTA
ID	2012-12487
Tel	4279 9021