```c
/*
 * Archivo: parser.c
 *
 * Descripcion: Recibe una entrada por la linea de comandos y retorna una
 * estructura con la informacion de las opciones activas de mytar y sus
 * argumentos.
 *
 * Autores:
 *  Carlos Alejandro Sivira Munoz     15-11377
 *  Cesar Alfonso Rosario Escobar   15-11295
 * Grupo: 18
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include "parser.h"
/*
 *  fArgument
 * -------------------
 *  Recibe una cadena de caracteres y verifica si esta es el nombre del archivo
 * .mytar.
 *
 *  string: Cadena de caracteres a verificar.
 *
 *  Retorno: Resultado de la llamada strcmp. -1 en caso de error.
 */
int fArgument(char *string){
  const char c[2] = ".";
  char *token;
  token = strtok(string, c);
  token = strtok(NULL, c);
  /*No es posible conseguir el token*/
  if (!token){
    return -1;
  }
  /*Verifica si .mytar esta contenido en la cadena*/
  return strcmp(token, "mytar");
}
/*
 *  setOptions
 * -------------------
 *  Guarda las opciones activas de mytar en la estructura mytar_instructions.
 *
 *  instructions: Estructura de opciones mytar.
 *  c: Caracter actual a ser verificado y/o guardado.
 *
 *  Retorno: 0 si la ejecucion fue correcta. -1 en caso de error.
 */
int setOptions(mytar_instructions *instructions, char c){
  /*Verifica si c es una opcion valida*/
  switch (c)
  {
    case 'c':
      instructions->mytar_options[C] = 1;
      break;
    case 't':
      instructions->mytar_options[T] = 1;
      break;
```

```
    case 'x':
      instructions->mytar_options[X] = 1;
      break;
    case 'o':
      instructions->mytar_options[O] = 1;
      break;
    case 'n':
      instructions->mytar_options[N] = 1;
      break;
    case 'z':
      instructions->mytar_options[Z] = 1;
      break;
    case 'y':
      instructions->mytar_options[Y] = 1;
      break;
    case 'v':
      instructions->mytar_options[V] = 1;
      break;
    case 'f':
      instructions->mytar_options[F] = 1;
      break;
    case 's':
      instructions->mytar_options[S] = 1;
      break;
    default:
      /*Arroja error por caracter desconocido*/
      printf("Undefined mytar option %c\n", c);
      return -1;
      break;
  }
  return 0;
}
/*
 *  parse
 * -------------------
 *  Recibe la entrada del comando mytar y retorna la estructura de opciones.
 *
 *  num_arguments: Numero de argumentos suministrados.
 *  arguments: Argumentos dados para almacenar en la estructura.
 *
 *  Retorno: Apuntador a la estructura de opciones mytar_instructions.
 */
mytar_instructions* parse(int num_arguments, char **arguments){
  int i;
  /*El numero de argumentos de la opcion f*/
  int f_count = 1;
  /*Se inicializa la estructura de opciones mytar*/
  mytar_instructions *instructions = instructionsInit();
  /*Verifica si la inicializacion fue correcta*/
  if (!instructions){
    perror("Error: ");
    return NULL;
  }
  /*Recorre todos los argumentos dados*/
  for (i = 1; i < num_arguments; i++){
    int j = 0;
    /*Salva el ultimo caracter visitado*/
    char last;
    /*Cadena que almacena temporalmente los argumentos*/
    char *arg = malloc(MAXLEN);
```

```
121     strcpy(arg, arguments[i]);
122     /*Verifica si el argumento actual es una serie opciones*/
123     if (arg[j] == '-' || arg[j] == '-'){
124       j++;
125       /*Recorre cada caracter de la serie de comandos*/
126       while(j < strlen(arg) && arg[j]){
127         /*Verifica si son comando validos*/
128         if (setOptions(instructions ,arg[j]) != 0){
129           return NULL;
130         }
131         last = arg[j];
132         j++;
133       }
134     /*Salva los argumentos de cada opcion de forma ordenada*/
135     } else {
136       char *aux_string = malloc(MAXLEN);
137       char *aux_val = malloc(MAXLEN);
138       int fldes;
139       /*Salva el argumento correspondiente a la ultima opcion visitada*/
140       switch (last)
141       {
142         case 'o':
143           strcpy(instructions->output_directory, (arg));
144           break;
145         case 'z':
146           instructions->encryption_offset = -atoi(arg);
147           instructions->is_encripted = 1;
148           break;
149         case 'y':
150           instructions->encryption_offset = atoi(arg);
151           break;
152         case 'v':
153           /*Verifica si existe el archivo destino para verboso*/
154           fldes = open(arg, O_WRONLY | O_TRUNC | O_CREAT);
155           if (fldes < 0){
156             perror("open");
157             return NULL;
158           }
159           instructions->output_verbose = fldes;
160           break;
161         case 'f':
162           /*Salva todos los argumentos de f*/
163           strcpy(aux_string, arg);
164           strcpy(aux_val, arg);
165           /*Verifica si el argumentos es el archivo .mytar*/
166           if (fArgument(aux_string) == 0){
167             instructions->creation_directory[0] = aux_val;
168           } else {
169             instructions->creation_directory[f_count] = aux_val;
170             f_count++;
171           }
172           break;
173         case 's':
174           strcpy(instructions->file_extraction, arg);
175           break;
176         default:
177           /*Arroja error por caracter desconocido*/
178           printf("Undefined mytar option %c\n", last);
179           return NULL;
180           break;
```

```
      }
      free(aux_string);
    }
    free(arg);
  }
  instructions->num_args = f_count;
  return instructions;
}
/*
 *  instructionsInit
 * ------------------
 *  Inicializa la estructura de opciones mytar_instructions.
 *
 *  Retorno: Apuntador a la estructura de opciones mytar_instructions. Si
 *  existe un error, returna NULL.
 */
mytar_instructions* instructionsInit(){
  int i;
  /*Nueva estructura de opciones mytar*/
  mytar_instructions *new_instructions = malloc(sizeof(mytar_instructions));
  /*Verifica si la solicitud de memoria fue correcta*/
  if (!new_instructions){
    perror("Error: ");
    return NULL;
  }
  /*Inicializa el arreglo de opciones*/
  for (i = 0; i < NUMOPTIONS; i++) {
    new_instructions->mytar_options[i] = 0;
  }
  /*Inicializa los argumentos de las opciones con sus valores por defecto*/
  new_instructions->encryption_offset = 0;
  new_instructions->output_verbose = 1;
  new_instructions->creation_directory[0] = "file.mytar";
  new_instructions->is_encripted = 0;
  new_instructions->num_args = 0;
  strcpy(new_instructions->output_directory, ".");
  strcpy(new_instructions->file_extraction, "");

  return new_instructions;
}
/*
 *  verboseMode
 * ------------------
 *  Agrega informacion adicional sobre la ejecucion mytar y sus opciones.
 *
 *  instructions: Estructura que contiene la informacion de las opciones de
 *          mytar.
 *  filePath: La ruta del archivo actual.
 *
 *  Retorno: Vacio.
 */
void verboseMode(mytar_instructions instructions, char *filePath){
  /*Cadenas de caracteres auxiliares*/
  char *output = malloc(MAXLEN);
  char *string = malloc(MAXLEN);
  memset(output, '\0', MAXLEN);
  memset(string, '\0', MAXLEN);

  /*Verifica si el modo c esta activo*/
  if(instructions.mytar_options[C]){
```

```
241     strcat(output, "Adding ");
242     strcat(output, filePath);
243     strcat(output, " to ");
244     strcat(output, instructions.creation_directory[0]);
245     /*Verifica si la opcion z esta activa*/
246     if(instructions.mytar_options[Z]){
247       sprintf(string,"%i", instructions.encryption_offset);
248       strcat(output, " encrypting with ");
249       strcat(output, string);
250     }
251     /*Verifica si la opcion n esta activa*/
252     if(instructions.mytar_options[N]){
253       strcat(output, " ignoring non regular file or directory");
254     }
255   }
256
257   /*Verifica si el modo t esta activo*/
258   if(instructions.mytar_options[T]){
259     strcat(output, "Showing ");
260     strcat(output, filePath);
261     strcat(output, " from ");
262     strcat(output, instructions.creation_directory[0]);
263     /*Verifica si la opcion y esta activa*/
264     if(instructions.mytar_options[Y]){
265       sprintf(string,"%i", instructions.encryption_offset);
266       strcat(output, " decrypting with ");
267       strcat(output, string);
268     }
269     /*Verifica si la opcion z esta activa*/
270     if(instructions.mytar_options[Z]){
271       sprintf(string,"%i", instructions.encryption_offset);
272       strcat(output, " encrypting with ");
273       strcat(output, string);
274     }
275     /*Verifica si la opcion n esta activa*/
276     if(instructions.mytar_options[N]){
277       strcat(output, " ignoring non regular file or directory");
278     }
279   }
280
281   /*Verifica si el modo x esta activo*/
282   if(instructions.mytar_options[X]){
283     strcat(output, "Extracting ");
284     /*Verifica si la opcion s esta activa*/
285     if(instructions.mytar_options[S]){
286       strcat(output, "specifically ");
287     }
288     strcat(output, filePath);
289     strcat(output, " from ");
290     strcat(output, instructions.creation_directory[0]);
291     strcat(output, " in ");
292     strcat(output, instructions.output_directory);
293     /*Verifica si la opcion y esta activa*/
294     if(instructions.mytar_options[Y]){
295       sprintf(string,"%i", instructions.encryption_offset);
296       strcat(output, " decrypting with ");
297       strcat(output, string);
298     }
299     /*Verifica si la opcion n esta activa*/
300     if(instructions.mytar_options[N]){
```

```
      strcat(output, " ignoring non regular file or directory");
    }
  }
  /*Muesta en la salida espeficada la descripcion*/
  strcat(output, "\n");
    write(instructions.output_verbose, output, strlen(output));
}

/*
 *  verifyOptions
 * -------------------
 *  Verifica si la entrada de opciones es correcta.
 *
 *  instructions: Estructura que contiene la informacion de las opciones de
 *          mytar.
 *
 *  Retorno: Retorna 0 si las isntrucciones estan corectas. -1 en caso
 *  contrario.
 */
int verifyOptions(mytar_instructions instructions){
  /*Verifica si se intenta encriptar y desencriptar al mismo tiempo*/
  if (instructions.mytar_options[Z] && instructions.mytar_options[Y]){
    printf("You can't use -z and -y at the same time\n");
    return -1;
  }

  /*Verifica si se intenta crear un .mytar y desencriptar*/
  if (instructions.mytar_options[C] && instructions.mytar_options[Y]){
    printf("You can't use -c and -y at the same time\n");
    return -1;
  }

  /*Verifica si se intenta crear un .mytar y dar directorio de salida*/
  if (instructions.mytar_options[C] && instructions.mytar_options[O]){
    printf("You can't use -c and -o at the same time\n");
    return -1;
  }

  /*Verifica si se intenta mostrar un .mytar y dar directorio de salida*/
  if (instructions.mytar_options[T] && instructions.mytar_options[O]){
    printf("You can't use -t and -o at the same time\n");
    return -1;
  }

  /*Verifica si se intenta extraer un .mytar y encriptar*/
  if (instructions.mytar_options[X] && instructions.mytar_options[Z]){
    printf("You can't use -x and -z at the same time\n");
    return -1;
  }

  /*Verifica si se intenta mostrar un .mytar sin argumento f*/
  if (instructions.mytar_options[T] && !instructions.mytar_options[F]){
    printf("You can't use -t without -f argument\n");
    return -1;
  }

  /*Verifica si se intenta extraer un .mytar sin argumento f*/
  if (instructions.mytar_options[X] && !instructions.mytar_options[F]){
    printf("You can't use -x without -f argument\n");
    return -1;
```

```
  }

  /*Verifica si se intenta extraer un archivo encriptado sin desencriptarlo*/
  if (instructions.mytar_options[X] && instructions.is_encripted){
    if(!instructions.mytar_options[Y]){
      printf("You can't use -x without -y. The file is encrypted\n");
      return -1;
    }
  }

  /*Verifica si se intenta crear y mostrar al mismo tiempo un .mytar*/
  if (instructions.mytar_options[X] && instructions.mytar_options[T]){
    printf("You can't use -c and -t at the same time\n");
    return -1;
  }

  /*Verifica si se intenta crear y extraer al mismo tiempo un .mytar*/
  if (instructions.mytar_options[C] && instructions.mytar_options[X]){
    printf("You can't use -c and -x at the same time\n");
    return -1;
  }

  /*Verifica si se intenta extraer y mostrar al mismo tiempo un .mytar*/
  if (instructions.mytar_options[X] && instructions.mytar_options[T]){
    printf("You can't use -x and -t at the same time\n");
    return -1;
  }

  return 0;
}

```