Code source projet de programmation système 2016/2017

Groupe IN501A2
Gautier DELACOUR
Bérangère SUBERVIE
Maxime VALLERON

6 janvier 2017

Table des matières

Chapitre 1

mapio.c

```c
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include "map.h"
#include "error.h"

#ifdef PADAWAN

void map_new(unsigned width, unsigned height){
  map_allocate(width, height);

  // Sol

  for(int x = 0; x < width; x++)
    map_set(x, height - 1, 0);

  // Mur

  for(int y = 0; y < height - 1; y++){
    map_set(0, y, 1);
    map_set(width - 1, y, 1);
  }

  map_object_begin(6);

  // Ajout des differents objets utilises

  map_object_add("images/ground.png", 1, MAP_OBJECT_SOLID);
  map_object_add("images/wall.png", 1, MAP_OBJECT_SOLID);
  map_object_add("images/grass.png", 1, MAP_OBJECT_SEMI_SOLID);
  map_object_add("images/marble.png", 1, MAP_OBJECT_SOLID | MAP_OBJECT_DESTRUCTIBLE);
  map_object_add("images/flower.png", 1, MAP_OBJECT_AIR);
  map_object_add("images/coin.png", 20, MAP_OBJECT_AIR | MAP_OBJECT_COLLECTIBLE);

  map_object_end();
}



void map_save(char *filename){
  int valeur, err, tmp;
  int height = map_height();
  int width = map_width();
  int frame, type1, type2, type3, type4;
  char name[100];
  int nb_object = map_objects();

  // Ouverture du fichier de sauvgarde

  int output = open("maps/saved.map", O_TRUNC | O_WRONLY | O_CREAT,0666);

  if(output == -1){
    fprintf(stderr, "Probleme dans maps/saved.map: %s\n", filename);
    exit(1);
  }

  lseek(output, 0, SEEK_SET);

  // Sauvgarde de la taille

  write(output, &width, sizeof(int));
  write(output, &height, sizeof(int));

  // Sauvgarde des objets

  write(output, &nb_object, sizeof(int));

  for(int i = 0; i < nb_object; i++){
    frame = map_get_frames(i);
    strcpy(name, map_get_name(i));
    tmp = strlen(name);
    write(output, &tmp, sizeof(int));
    for(int j = 0; j < tmp; j++){
      write(output, &name[j], sizeof(int));
    }
    type1 = map_get_solidity(i);
    type2 = map_is_destructible(i);
    type3 = map_is_collectible(i);
    type4 = map_is_generator(i);
    write(output,&frame,sizeof(int));
    write(output,&type1,sizeof(int));
    write(output,&type2,sizeof(int));
    write(output,&type3,sizeof(int));
    write(output,&type4,sizeof(int));
  }

  // Lecture de map

  for(int y = 0; y < height; y++){
    for(int x = 0; x < width; x++){
      valeur = map_get(x, y);
      err = write(output, &valeur, sizeof(int));
      if(err == -1){
      fprintf(stderr, "Probleme de sauvegarde: %s\n", filename);
      exit(1);
      }
    }
  }

  close(output);
}



void map_load(char *filename){
  int err, fd;
  char n, type1, type2, type3, type4;
  char *adress = malloc(sizeof(char));

  // Ouverture du fichier de sauvgarde

  fd = open(filename, O_RDONLY, 0666);

  if(fd == -1){
    fprintf(stderr, "Desole: %s n'existe pas\n", filename);
    exit(1);
  }

  err = lseek(fd, 0, SEEK_SET);

  if(err == -1){
    fprintf(stderr, "Probleme de format: %s\n", filename);
    exit(1);
  }

  // Chargement de la taille

  err = read(fd, &n, sizeof(int));

  if(err == -1){
    fprintf(stderr, "Probleme de format: %s\n", filename);
    exit(1);
  }

  unsigned width = n;
  err = read(fd, &n, sizeof(int));

  if(err == -1){
    fprintf(stderr, "Probleme de format: %s\n", filename);
    exit(1);
  }

  unsigned height = n;
  map_allocate(width, height);
  err = read(fd, &n, sizeof(int));

  if(err == -1){
    fprintf (stderr, "Probleme de format: %s\n", filename);
    exit(1);
  }

  int nb_object = n;

  // Chargement des objets

  map_object_begin(nb_object);
  char *tmp1 = malloc(sizeof(char) * 20);

  for(int i = 0; i < nb_object; i++){
    err = read(fd, &adress[0], sizeof(int));
    if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
    }
    tmp1 = realloc(tmp1, ((int)adress[0]) * sizeof(char));
    for(int j = 0; j < (int)adress[0]; j++){
      err = read(fd, &tmp1[j], sizeof(int));
      if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
      }
    }
    err = read(fd, &n, sizeof(int));
    if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
    }
    err = read(fd, &type1, sizeof(int));
    if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
    }
    err = read(fd, &type2, sizeof(int));
    if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
    }
    err = read(fd, &type3, sizeof(int));
    if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
    }
    err = read(fd, &type4, sizeof(int));
    if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
    }
    if(type2){
      map_object_add(tmp1, n, type1 | MAP_OBJECT_DESTRUCTIBLE);
    }
    else if(type3){
      map_object_add(tmp1, n, type1 | MAP_OBJECT_COLLECTIBLE);
    }
    else if(type4){
      map_object_add(tmp1, n, type1 | MAP_OBJECT_GENERATOR);
    }
    else{
      map_object_add(tmp1, n, type1);
    }
    for(int k = 0; k < adress[0]; k++){
      strcpy(tmp1 + k, "");
    }
  }

  map_object_end ();
  free(tmp1);
  free(adress);

  // Ecriture de map

  for (int y = 0; y < height; y++){
    for (int x = 0; x < width; x++){
      err = read(fd, &n, sizeof(int));
      if(err == -1){
      fprintf(stderr, "Probleme de format: %s\n", filename);
      exit(1);
      }
      map_set(x, y, n);
    }
  }

  close(fd);
}

#endif
```

Chapitre 2

maputil.c

```c
#define _XOPEN_SOURCE 500

#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include "map.h"
#include "error.h"

#ifdef PADAWAN

#define NB_OPTIONS 8

static void usage(char *arg){
  fprintf(stderr, "%s <file> <option>\n", arg);
  exit(EXIT_FAILURE);
}



// Alloue un tableau contenant toutes les options possibles de maputil

void optionsAlloc(char *t[]){
  t[0] = malloc(10 * sizeof(char));
  strcpy(t[0], "--getwidth");
  t[1] = malloc(11 * sizeof(char));
  strcpy(t[1], "--getheight");
  t[2] = malloc(12 * sizeof(char));
  strcpy(t[2], "--getobjects");
  t[3] = malloc(9 * sizeof(char));
  strcpy(t[3], "--getinfo");
  t[4] = malloc(10 * sizeof(char));
  strcpy(t[4], "--setwidth");
  t[5] = malloc(11 * sizeof(char));
  strcpy(t[5], "--setheight");
  t[6] = malloc(12 * sizeof(char));
  strcpy(t[6], "--setobjects");
  t[7] = malloc(14 * sizeof(char));
  strcpy(t[7], "--pruneobjects");
}



// Libere le tableau contenant toutes les options de maputil

void optionsFree(char *t[]){
  for(int i = 0; i < NB_OPTIONS; i++)
```

```c
    free(t[i]);
}



// Renvoie la largeur d'une carte

int getWidth(int Fd){
  int i;
  int e = lseek(Fd, 0, SEEK_SET);

  if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

  e = read(Fd, &i, sizeof(int));
  if(e == -1){
    perror("read");
    exit(EXIT_FAILURE);
  }

  return i;
}



// Renvoie la hauteur d'une carte

int getHeight(int Fd){
  int i;
  int e = lseek(Fd, sizeof(int), SEEK_SET);

  if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

  e = read(Fd, &i, sizeof(int));

  if(e == -1){
    perror("read");
    exit(EXIT_FAILURE);
  }

  return i;
}



// Renvoie le nombre d'objets d'une carte

int getObjects(int Fd){
  int i;
  int e = lseek(Fd, 2 * sizeof(int), SEEK_SET);

  if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

  e = read(Fd, &i, sizeof(int));

  if(e == -1){
    perror("read");
    exit(EXIT_FAILURE);
  }

  return i;
}



// Change l'ancienne largeur par la nouvelle w

void setWidth(int Fd, int w){
  if(16 <= w && w <= 1024){
    int oldW = getWidth(Fd);
    if(oldW == w)
    return;
    int h = getHeight(Fd);
    int j = 0;
    int lenName, k;
    int t[h * w];
    int nbObjects = getObjects(Fd);
    int nbCaracObj = 5;
    int e = lseek(Fd, 0, SEEK_SET);
    if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }
    e = write(Fd, &w, sizeof(int));
    if(e == -1){
    perror("write");
    exit(EXIT_FAILURE);
  }

    // Place le curseur au debut de la liste des objets

    e = lseek(Fd, 3 * sizeof(int), SEEK_SET);
    if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

    // Recupere la taille du nom de chaque fichier dans lenName, et decale de cette taille
        plus le nombre de caracteristiques des objets

    for(int i = 0; i < nbObjects; i++){
    e = read(Fd, &lenName, sizeof(int));
    if(e == -1){
      perror("read");
      exit(EXIT_FAILURE);
    }
    e = lseek(Fd, (lenName + nbCaracObj) * sizeof(int), SEEK_CUR);
    if(e == -1){
      perror("lseek");
      exit(EXIT_FAILURE);
    }
  }

    // Recopie des elements communs aux nouvelles et anciennes tailles

    for(int y = 0; y < h; y++){
      for(int x=0;x<oldW;x++){
        if( x < w){
          e = read(Fd, &(t[j]), sizeof(int));
          if(e == -1){
            perror("read");
            exit(EXIT_FAILURE);
          }
          j++;
        }
      }

      // Si la taille est retrecie

      if(oldW > w)
      lseek(Fd, (oldW - w) * sizeof(int), SEEK_CUR);

      // Si la taille est augmentee

      if(oldW < w){
        for(k = j; k < j + (w - oldW); k++){
          t[k] = MAP_OBJECT_NONE;
        }
        j = k;
      }
    }

    // Place le curseur au debut de la liste des objets

    e = lseek(Fd, 3 * sizeof(int), SEEK_SET);
    if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

    // Recupere la taille du nom de chaque fichier dans lenName, et decale de cette taille
       plus le nombre de caracteristiques des objets

    for(int i = 0; i < nbObjects; i++){
    e = read(Fd, &lenName, sizeof(int));
    if(e == -1){
      perror("read");
      exit(EXIT_FAILURE);
    }
    e = lseek(Fd, (lenName + nbCaracObj) * sizeof(int), SEEK_CUR);
    if(e==-1){
      perror("lseek");
      exit(EXIT_FAILURE);
    }
  }

    // Ecrit les elements du tableau dans le fichier Fd

    for(int y = 0; y < h; y++){
      for(int x = 0; x < w; x++){
        e = write(Fd, t + (y * w + x), sizeof(int));
        if(e == -1){
          perror("write");
          exit(EXIT_FAILURE);
        }
      }
    }

    // Tronque le fichier s'il est plus petit

    if(oldW > w){
      int offset = lseek(Fd, oldW - w, SEEK_END);
      ftruncate(Fd, offset);
    }
  }
  else
    printf("Nouvelle␣largeur␣non␣autorisee!\n");
}



// Change l'ancienne hauteur par la nouvelle h

void setHeight(int Fd, int h){
  if(12 <= h && h <= 20){
    int oldH = getHeight(Fd);
    if(oldH == h)
    return;
    int w = getWidth(Fd);
    int j = 0;
    int lenName, k;
    int t[h * w];
    int nbObjects = getObjects(Fd);
    int nbCaracObj = 5;
    int e = lseek(Fd, sizeof(int), SEEK_SET);
    if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }
    e = write(Fd, &h, sizeof(int));
    if(e==-1){
    perror("write");
    exit(EXIT_FAILURE);
  }

    // Place le curseur au debut de la liste des objets

    e = lseek(Fd, 3 * sizeof(int), SEEK_SET);
    if(e==-1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

    // Recupere la taille du nom de chaque fichier dans lenName, et decale de cette taille
        plus le nombre de caracteristiques des objets

    for(int i = 0; i < nbObjects; i++){
    e = read(Fd, &lenName, sizeof(int));
    if(e == -1){
      perror("read");
      exit(EXIT_FAILURE);
    }
    e = lseek(Fd, (lenName + nbCaracObj) * sizeof(int), SEEK_CUR);
    if(e == -1){
      perror("lseek");
      exit(EXIT_FAILURE);
    }
  }

    // Recopie des elements communs aux nouvelles et anciennes tailles

    int tmpH = 0;
    if( oldH > h)
    tmpH = oldH + 1;
    else
    tmpH = h;
    for(int y = 0; y < tmpH; y++){

      // Si la taille est augmentee

      if(y < h - oldH){
        for(k = 0; k < w; k++){
          t[k + y * w] = MAP_OBJECT_NONE;
        }
        j = k + y * w;
      }
      else if( y > oldH - h){
        for(int x = 0; x < w; x++){
          e = read(Fd, &(t[j]), sizeof(int));
          if(e == -1){
            perror("read");
            exit(EXIT_FAILURE);
          }
          j++;
          }
        }

        // Si la taille est retrecie

        else if(y < oldH - h){
        lseek(Fd, w * sizeof(int), SEEK_CUR);
      }
    }

      // Place le curseur au debut de la liste des objets

      e = lseek(Fd, 3 * sizeof(int), SEEK_SET);
      if(e == -1){
      perror("lseek");
      exit(EXIT_FAILURE);
    }

      // Recupere la taille du nom de chaque fichier dans lenName, et decale de cette taille
         plus le nombre de caracteristiques des objets

      for(int i = 0; i < nbObjects; i++){
      e = read(Fd, &lenName, sizeof(int));
      if(e == -1){
        perror("read");
        exit(EXIT_FAILURE);
      }
      e = lseek(Fd, (lenName + nbCaracObj) * sizeof(int), SEEK_CUR);
      if(e == -1){
        perror("lseek");
        exit(EXIT_FAILURE);
      }
    }

      // Ecrit les elements du tableau dans le fichier Fd

      for(int y = 0; y < h; y++){
        for(int x = 0; x < w; x++){
          e = write(Fd, t + (y * w + x), sizeof(int));
          if(e == -1){
            perror("write");
            exit(EXIT_FAILURE);
          }
        }
      }

      // Tronque le fichier s'il est plus petit

      if(oldH > h){
        int offset = lseek(Fd, (oldH - h), SEEK_END);
        ftruncate(Fd, offset);
      }
    }
    else
      printf("Nouvelle␣hauteur␣non␣autorisee!\n");
}



void setObjects(int Fd, char *name, int frame, int solid, int destructible, int collectible,
     int generator){
  int nbObjects = getObjects(Fd);
  int w = getWidth(Fd);
  int h = getHeight(Fd);
  int t[h * w];

  // Place le curseur au debut de la liste des objets

```

```c
  int e = lseek(Fd, 3 * sizeof(int), SEEK_SET);

  if(e==-1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

  char *adress = malloc(sizeof(char));
  char *tmp1 = malloc(sizeof(char) * 20);

  for(int i = 0; i < nbObjects; i++){
    e = read(Fd, &adress[0], sizeof(int));
    if(e == -1){
    fprintf (stderr, "Probleme␣de␣format␣du␣fichier␣de␣sauvegarde\n");
    exit(1);
  }
    tmp1 = realloc(tmp1, ((int)adress[0]) * sizeof(char));
    for(int j = 0; j < ((int)adress[0]); j++){
    e = read(Fd, &tmp1[j], sizeof(int));
    if(e == -1){
      fprintf (stderr, "Probleme␣de␣format␣du␣fichier␣de␣sauvegarde\n");
      exit(1);
    }
  }
    if(strcmp(tmp1, name) == 0){
    e = write(Fd, &frame, sizeof(int));
    if(e == -1){
      perror("write");
      exit(1);
    }
    e = write(Fd, &solid, sizeof(int));
    if(e == -1){
      perror("write");
      exit(1);
    }
    e = write(Fd, &destructible, sizeof(int));
    if(e == -1){
      perror("write");
      exit(1);
    }
    e = write(Fd, &collectible, sizeof(int));
    if(e == -1){
      perror("write");
      exit(1);
    }
    e = write(Fd, &generator, sizeof(int));
    if(e == -1){
      perror("write");
      exit(1);
    }
    free(tmp1);
    free(adress);
    return;
    }
    else{
    e = lseek(Fd, 5 * sizeof(int), SEEK_CUR);
    if(e == -1){
      perror("lseek");
      exit(EXIT_FAILURE);
    }
  }
    for(int k = 0; k < adress[0]; k++)
    strcpy(tmp1 + k, "");
  }

  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
      e = read(Fd, t + (y * w + x), sizeof(int));
      if(e == -1){
        perror("read");
        exit(EXIT_FAILURE);
      }
    }
  }

  e = lseek (Fd, 2 * sizeof(int), SEEK_SET);

  if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

  nbObjects++;
  e = write(Fd, &nbObjects, sizeof(int));

  if(e == -1){
    perror("write");
    exit(1);
  }

  int lenName;
  e = lseek(Fd, 3 * sizeof(int), SEEK_SET);

  if(e == -1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

  for(int i = 0; i < nbObjects - 1; i++){
    e = read(Fd, &lenName, sizeof(int));
    if(e == -1){
    perror("read");
    exit(EXIT_FAILURE);
  }
    e = lseek(Fd, (lenName + 5) * sizeof(int), SEEK_CUR);
    if(e == -1){
      perror("lseek");
      exit(EXIT_FAILURE);
    }
  }

  lenName = strlen(name);
  e = write(Fd, &lenName, sizeof(int));

  if(e == -1){
    perror("write");
    exit(1);
  }

  for(int j = 0; j < lenName; j++){
    e = write(Fd, &name[j], sizeof(int));
    if(e == -1){
      perror("write");
      exit(1);
    }
  }

  if(e == -1){
    perror("write");
    exit(1);
  }

  e = write(Fd, &frame, sizeof(int));

  if(e == -1){
    perror("write");
    exit(1);
  }

  e = write(Fd, &solid, sizeof(int));

  if(e == -1){
    perror("write");
    exit(1);
  }

  e = write(Fd, &destructible, sizeof(int));

  if(e == -1){
    perror("write");
    exit(1);
  }

  e = write(Fd, &collectible, sizeof(int));

  if(e == -1){
    perror("write");
    exit(1);
  }

  e = write(Fd, &generator, sizeof(int));

  if(e == -1){
    perror("write");
    exit(1);
  }

  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
      e = write(Fd, t + (y * w + x), sizeof(int));
      if(e == -1){
        perror("write");
        exit(EXIT_FAILURE);
      }
    }
  }

  free(tmp1);
  free(adress);
}



void pruneOjects(int Fd){
  int nbObjects = getObjects(Fd);
  int nbObjectsTmp = 0;
  int t[nbObjects];
  int tlenName[nbObjects];
  char **tname = malloc(sizeof(char *) * nbObjects);

  for(int k = 0; k < nbObjects; k++)
    tname[k]= malloc(sizeof(char) * 20);

  int tframe[nbObjects];
  int tsolid[nbObjects];
  int tdestructible[nbObjects];
  int tcollectible[nbObjects];
  int tgenerator[nbObjects];

  for(int i = 0; i < nbObjects; i++)
    t[i] = 0;

  int w = getWidth(Fd);
  int h = getHeight(Fd);
  int tab[w * h];
  int object;
  int e = lseek(Fd, 3 * sizeof(int), SEEK_SET);

  if(e==-1){
    perror("lseek");
    exit(EXIT_FAILURE);
  }

  for(int i = 0; i < nbObjects; i++){
    e = read(Fd, &tlenName[i], sizeof(int));
    if(e == -1){
    perror("read1");
    exit(EXIT_FAILURE);
  }
    for(int j = 0; j < tlenName[i]; j++){
    e = read(Fd, tname[i] + j, sizeof(int));
    if(e == -1){
      perror("read2");
      exit(EXIT_FAILURE);
    }
  }
    e = read(Fd, &tframe[i], sizeof(int));
    if(e == -1){
    perror("read3");
    exit(EXIT_FAILURE);
  }
    e = read(Fd, &tsolid[i], sizeof(int));
    if(e == -1){
    perror("read4");
    exit(EXIT_FAILURE);
  }
    e = read(Fd, &tdestructible[i], sizeof(int));
    if(e == -1){
    perror("read5");
    exit(EXIT_FAILURE);
  }
    e = read(Fd, &tcollectible[i], sizeof(int));
    if(e == -1){
    perror("read6");
    exit(EXIT_FAILURE);
  }
    e = read(Fd, &tgenerator[i], sizeof(int));
    if(e == -1){
      perror("read7");
      exit(EXIT_FAILURE);
    }
  }

  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
    e = read(Fd, &object, sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
      tab[x + y * w] = object;
      if(object != -1 && t[object] != 1){
        t[object] = 1;
        nbObjectsTmp++;
      }
    }
  }

  e = lseek(Fd, 2 * sizeof(int), SEEK_SET);

  if(e == -1){
    perror("lseek");
```

```c
    exit(EXIT_FAILURE);
  }

  e = write(Fd, &nbObjectsTmp, sizeof(int));

  if(e == -1){
    perror("write");
    exit(EXIT_FAILURE);
  }

  int tmp = 0;

  for(int i = 0; i < nbObjects; i++){
    if(t[i]){
    e = write(Fd, &tlenName[i], sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
    for(int j = 0; j < tlenName[i]; j++){
      e = write(Fd, &tname[i][j], sizeof(int));
        if(e == -1){
          perror("write");
          exit(EXIT_FAILURE);
        }
      }
    e = write(Fd, &tframe[i], sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
    e = write(Fd, &tsolid[i], sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
    e = write(Fd, &tdestructible[i], sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
    e = write(Fd, &tcollectible[i], sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
    e = write(Fd, &tgenerator[i], sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
      t[i] = tmp;
      tmp++;
    }
  }

  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
    if(tab[y * w + x] == -1)
      e = write(Fd, &tab[y * w + x], sizeof(int));
    else
      e = write(Fd , &t[tab[y * w + x]],sizeof(int));
    if(e == -1){
      perror("write");
      exit(EXIT_FAILURE);
    }
  }
    printf("\n");
  }
}



// Teste la correspondance entre l'option demandee et les options existantes, et appelle une
    fonction correspondante si elle existe

int traitementOption(char *optTab[], int Fd, char *argv[], int k, int argc){
  char *option = argv[k];
  char *arg;

  if(k < argc)
    arg = argv[k + 1];

  int n = 0;
  printf("\nOption choisie : \t");

  // getwidth

  if(!strcmp(option, optTab[0])){
    printf("%s\t", optTab[0]);
    printf("%d\n", getWidth(Fd));
    n = 1;
  }

  // getheight

  else if(!strcmp(option, optTab[1])){
    printf("%s\t", optTab[1]);
    printf("%d\n", getHeight(Fd));
    n = 1;
  }

  // getobjects

  else if(!strcmp(option, optTab[2])){
    printf("%s\t", optTab[2]);
    printf("%d\n", getObjects(Fd));
    n = 1;
  }

  // getinfo

  else if(!strcmp(option, optTab[3])){
    printf("%s\t", optTab[3]);
    printf("Largeur : %d\tHauteur : %d\tNombre d'objets : %d\n", getWidth(Fd), getHeight(Fd)
        , getObjects(Fd));
    n = 1;
  }

  // setwidth

  else if(!strcmp(option, optTab[4])){
    printf("%s\t", optTab[4]);
    int w = atoi(arg);
    printf("%d\n", w);
    setWidth(Fd, w);
    n = 2;
  }

  // setheight

  else if(!strcmp(option, optTab[5])){
    printf("%s\t", optTab[5]);
    int h = atoi(arg);
    printf("%d\n", h);
    setHeight(Fd, h);
    n = 2;
  }

  // setobjects

  else if(!strcmp(option, optTab[6])){
    printf("%s\n", optTab[6]);
    if((argc - 3) % 6 != 0){
      fprintf(stderr, "Erreur,␣nombre␣d'arguments␣non␣valide\n");
      exit(1);
    }
    char *name = argv[k + 1];
    arg = argv[k + 2];
    int frame = atoi(arg);
    arg = argv[k + 3];
    int solid, destructible, collectible, generator;
    if(!strcmp(argv[k + 3], "solid"))
    solid = 2;
    else if(!strcmp(argv[k + 3], "semi_solid"))
    solid = 1;
    else if(!strcmp(argv[k + 3], "air"))
    solid = 0;
    else{
    fprintf(stderr, "Erreur,␣arguments␣non␣valide:\tsolid/semi_solid/air\n\n");
    exit(1);
  }
    if(!strcmp(argv[k + 4], "destructible"))
    destructible = 1;
    else if(!strcmp(argv[k+4], "not-destructible"))
    destructible = 0;
    else{
    fprintf(stderr, "Erreur,␣arguments␣non␣valide:\tdestructible/not-destructible\n\n");
    exit(1);
  }
    if(!strcmp(argv[k + 5], "collectible"))
    collectible = 1;
    else if(!strcmp(argv[k + 5], "not-collectible"))
    collectible = 0;
    else{
    fprintf(stderr, "Erreur,␣arguments␣non␣valide:\tcollectible/not-collectible\n\n");
    exit(1);
  }
    if(!strcmp(argv[k + 6], "generator"))
    generator = 1;
    else if(!strcmp(argv[k + 6], "not-generator"))
  generator = 0;
    else{
    fprintf(stderr, "Erreur,␣arguments␣non␣valide:\tgenerator/not-generator\n\n");
    exit(1);
  }
    setObjects(Fd, name, frame, solid, destructible, collectible, generator);
    n = argc - 2;
  }

  // pruneobjects

  else if(!strcmp(option, optTab[7])){
    printf("%s\n", optTab[7]);
    pruneOjects(Fd);
    n = 1;
  }
  else{
    printf("\nOption␣inconnue!\n");
    n = 1;
  }

  printf("\n");
  return n;
```

```c
}



int main(int argc, char *argv[]){
  char *optTab[NB_OPTIONS];
  int k = 2;
  int n, Fd;

  if(argc < 3)
    usage(argv[0]);

  Fd = open(argv[1], O_RDWR);

  if(Fd==-1){
    perror("open");
    exit(EXIT_FAILURE);
  }

  optionsAlloc(optTab);

  for(int i = 0; i < NB_OPTIONS; i++)
    printf("%d : %s\n", i, optTab[i]);

  // k est la position de l'option traitee dans argv

  while (k < argc){

    //n est le nombre d'arguments utilises

    n = traitementOption(optTab, Fd, argv, k, argc);
    k+= n;
    if(k >= 2){
    printf("maputil ne gere actuellement qu'une seule option a la fois!\n");
    break;
  }
  }

  optionsFree(optTab);
  close(Fd);
  return EXIT_SUCCESS;
}

#endif
```

Chapitre 3

tempo.c

```c
#define _XOPEN_SOURCE 700

#include <SDL.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <sys/time.h>
#include <signal.h>
#include <pthread.h>
#include "timer.h"



 static unsigned long get_time (void){
   struct timeval tv;
   gettimeofday (&tv ,NULL);

   // Compte seulement les secondes a partir de 2016

   tv.tv_sec -= 3600UL * 24 * 365 * 46;
   return tv.tv_sec * 1000000UL + tv.tv_usec;
 }



#ifdef PADAWAN



 struct evenement{
   void *parametre;
   unsigned long temps;
 };

 struct evenement t[100];
 int compteur = 0;



 void trie(struct evenement t[], int compteur){
     void* tmpparametre = 0;
     unsigned long tmptemps = 0;
     for(int i = 0; i < compteur; i++){
         for(int j = i + 1; j < compteur; j++){
           if(t[j].temps < t[i].temps){
                tmpparametre = t[i].parametre;
                  tmptemps = t[i].temps;
                  t[i].parametre = t[j].parametre;
                  t[i].temps = t[j].temps;
                  t[j].parametre = tmpparametre;
                  t[j].temps = tmptemps;
        }
         }
     }
 }



 void traitant(int s){
     printf ("sdl_push_event(%p) appelee au temps %ld\n", t[0].parametre, get_time ());
     sdl_push_event(t[0].parametre);

     for(int i = 0; i < compteur + 1; i++){
        t[i].temps = t[i + 1].temps;
        t[i].parametre = t[i + 1].parametre;
     }

     struct itimerval timer;
     timer.it_interval.tv_sec = 0;
     timer.it_interval.tv_usec = 0;
     timer.it_value.tv_sec = (t[0].temps-get_time()) / 1000000;
     timer.it_value.tv_usec = (t[0].temps-get_time()) % 1000000;
     int err = setitimer(ITIMER_REAL, &timer, 0);

     if(err){
        perror("setitimer");
        exit(1);
     }

     compteur--;
 }



 void *f(void *i){
     sigset_t mask, empty_mask;
     sigemptyset(&mask);
     sigemptyset(&empty_mask);
     sigaddset(&mask, SIGALRM);
     sigprocmask(SIG_BLOCK, &mask, NULL);
     struct sigaction s;
     s.sa_handler = traitant;
     sigemptyset(&s.sa_mask);
     s.sa_flags = 0;
     sigaction(SIGALRM, &s, NULL);

        while(1){
        sigsuspend(&empty_mask);
     }

 }



 // timer_init retourne 1 si les temporisateurs sont totalement implementes, sinon retourne 0

 int timer_init (void){
     pthread_t pid = (pthread_t)NULL;
     pthread_create(&pid, NULL, f, (void *)pid);

     // L'implementation est prete

     return 1;
 }


```

```

void timer_set (Uint32 delay, void *param){
    unsigned long time = (unsigned long)(delay * 1000) + get_time();
    t[compteur].temps = time;
    t[compteur].parametre = param;
    compteur++;
    trie(t, compteur);
    struct itimerval timer;
    timer.it_interval.tv_sec = 0;
    timer.it_interval.tv_usec = 0;
    timer.it_value.tv_sec = (t[0].temps-get_time()) / 1000000;
    timer.it_value.tv_usec = (t[0].temps-get_time()) % 1000000;
    int err = setitimer(ITIMER_REAL, &timer, 0);

    if(err){
      perror("setitimer");
      exit(1);
    }
}



#endif

// timerset :enfiler parametre, trie, timer init

// traitant :trie, timer init
```