
e-crops
etsi version

ÍNDICE:

1. Introducción

El trabajo realizado para la asignatura Sistemas Operativos se trata de un videojuego programado en C completamente desde cero. Es un juego tipo arcade que permite ver un primer acercamiento al funcionamiento de un sistema de gestión de plantas real de forma simple y entretenida. El videojuego consiste en mantener un huerto virtual, en el que se puede plantar 3 tipos de plantas. Cada planta tiene su nivel de vida, además de una serie de propiedades como son el nivel de agua o abono. El jugador tiene que encargarse de que ninguna planta se muera manteniendo su nivel de agua por encima de un umbral regando cada planta en el momento que cada jugador considere, igual con el abono, etc. La dificultad de éste juego la hemos conseguido limitando el número de acciones a 3 acciones por día.

2. Funcionamiento del videojuego

2.1.-Comienzo

El videojuego comienza con una pregunta que permite al jugador cargar una partida guardada anteriormente o empezar una nueva partida desde cero.

Puesto que el objetivo es explicar el funcionamiento del videojuego, suponemos que escogemos empezar una nueva partida. Después de contestar la primera cuestión, lo siguiente que se le pregunta al jugador es que plantas quiere plantar. Éste momento es importante puesto que es solo en éste momento donde el jugador puede escoger las plantas que va a intentar mantener toda la partida. Cabe destacar que el número máximo de plantas que se pueden plantar es de 5 plantas, hemos escogido este número para tratar de no ralentizar mucho el sistema operativo, pero éste número es fácilmente escalable.

Cada planta se crea con una serie de propiedades, que son las siguientes: tipo de planta, nivel de agua, nivel de abono, nivel de vida, estado (sana o enferma) y longevidad.

Se pueden plantar 3 tipos de plantas: Romero, Tulipán, Tomates. Al igual que el número máximo de plantas, el tipo de planta es fácilmente escalable e incluir más tipos de plantas en futuras actualizaciones.

El nivel de agua, nivel de abono y nivel de vida se inicializa en cada planta a su máximo qué es 99.

El estado de cada planta cuando se plantan es SANO.

La propiedad de longevidad hace referencia al número de días que cada planta ha sobrevivido, teniendo en cuenta esto, al inicio éste parámetro se encuentra a cero.

2.2.-Transcurso del tiempo

Una vez que elegimos las plantas que queremos tener en nuestro huerto, el siguiente paso que se hace automáticamente es plantarlas. A partir de éste momento tenemos que tener en cuenta de que el tiempo pasa independientemente de las acciones que hagamos, en segundo plano. Cada día en el huerto virtual dura un minuto en tiempo real.

Una vez que se han plantado todas las plantas que el jugador ha elegido, dará comienzo el primer día. Durante cada día el jugador tiene las siguientes opciones: **REGAR, ABONAR, FERTILIZAR, CONSULTAR, SALIR**.

Durante cada día, el jugador tiene 3 acciones como máximo a repartir entre todas las plantas que han sido plantadas (excluyendo aquellas de control como las consultas a las estadísticas de las plantas), teniendo la posibilidad de no hacer nada absolutamente.

En el momento en el que el jugador desea realizar una acción, debe escribirla y posteriormente se le preguntará sobre qué planta desea realizarla, a excepción de **SALIR**.

REGAR y **ABONAR** aumentan el nivel de agua y abono, respectivamente, en 10 puntos, y no se realizan instantáneamente, tardan 7 u 8 segundos en tiempo real, dato que hay que tener en cuenta puesto que sabemos que un día es un minuto en tiempo real, 7 u 8 segundos es una fracción del día considerable que no hay que despreciar. Esta implementación de tiempo artificial se debe a la intención de acercamiento a un sistema real, donde estas acciones, aún automatizadas, tienen un consumo de tiempo considerable.

FERTILIZAR es una acción que debemos hacer para curar una planta que ha enfermado, de tal forma que cambiaremos su estado a sana. Si realizamos ésta acción a una planta que se encuentra sana disminuiremos su vida en 30 puntos, cosa que no es aconsejable si queremos que nuestra planta sobreviva el número máximo de días.

CONSULTAR no disminuye el número de acciones, así que podemos consultar cada planta las veces que necesitemos. Ésta opción nos muestra por pantalla todas las estadísticas de la planta que elijamos en el momento que decidimos consultarla. El sentido de esta acción es poder representar en un sistema real, la posibilidad de acceder a toda la información de forma concisa, ordenada e inmediata, de forma que sea fácilmente trasladable.

Tras pasar 60 segundos da por finalizado el primer día, y la propiedad de longevidad de cada planta viva aumenta 1 punto. El agua y el abono

disminuyen en función del tiempo que pasa, es decir, de los días que transcurren. Como hemos mencionado anteriormente, cada planta al plantarse al comienzo del juego está sana, pero existe una probabilidad de que cada planta enferme al acabar un día. Ésta probabilidad se obtiene aleatoriamente utilizando el tiempo del sistema.

Cuando una planta tiene un nivel de agua y/o abono por debajo de 50, ésta planta pierde 25 puntos de vida.

También se puede perder 25 puntos de vida en una planta que haya enfermado y no se haya utilizado fertilizante.

2.3.-Final

Una planta muere cuando alcanza 0 puntos de vida. Y el juego se acaba cuando todas las plantas mueren, cuando no quede ninguna planta con vida.
Es importante comentar que tal y como funciona el videojuego, es obvio que para aumentar la dificultad basta con aumentar el número de plantas, consiguiendo el nivel máximo de dificultad cuando escojamos tener 5 plantas en nuestro huerto.

Cabe destacar que el videojuego cuenta con una opción de guardar la partida, de tal forma que cuando escribimos **SALIR**, nos pregunta si queremos **GUARDAR** o **SALIR**. Si elegimos **GUARDAR** podremos guardar la partida y en otro momento cargar la partida guardada y de esa forma poder seguir jugando con el mismo huerto que teníamos previamente.

3. Composición del videojuego

Para el correcto funcionamiento del videojuego, y con el objetivo de programar éste juego de la forma más clara, organizada y comprensible para cualquier persona, hemos decidido dividir el programa en 3 archivos principales: main.c, aux.h, function.c. Pasaremos a explicar cada uno de éstos archivos detalladamente a continuación.

3.1.-main.c

Éste archivo es el programa principal y consta de varias partes.
La primera parte del programa se encarga de preguntar al jugador si quiere empezar una nueva partida, o si desea cargar una partida empezada anteriormente. Si elegimos cargar una partida, pero en el directorio del juego no existe ninguna partida guardada se procederá a empezar una nueva partida. En el caso de crearse una nueva partida el siguiente paso sería preguntarle al jugador cuantas plantas y de qué tipo desea plantar en su

huerto virtual. Podemos observar el desarrollo de este procedimiento en la *Ilustración 1* e *Ilustración 2*.

```
imprimearte(1,0);
 printf("-------------------------------------------------------------------------------------\n");
 printf("-Bienvenido al huerto virtual. ¿Desea cargar una partida guardada o empezar una nueva? -\n");
 printf("-Seleccione una opción: |CARGAR| |EMPEZAR|                                          -\n");
 printf("-------------------------------------------------------------------------------------\n");

 fgets(input, sizeof(input), stdin);
 input[strlen(input)-1] = '\0';
 for (int k=0 ; k < (int) sizeof(input) ; k++) {
   input[k] = toupper(input[k]);
 }

 b = elegir_opcion(input);

 while (b==1) {
   printf("La opción indicada no se encuentra entre las opciones\n");
   printf("Por favor, escoja una de las opciones indicadas: ");

   if(fgets(input, sizeof(input), stdin)) {

     input[strlen(input)-1] = '\0';
     for (int k=0 ; k < (int) sizeof(input) ; k++) {
       input[k] = toupper(input[k]);
     }

     b = elegir_opcion(input);
   }
 }

 if(strcmp(input,"CARGAR")==0){
   u = cuentafich() - 3;
   numero_plantas = u;
   if ( u > 0 ) {
     for(j = 0; j < u; j++){
       planta_array[j] = leer(j+1);
     }
   } else {
     printf("-----------------------------------------\n");
     printf("-   NO EXISTE NINGUNA PARTIDA GUARDADA   -\n");
     printf("-SE PROCEDERA A EMPEZAR UNA NUEVA PARTIDA-\n");
     printf("-----------------------------------------\n");

     no_game_saved = true;
   }
 }
```

Ilustración 1

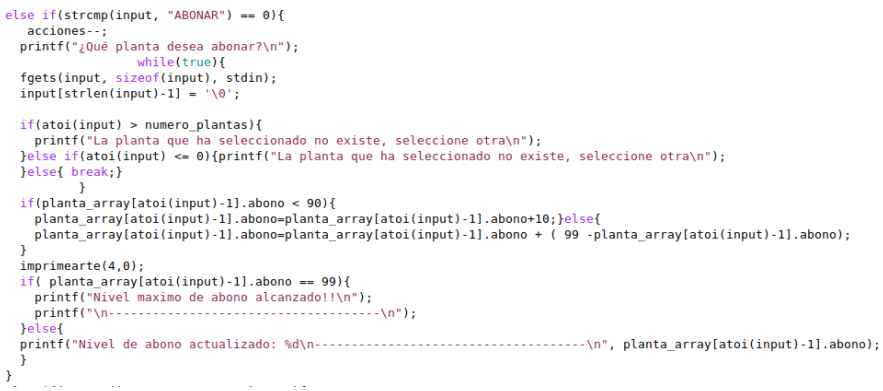
```
if (strcmp(input, "EMPEZAR") == 0 || no_game_saved == true) {

   u = cuentafich() - 3;
   borrar_all_fich(u);

   fichplantas = j;
   printf("Bienvenido al huerto virtual, actualmente hay %d plantas\n",j);
   printf("¿Desearia plantar alguna mas?\n");
   printf("Las plantas disponibles son:\n |ROMERO| |TULIPAN| |TOMATES|\n");

   while(j < MAX_THREADS && j != NO_MAS) {
     haplantado = true;
     printf("¿Cual de ellas desearia plantar?: ");

     if(fgets(input, sizeof(input), stdin)) {

       input[strlen(input)-1] = '\0';
       for (int k=0 ; k < (int) sizeof(input) ; k++) {
         input[k] = toupper(input[k]);
       }

       i = elegir_planta(input);

       while (i == ERROR_INPUT && j < 5) {
         printf("La planta indicada no se encuentra entre las opciones\n");
         printf("Por favor, escoja una de las plantas indicadas: ");

         if(fgets(input, sizeof(input), stdin)) {

           input[strlen(input)-1] = '\0';
           for (int k=0 ; k < (int) sizeof(input) ; k++) {
             input[k] = toupper(input[k]);
           }

           i = elegir_planta(input);
         }
       }

       plantas[j] = i;

       j++;
       numero_plantas = j;
```

Ilustración 2

La segunda parte se encarga de crear las plantas para ello recorremos un bucle donde crearemos un hilo, de forma paralela, por cada planta creada. Profundizaremos en éste concepto en el apartado 4, cuando expliquemos los contenidos de la asignatura reflejados en el videojuego.

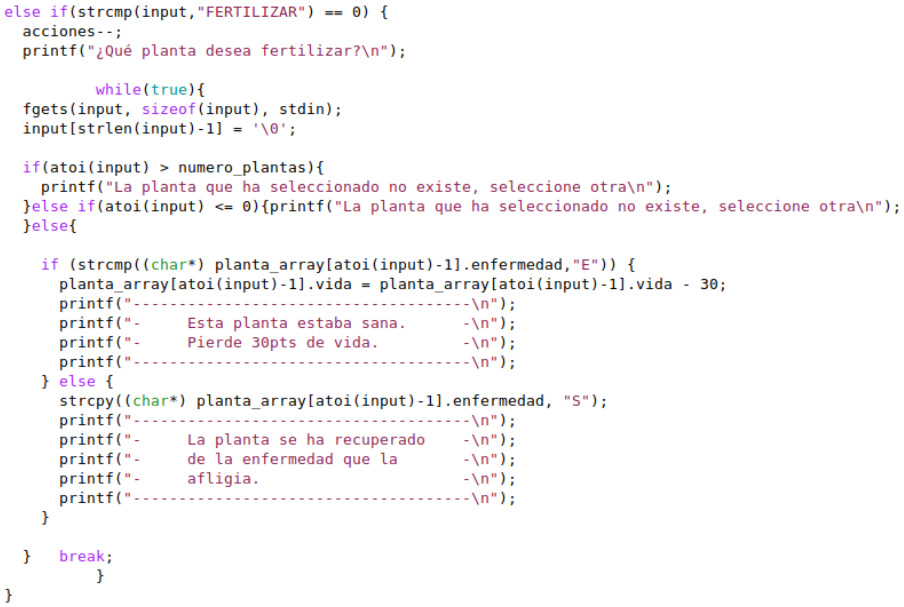
La tercera parte trata la parte más interesante del videojuego. En esta parte dará comienzo el primer día y transcurren todos los demás. En esta parte del fichero comenzamos iniciando un temporizador, a continuación, preguntamos al jugador qué acción desea realizar, y la capturamos. Dependiendo de lo que el jugador desee hacer, se hace una cosa u otra. En el caso de que el jugador quiera REGAR, ABONAR o FERTILIZAR se modificarán los valores de las plantas correspondientes. Si el jugador lo único que quiere hacer es consultar las estadísticas de una planta, éstas se muestran por pantalla. todas éstas acciones están condicionalmente protegidas para evitar sobrecargas los parámetros de las plantas (ej., aumentar el agua por encima de 99) o acceder a

plantas inexistentes, manteniendo así cierta seguridad en el acceso controlado. Y, por último, si el jugador quiere salir del juego, tendrá la opción de guardar la partida para continuarla más adelante o salir sin guardar. Cada acción se encarga de modificar el valor de alguna propiedad de nuestra planta, para modificar las estadísticas de cada planta accedemos a cada planta gracias a guardarlas como un array, de ésta forma planta_array [0] se corresponde con la primera planta que hemos plantado en nuestro huerto al comienzo del juego, planta_array [1] sería la segunda planta, y así sucesivamente en el caso de tener más de dos. Cada elemento de ese array es un struct con las estadísticas de cada planta. En las siguientes ilustraciones podemos observar éste procedimiento con más claridad:



```
else if(strcmp(input, "ABONAR") == 0){
   acciones--;
  printf("¿Qué planta desea abonar?\n");
                while(true){
  fgets(input, sizeof(input), stdin);
  input[strlen(input)-1] = '\0';

  if(atoi(input) > numero_plantas){
    printf("La planta que ha seleccionado no existe, seleccione otra\n");
  }else if(atoi(input) <= 0){printf("La planta que ha seleccionado no existe, seleccione otra\n");
  }else{ break;}
          }
  if(planta_array[atoi(input)-1].abono < 90){
    planta_array[atoi(input)-1].abono=planta_array[atoi(input)-1].abono+10;}else{
    planta_array[atoi(input)-1].abono=planta_array[atoi(input)-1].abono + ( 99 -planta_array[atoi(input)-1].abono);
  }
  imprimearte(4,0);
  if( planta_array[atoi(input)-1].abono == 99){
    printf("Nivel maximo de abono alcanzado!!\n");
    printf("\n-------------------------------------\n");
  }else{
  printf("Nivel de abono actualizado: %d\n-------------------------------------\n", planta_array[atoi(input)-1].abono);
  }
}
```

Ilustración 3.



```
else if(strcmp(input,"FERTILIZAR") == 0) {
  acciones--;
  printf("¿Qué planta desea fertilizar?\n");

          while(true){
  fgets(input, sizeof(input), stdin);
  input[strlen(input)-1] = '\0';

  if(atoi(input) > numero_plantas){
    printf("La planta que ha seleccionado no existe, seleccione otra\n");
  }else if(atoi(input) <= 0){printf("La planta que ha seleccionado no existe, seleccione otra\n");
  }else{

    if (strcmp((char*) planta_array[atoi(input)-1].enfermedad,"E")) {
      planta_array[atoi(input)-1].vida = planta_array[atoi(input)-1].vida - 30;
      printf("-------------------------------------\n");
      printf("-     Esta planta estaba sana.      -\n");
      printf("-     Pierde 30pts de vida.         -\n");
      printf("-------------------------------------\n");
    } else {
      strcpy((char*) planta_array[atoi(input)-1].enfermedad, "S");
      printf("-------------------------------------\n");
      printf("-     La planta se ha recuperado    -\n");
      printf("-     de la enfermedad que la       -\n");
      printf("-     afligia.                      -\n");
      printf("-------------------------------------\n");
    }

  }   break;
          }
}
```

Ilustración 4.

```
if (strcmp(input, "REGAR") == 0) {
   acciones--;
  printf("¿Qué planta desea regar?\n");
          while(true){
  fgets(input, sizeof(input), stdin);
  input[strlen(input)-1] = '\0';

  if(atoi(input) > numero_plantas){
    printf("La planta que ha seleccionado no existe, seleccione otra\n");
  }else if(atoi(input) <= 0){printf("La planta que ha seleccionado no existe, seleccione otra\n");
  }else{ break;}
          }
  if(planta_array[atoi(input)-1].agua < 90){
    planta_array[atoi(input)-1].agua=planta_array[atoi(input)-1].agua+10;}else{
    planta_array[atoi(input)-1].agua=planta_array[atoi(input)-1].agua + ( 99 -planta_array[atoi(input)-1].agua);
  }
  imprimearte(3,0);
  if( planta_array[atoi(input)-1].agua == 99){
    printf("Nivel maximo de agua alcanzado!!\n");
    printf("\n-------------------------------------\n");
  }else{
  printf("Nivel de agua actualizado: %d\n-------------------------------------\n ", planta_array[atoi(input)-1].agua);
  }
 }
```

Ilustración 5.

```
else if(strcmp(input, "CONSULTAR") == 0){
  printf("¿Qué planta desea consultar?\n");
  fgets(input, sizeof(input), stdin);
  input[strlen(input)-1] = '\0';
  if(atoi(input) <= 0){ printf("La planta no esta en la lista\n");}else if(atoi(input) > numero_plantas){
    printf("La planta no esta en la lista\n");
  }else{
    printf("--------------------------------------\n");
    printf("-           ESTADISTICAS            -\n");
    printf("--------------------------------------\n");
    printf("TIPO: %s\n",planta_array[atoi(input)-1].tipo);
    printf("AGUA: %d\n",planta_array[atoi(input)-1].agua);
    printf("ABONO: %d\n",planta_array[atoi(input)-1].abono);
    printf("VIDA: %d\n",planta_array[atoi(input)-1].vida);
    printf("LUZ: %d\n",planta_array[atoi(input)-1].luz);
    printf("ESTADO: %s\n",planta_array[atoi(input)-1].enfermedad);
    printf("LONGEVIDAD: %d\n",planta_array[atoi(input)-1].longevidad);
    printf("--------------------------------------\n");
  }
}
```

Ilustración 6.

3.2 aux.h

En este archivo están recogidos todos los includes de las librerías que son necesarias para el correcto funcionamiento del videojuego. Además de ésto, también recoge todas las constantes que se utilizan en el programa, como por ejemplo las estadísticas máximas y mínimas de cada planta, entre otras constantes. Por último, también se define los dos tipos de struct utilizados en el programa. En la *Ilustración 7* vemos todas las librerías y constantes utilizadas.

```c
#include <stdio.h>
 #include <unistd.h>
#include <pthread.h>
#include <sys/stat.h>           /* struct stat y fstat */
#include <sys/mman.h>           /* mmap */
#include <sys/types.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <stdbool.h>
#include <sys/times.h>
#include <sys/wait.h>
#include <sys/msg.h>
#include <dirent.h>
#include <math.h>

#define MAX_THREADS 5
#define DURACION_DIA 5

#define ROMERO 10
#define TULIPAN 20
#define TOMATES 30

#define NO_MAS 40
#define SI_MAS 45

#define ERROR_INPUT 50

#define VIDA_MAX 99
#define VIDA_MIN 00

#define MAX_AGUA 99
#define MIN_AGUA 00

#define MAX_ABONO 99
#define MIN_ABONO 00

#define MAX_LUZ 99
#define MIN_LUZ 00

#define SANA S
#define ENFERMA E

#define BAJA_STAT 49
#define PIERDE_VIDA -10
```

Ilustración 7

3.3 function.c

Éste archivo es el encargado de implementar todas las funciones que se utilizan en el main.c. De esta forma, todo el código queda más organizado y claro. También conseguimos ahorrar líneas de código, puesto que cada función está definida e implementada y cada vez que utilizamos una función de este archivo solo tenemos que llamar a la función correspondiente con los argumentos que correspondan.

Además, se incluye en éste fichero la función void *planta(void *arg) que es la función que se ejecuta cuándo se crean los hilos de cada planta. Ésta función se encarga de iniciar cada planta con sus propiedades correspondientes al comienzo del juego, además de crear un fichero dónde se van a guardar estas estadísticas. Observamos esta función en la *Ilustración 8*.

```
void *planta(void *arg) {


  int tipo_planta =  *(int *)arg;
  int numero_planta = thr_number;
  thr_number++;

  if (tipo_planta == ROMERO || tipo_planta == TULIPAN || tipo_planta == TOMATES) {
    if(tipo_planta == ROMERO){strcpy((char *)planta_array[numero_planta].tipo, "romero");}
     if(tipo_planta == TOMATES){strcpy((char *)planta_array[numero_planta].tipo, "tomate");}
      if(tipo_planta == TULIPAN){strcpy((char *)planta_array[numero_planta].tipo, "tulipan");}
     planta_array[numero_planta].agua = MAX_AGUA;
    planta_array[numero_planta].vida = VIDA_MAX;
    planta_array[numero_planta].abono = MAX_ABONO;
    planta_array[numero_planta].luz = MAX_LUZ;
    strcpy((char *)planta_array[numero_planta].enfermedad, "S");

    if (strcmp((char *)planta_array[numero_planta].tipo,"romero") == 0  ) {
        printf("--------------------------\n");
        printf("Se ha plantado: ROMERO\n");
      creaFICH(thr_number, "romero");

    } else if (strcmp((char *)planta_array[numero_planta].tipo,"tulipan") == 0  ) {
        printf("--------------------------\n");
        printf("Se ha plantado: TULIPAN\n");;
      creaFICH(thr_number, "tulipan");

    } else if(strcmp((char *)planta_array[numero_planta].tipo,"tomate") == 0  ) {
        printf("--------------------------\n");
        printf("Se han plantado: TOMATES\n");
       creaFICH(thr_number, "tomate");

    } else {
      perror("ERROR: PLANTA NO EXISTENTE\n");
    }


  } else {
    perror("Planta no reconocida\n");
  }

  pthread_exit(NULL);

}
```

Ilustración 8

4. Contenidos de la asignatura reflejados en el videojuego

4.1. -Hilos

En nuestro proyecto los hilos se crean una vez escogidas las plantas que vamos a plantar, para la implementación de los hilos se incluye la librería <pthread.h> así como la orden -lpthread a la hora de compilar. Para la creación de dichos hilos inicializamos las variables tal y como se muestra en la *Ilustración 9*, los hilos serán dependientes los unos de los otros y no finalizarán hasta que se haga en el main.c de pthread_join, tal y como se ve en la *Ilustración 10.*

```
pthread_attr_t attr;
pthread_t thid[MAX_THREADS];

/*Iniciamos los atributos y los marcamos como dependientes*/
pthread_attr_init(&attr);
pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
```

Ilustración 9.

```
/* Con este bucle vemos que plantas se han escogido y creamos los hilos para
 cada una de ellas */
thr_number = fichplantas;

for (int k=0; k < (int) ((sizeof(plantas)/sizeof(plantas[0]))-1) ; k++) {

  if (plantas[k] == ROMERO || plantas[k] == TULIPAN || plantas[k] == TOMATES) {
    switch(plantas[k]) {
    case ROMERO:

      pthread_create(&thid[k], &attr, planta, &plantas[k]);
      sleep(1);
      break;
    case TULIPAN:

      pthread_create(&thid[k], &attr, planta, &plantas[k]);
              sleep(1);
      break;
    case TOMATES:

      pthread_create(&thid[k], &attr, planta, &plantas[k]);
              sleep(1);
      break;
    }
  }
}


for (int k = 0; k < MAX_THREADS ; k++) {
  if (plantas[k] == ROMERO || plantas[k] == TULIPAN || plantas[k] == TOMATES) {
    if((pthread_join(thid[k], NULL)) == 0) {
    } else {
      printf("Error terminando los hilos\n");
    }
  }
}
```

Ilustración 10.

En cada uno de los hilos se creen (1 por cada planta que se desea plantar se ejecutará el proceso ligero denominado "***planta***", al cual se le pasará como argumento el tipo de planta escogida y dicho proceso se encargará de crear un array global con la estructura de cada planta que contiene las estadísticas de cada una, además se encargará de crear los ficheros donde se guardaran las estadísticas de las plantas, dichos ficheros son creados a partir de otros ficheros en la carpeta **./Trabajo Final/Data** y recibirán el nombre del número de hilo que lo haya creado. La función "***planta***" se muestra en la *Ilustración 11.*

```
void *planta(void *arg) {


  int tipo_planta =  *(int *)arg;
  int numero_planta = thr_number;
  thr_number++;

  if (tipo_planta == ROMERO || tipo_planta == TULIPAN || tipo_planta == TOMATES) {
    if(tipo_planta == ROMERO){strcpy((char *)planta_array[numero_planta].tipo, "romero");}
     if(tipo_planta == TOMATES){strcpy((char *)planta_array[numero_planta].tipo, "tomate");}
      if(tipo_planta == TULIPAN){strcpy((char *)planta_array[numero_planta].tipo, "tulipan");}
     planta_array[numero_planta].agua = MAX_AGUA;
    planta_array[numero_planta].vida = VIDA_MAX;
    planta_array[numero_planta].abono = MAX_ABONO;
    planta_array[numero_planta].luz = MAX_LUZ;
    strcpy((char *)planta_array[numero_planta].enfermedad, "S");

    if (strcmp((char *)planta_array[numero_planta].tipo,"romero") == 0  ) {
        printf("--------------------------\n");
        printf("Se ha plantado: ROMERO\n");
      creaFICH(thr_number, "romero");

    } else if (strcmp((char *)planta_array[numero_planta].tipo,"tulipan") == 0  ) {
        printf("--------------------------\n");
        printf("Se ha plantado: TULIPAN\n");;
      creaFICH(thr_number, "tulipan");

    } else if(strcmp((char *)planta_array[numero_planta].tipo,"tomate") == 0  ) {
        printf("--------------------------\n");
        printf("Se han plantado: TOMATES\n");
       creaFICH(thr_number, "tomate");

    } else {
      perror("ERROR: PLANTA NO EXISTENTE\n");
    }


  } else {
    perror("Planta no reconocida\n");
  }

  pthread_exit(NULL);

}
```

Ilustración 11.

4.2. El reloj

Hemos utilizado los conocimientos adquiridos en la práctica 8 de la asignatura sobre el funcionamiento del reloj y las interrupciones del sistema para aplicarlas a nuestro videojuego. Al principio de cada día, iniciamos el reloj que finaliza cuando acaba el día, de esta forma podemos obtener el tiempo transcurrido, es decir, el número de días que han pasado. En la *Ilustración 12* se observan las líneas de código correspondientes al inicio y terminación del reloj.

```
/*Damos comienzo al dia con un bucle que depende del elapsed_time*/
time(&start_day); //cada vez que empieza un dia hay resetear la variable start_day
elapsed_time = 0;
time(&end_day);

elapsed_time = difftime(end_day, start_day);

elapsed_time = (elapsed_time + tiempo_acumulado) / 60;
tiempo_acumulado = elapsed_time;
```

Ilustración 12

4.3. Directorios y mapeado de archivos

Las funciones de directorio y mapeado de archivos son una parte fundamental del trabajo, ya que son las funciones para poder crear los archivos donde se guardarán las plantas, así como funciones de carga que se encargan de leer los archivos para su posterior uso.

Hemos utilizado los conocimientos de la práctica 9 en las siguientes situaciones; en la creación de un nuevo fichero en el caso de que se empiece una nueva partida (función void **creaFICH**(int name, char *tipo)), en caso contrario para la apertura del fichero guardado, leer estos datos guardados y continuar con la partida (función struct estadisticas **leer**(int num)) y, por último, para escribir los datos de nuestra partida y tener la posibilidad de guardarla (función void **escribir**(int num, struct estadisticas stat)). En todo momento se utiliza el servicio de cambio de directorio chdir("./Data") de ésta manera sabemos que la carpeta dónde se guardan los ficheros es Data. La utilización de los mismos es de vital importancia en el proyecto, debido a que el mapeado de memoria permite el escalado de la cantidad de plantas o los datos de estas de forma que permita trabajar de forma crítica aprovechando la alta velocidad de utilización de esta(alrededor del doble de rápido según lo probado respecto a los métodos tradicionales de FILE(fopen,write, etc)).

En la *Ilustración 13* se muestran algunos de estos ejemplos utilizados en las funciones nombradas anteriormente, dónde proyectamos los ficheros y posteriormente eliminamos la proyección de estos ficheros, también consultamos los atributos de los ficheros con fstat (f, &bstat) visto en la práctica.

```
faux = open(tipo,  O_RDONLY,0666);

 if ((f = open(str2 , O_CREAT|O_RDWR,0666)) < 0)
  {
    perror ("No puede crearse el fichero\n");           //Apertura del fichero
  }
 else if (fstat (faux, &bstat) < 0)           //Obtención de estadísticas
  {
    perror ("Error en fstat");
  } else if (ftruncate(f, bstat.st_size)<0){
   perror("Error en truncate");
 } else if ((mod=(char *)mmap((caddr_t) 0, bstat.st_size, PROT_READ,
                              MAP_SHARED, faux, 0)) == MAP_FAILED)
   {
     perror("Error en la proyeccion del fichero origen\n");
   }
 else
   if ((mapeo  = (char *) mmap (NULL, bstat.st_size, PROT_READ|PROT_WRITE, MAP_SHARED, f,0)) == MAP_FAILED)     //Mapeo del fichero
     {
       perror ("Error en la proyeccion del fichero\n");
     }
   else
     {
       close(faux);
       close (f);

       p = mapeo;
       s = mod;

       /* Bucle de copia */
       for (i=0; i<bstat.st_size; i++)
         *p++= *s++;
       /* Se eliminan las proyecciones */

       munmap (mod, bstat.st_size);
       munmap (mapeo, bstat.st_size);
     }
```

Ilustración 13

Disclaimer: los conceptos artísticos del proyecto (mayoría de ASCII art y logotipo) han sido obtenidos en Internet. Propiedad de Nastya Yu (logotipo sin letras) y ascii.uk.co.uk (ASCII art). La fuente utilizada para el logotipo es de uso abierto cedida por Chucklefish Games