

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Sistemas Operativos 1
Laboratorio

Manual Técnico
Practica 2
Space Invaders

Luis Aroldo Morales Noriega
201020944
20/09/2016

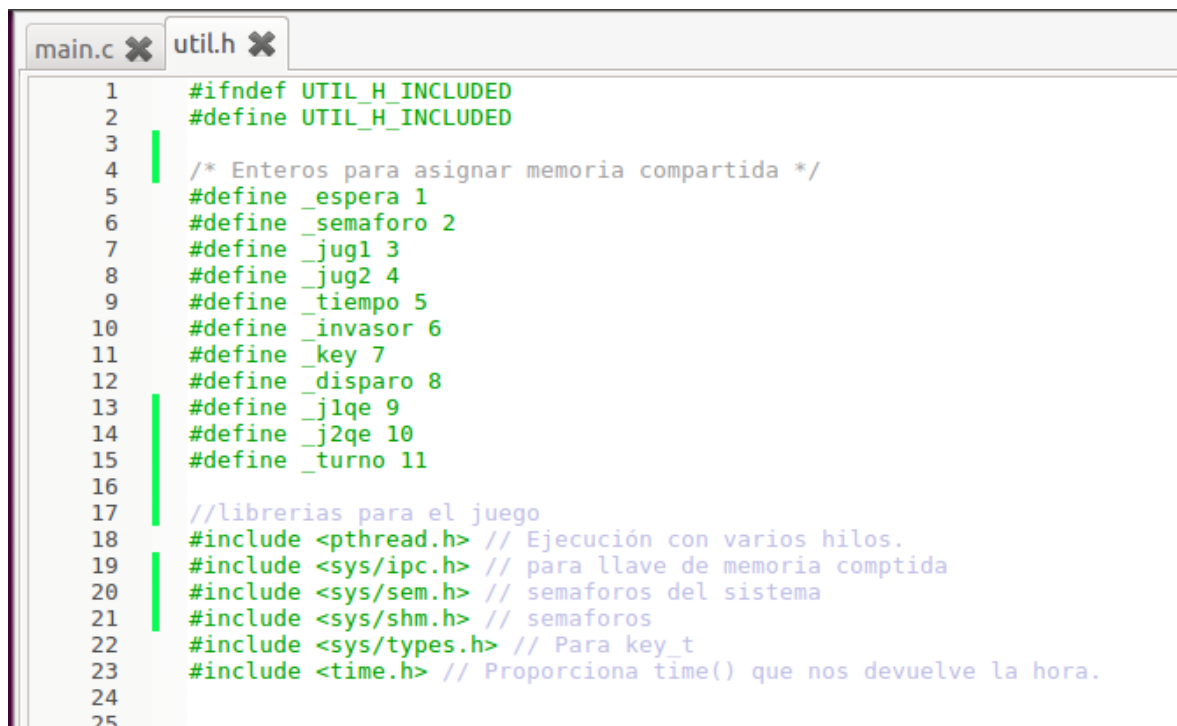
Space Invaders

La práctica realizada trata sobre el manejo de memoria compartida entre instancias de un juego llamado space invaders, el juego maneja varias pantallas que se mostraran más adelante, el funcionamiento de estas pantallas o el juego en si se aplican algoritmos para el manejo de nuestra memoria compartida como los son semáforos y dekker así es como funciona nuestro juego.

Para manejar el juego se utilizaron 2 archivos un llamado útil.h que maneja variables, librerías entre otras cosas que se utilizan para la lógica del juego, y el otro llamado main.c donde se encuentra el método principal main y los métodos y funciones que sirven para ejecutar nuestro juego a continuación se detallaran los contenidos de nuestros archivos así como la lógica del juego.

util.h:

Como podemos ver en la imagen se definen enteros para poder asignar memoria compartida como también se muestran algunas librerías que se utilizaran en el programa más la definición de su utilización.



```
1  #ifndef UTIL_H_INCLUDED
2  #define UTIL_H_INCLUDED
3
4  /* Enteros para asignar memoria compartida */
5  #define _espera 1
6  #define _semaforo 2
7  #define _jug1 3
8  #define _jug2 4
9  #define _tiempo 5
10 #define _invasor 6
11 #define _key 7
12 #define _disparo 8
13 #define _j1qe 9
14 #define _j2qe 10
15 #define _turno 11
16
17 //librerias para el juego
18 #include <pthread.h> // Ejecución con varios hilos.
19 #include <sys/ipc.h> // para llave de memoria compartida
20 #include <sys/sem.h> // semaforos del sistema
21 #include <sys/shm.h> // semaforos
22 #include <sys/types.h> // Para key_t
23 #include <time.h> // Proporciona time() que nos devuelve la hora.
24
25
```

En la imagen siguiente se muestran las estructuras que se utilizan para el juego y una breve descripción de estas, se utiliza Jugador, Invasor, List_invasores, Disparo, List_disparos.

```

main.c x util.h x
24
25
26 #ifdef _cplusplus
27 extern "C" {
28 #endif
29
30 //Estructuras
31 typedef struct {
32     int pos_x; // cooredenad inicial en x
33     int pos_y; // coordenada inicial en y
34     int puntos; // puntos obtenidos
35     int vida;
36 } Jugador; /* Representa a un jugador */
37
38 typedef struct {
39     int pos_x; // cooredenad inicial en x
40     int pos_y; // coordenada inicial en y
41     int tipo; // tipo de invasor
42     bool vivo; // nos muestra si esta vivo
43 } Invasor; /* Representa a un invasor */
44
45 typedef struct {
46     Invasor lista[20]; // lista de invasores para el juego
47 } List_invasores; /* Representa una lista de invasores */
48
49 typedef struct {
50     int pos_x; // posicion de disparo en x
51     int pos_y; // posicion de disparo en y
52 } Disparo; /* Representa un disparo */
53
54 typedef struct {
55     Disparo lista[1]; // lista de disparos
56 } List_disparos; /* Representa una lista de disparos */
57

```

También se muestran las variables que se utilizaran para la memoria compartida y una descripción de estas.

```

main.c x util.h x
59 //Memoria Compartida
60
61 // Bandera espera
62 key_t llave_espera; // Llave para la memoria compartida de la bandera de espera
63 int id_espera; // Identificador para la memoria de la bandera de espera
64 int *espera = NULL; //Apuntador a la zona de memoria de la bandera de espera
65
66 // Semaforo
67 key_t llave_semaforo; // Llave para el semaforo
68 int id_semaforo; // identificador para el array de semaforos
69
70 //judafor 1
71 key_t llave_jug1; // Llave para la memoria compartida del jugador 1
72 int id_jug1; // Identificador para la memoria del jugador 1
73 Jugador *jug1; // Apuntador a la zona de memoria del jugador 1
74
75 //judafor 2
76 key_t llave_jug2; // Llave para la memoria compartida del jugador 2
77 int id_jug2; // Identificador para la memoria del jugador 2
78 Jugador *jug2; // Apuntador a la zona de memoria del jugador 2
79
80 // Tiempo del juego
81 key_t llave_tiempo; // Llave para la memoria compartida del tiempo
82 int id_tiempo; // Identificador para la memoria del tiempo
83 int *tiempo; // Apuntador a la zona de memoria del tiempo
84
85 //invasores
86 key_t llave_invasor; // Llave para la memoria compartida de la(s) pelota(s)
87 int id_invasor; // Identificador para la memoria de la(s) pelota(s)
88 List_invasores *invasores; // Apuntador a la zona de memoria de la(s) pelota(s)
89

```

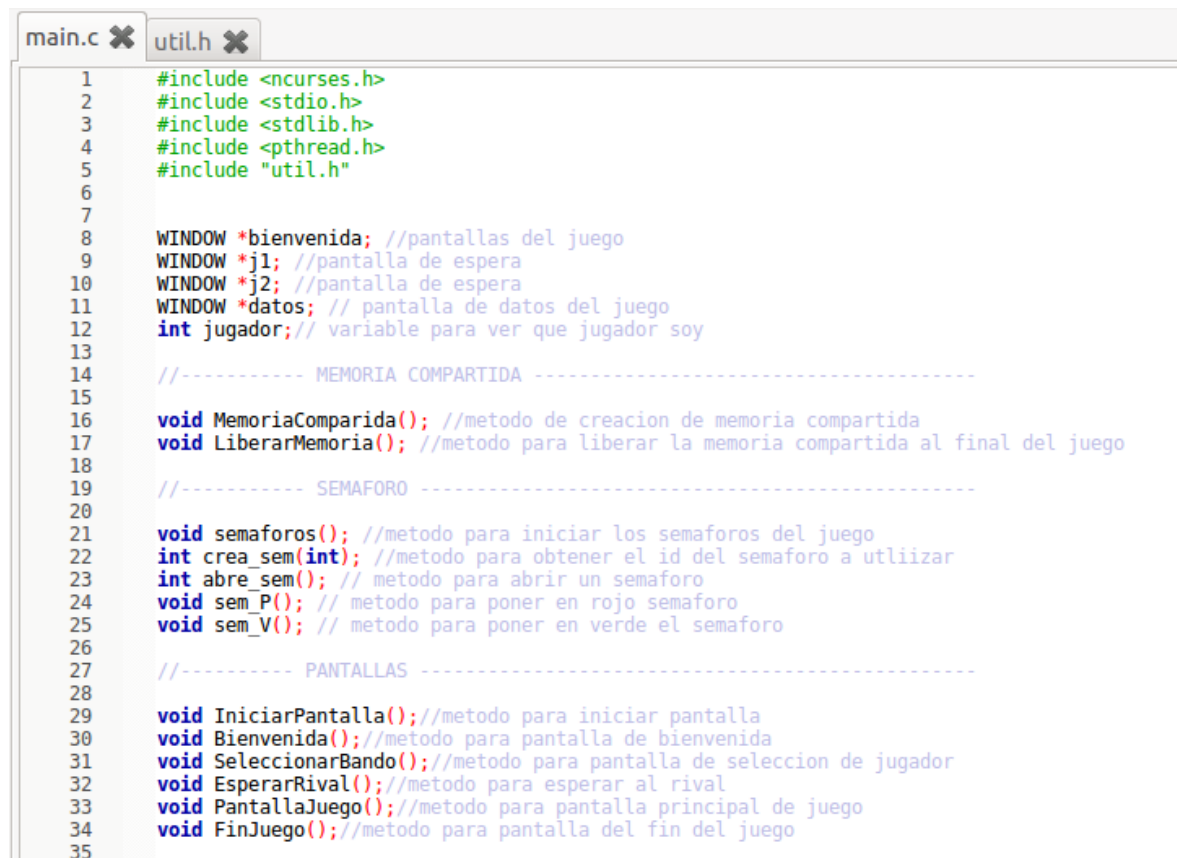
Por último, se muestran las variables que se utilizarán para los hilos que se ejecutaran en el juego.

```
main.c x util.h x
89
90 //disparos
91 key_t llave_disparo; // Llave para la memoria compartida de la(s) pelota(s)
92 int id_disparo; // Identificador para la memoria de la(s) pelota(s)
93 List_disparos *disparos; // Apuntador a la zona de memoria de la(s) pelota(s)
94
95 // Key del teclado
96 key_t llave_key; //Llave para la memoria compartida de key
97 int id_key; // Identificador para la memoria de key
98 char *key; // Apuntador a la zona de memoria de key
99
100
101 key_t llave_j1qe; // Llave para la memoria compartida de Judarol quiere entrar
102 int id_j1qe; // Identificador para la memoria de judorl quiere entrar
103 bool *j1_quiere_entrar; // Apuntador a la zona de memoria de la bandera para el proceso 1
104
105 key_t llave_j2qe; // Llave para la memoria compartida de jugador2 quiere entrar
106 int id_j2qe; // Identificador para la memoria de jugador2 quiere entrar
107 bool *j2_quiere_entrar; // Apuntador a la zona de memoria de la bandera para el proceso 2
108
109 key_t llave_turno; // Llave para la memoria compartida de turno
110 int id_turno; // Identificador para la memoria de turno
111 int *turno; // Apuntador a la zona de memoria de turno
112
113
114 //HILOS DEL JUEGO
115 pthread_t id_hilo_p1; // Identificador de p_jugador1
116
117 pthread_t id_hilo_p2; // Identificador de p_jugador2
118
119 pthread_t id_hilo_juego; // Identificador del hilo_juego
120 //hilo de tiempo
121 pthread_t id_hilo_tiempo; // Identificador del hilo del hilo_tiempo
122 //hilo de invasores
123 pthread_t id_hilo_invasores; // Identificador del hilo del hilo_tiempo
124
125
126 #ifdef cplusplus
```

main.c:

Este archivo contiene los métodos para ejecutar el juego y se describirán a continuación.

En la imagen siguiente se muestran librerías útiles para el juego como ncurses que se usa para graficar el juego como algunas variables que se utilizan para el juego, también se muestran los métodos con una breve descripción de su utilización en el programa.



```
1  #include <ncurses.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <pthread.h>
5  #include "util.h"
6
7
8  WINDOW *bienvenida; //pantallas del juego
9  WINDOW *j1; //pantalla de espera
10 WINDOW *j2; //pantalla de espera
11 WINDOW *datos; // pantalla de datos del juego
12 int jugador; // variable para ver que jugador soy
13
14 //----- MEMORIA COMPARTIDA -----
15
16 void MemoriaCompartida(); //metodo de creacion de memoria compartida
17 void LiberarMemoria(); //metodo para liberar la memoria compartida al final del juego
18
19 //----- SEMAFORO -----
20
21 void semaforos(); //metodo para iniciar los semaforos del juego
22 int crea_sem(int); //metodo para obtener el id del semaforo a utliizar
23 int abre_sem(); // metodo para abrir un semaforo
24 void sem_P(); // metodo para poner en rojo semaforo
25 void sem_V(); // metodo para poner en verde el semaforo
26
27 //----- PANTALLAS -----
28
29 void IniciarPantalla(); //metodo para iniciar pantalla
30 void Bienvenida(); //metodo para pantalla de bienvenida
31 void SeleccionarBando(); //metodo para pantalla de seleccion de jugador
32 void EsperarRival(); //metodo para esperar al rival
33 void PantallaJuego(); //metodo para pantalla principal de juego
34 void FinJuego(); //metodo para pantalla del fin del juego
35
```

Por ultimo mostramos las secciones donde se encuentran los métodos en el programa sabiendo por su descripción cual es el fin de cada uno.

```
main.c x util.h x
35
36 //----- JUEGO -----
37 void IniciarJugadores(); //metodo para iniciar las variables de los jugadores
38 void IniciarInvasores(); //metodo para iniciar las variables de los invasores
39 void CrearTablero(); //metodo para crear el tablero del juego
40 void DibujarInvasores(); //metodo para dibujar los invasores en el tablero
41 void Disparar(); //metodo para disparar en el juego
42
43 //----- HILOS -----
44 void *hilo_juego(); //metodo de hilo de juego
45 void *hilo_tiempo(); //metodo para el tiempo de juego
46 void *hilo_invasores(); //metodo para los invasores del juego
47
48 //----- DEKKER -----
49 void inicializar_dekker(); //metodo para iniciar las variables del metodo de dekker
50 void delay(); //metodo para el retardo de dekker
51 void region_critica(); //metodo para entrar a la region critica
52 void *p_j1(); //Hilo donde pide entrar a la region critica el jugador 1
53 void *p_j2(); //Hilo donde pide entrar a la region critica el jugador 2
54
55 //----- AREA MAIN -----
56 //
57
58
59 int main(){
60
61 //----- AREA MEMORIA COMPARTIDA -----
62 //
63
64 void MemoriaCompartida(){
65
66
67 void LiberarMemoria(){
68
69
70
71 //----- AREA PANTALLAS -----
72
```

```
main.c x util.h x
232
233 //----- AREA PANTALLAS -----
234 //
235
236
237 void IniciarPantalla(){
238
239 void Bienvenida(){
240
241 void SeleccionarBando(){
242
243 void EsperarRival(){
244
245 void PantallaJuego(){
246
247 void FinJuego() {
248
249 void IniciarJugadores(){
250
251 void IniciarInvasores(){
252
253 void MostrarDatos(){
254
255 void CrearTablero(){
256
257 void DibujarJugadores(){
258
259 void DibujarInvasores(){
260
261 void Disparar(){
262
```

```
711
712
713 //----- AREA SEMAFORO -----
714 //-----
715
716 void semaforos() {
731
732 int crea sem(int valor inicial){
738
739 int abre sem(){
742
743 void sem P() {
750
751 void sem V() {
757
758
759 //----- AREA HILOS -----
760 //-----
761
762 void* hilo juego() {
784
785 void* hilo tiempo() {
795
796 void* hilo invasores(){
815
816 //----- AREA DEKKER -----
817 //-----
818
819 void inicializar dekker() {
824
825 void delay() {
838
839 void region critica() {
873
874 void* p j1() {
899
900 void* p j2() {
925
```