```c
#include <dotneato.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>
#include "binTree.h"
Agraph_t * g;
void modifierAttr(void * obj, char type, char * attr, char * value) {
  Agsym_t * a;
  if(!(a = agfindattr(g->proto->n, attr)))
    switch(type) {
    case 'n': a = agnodeattr(g, attr, ""); break;
    case 'e': a = agedgeattr(g, attr, ""); break;
    default: break;
    }
  agxset(obj, a->index, value);
}
void ajouterNoeud(Agnode_t * parent, int v, AB * apartir, char * pos) {
  static int count = 0;
  char buf[BUFSIZ];
  sprintf(buf, "n%d", count);
  *apartir = malloc(sizeof (*apartir)[0]); assert(*apartir);
  (*apartir)->v = v; (*apartir)->fg =(*apartir)->fd = NULL;
  (*apartir)->n = agnode(g, buf);
  modifierAttr((*apartir)->n, 'n', "shape", "record");
  modifierAttr((*apartir)->n, 'n', "width", "2.0");
  sprintf(buf, "{%d|{<fg>|<fd>}}", v);
  modifierAttr((*apartir)->n, 'n', "label", buf);
  if(parent) {
    Agedge_t * e;
    modifierAttr(e = agedge(g, parent, (*apartir)->n), 'e', "tailport", pos);
    modifierAttr(e, 'e', "arrowhead", "none");
  }
  count++;
}
void placerNoeud(Agnode_t * parent, int v, AB * apartir, char * pos) {
  if(*apartir == NULL)
    ajouterNoeud(parent, v, apartir, pos);
  else if((*apartir)->v >= v)
    placerNoeud((*apartir)->n, v, &((*apartir)->fg), "fg");
  else
    placerNoeud((*apartir)->n, v, &((*apartir)->fd),  "fd");
}
void initArbre(AB * pracine) {
  int i, t[NB_NOEUDS], j, k = 0;
  srand(time(NULL));
  for(i = 0; i < NB_NOEUDS; i++) t[i] = i + 1;
  for(i = 0; i < NB_NOEUDS; i++) {
    j = ALEA; while(k == j) k = ALEA;
    t[j] ^= t[k]; t[k] ^= t[j]; t[j] ^= t[k];
  }
  for(i = 0; i < NB_NOEUDS; i++)
    placerNoeud(NULL, t[i], pracine, "");
}
void parcourtPrefix(AB apartir) {
  static AB precedent = NULL;
  if(apartir->fg)
    parcourtPrefix(apartir->fg);
  if(precedent)
    modifierAttr(agedge(g, precedent->n, apartir->n), 'e', "style", "dotted");
  precedent = apartir;
  if(apartir->fd)
    parcourtPrefix(apartir->fd);
}
int main(int argc, char ** argv) {
  GVC_t    * gvc;
  AB racine = NULL;
  gvc = gvContext();
  dotneato_initialize(gvc, argc, argv);
  g = agopen("g", AGDIGRAPH);
  initArbre(&racine);
  parcourtPrefix(racine);
  gvBindContext(gvc, g);
  dot_layout(g);
  dotneato_write(gvc);
  neato_cleanup(g);
  agclose(g);
  dotneato_terminate(gvc);
  return 0;
}
```

FIG. 1 – Rendu d'un arbre binaire en utilisant *Graphviz Library* «`binTree.c`».

```
1  #define NB_NOEUDS 20
2  #define ALEA (NB_NOEUDS * (rand()/(RAND_MAX + 1.0)))
3  typedef struct Noeud Noeud;
4  typedef struct Noeud * AB;
5  struct Noeud {
6    int v;
7    AB fg, fd;
8    Agnode_t * n;
9  };
```

FIG. 2 – Fichier «`binTree.h`».

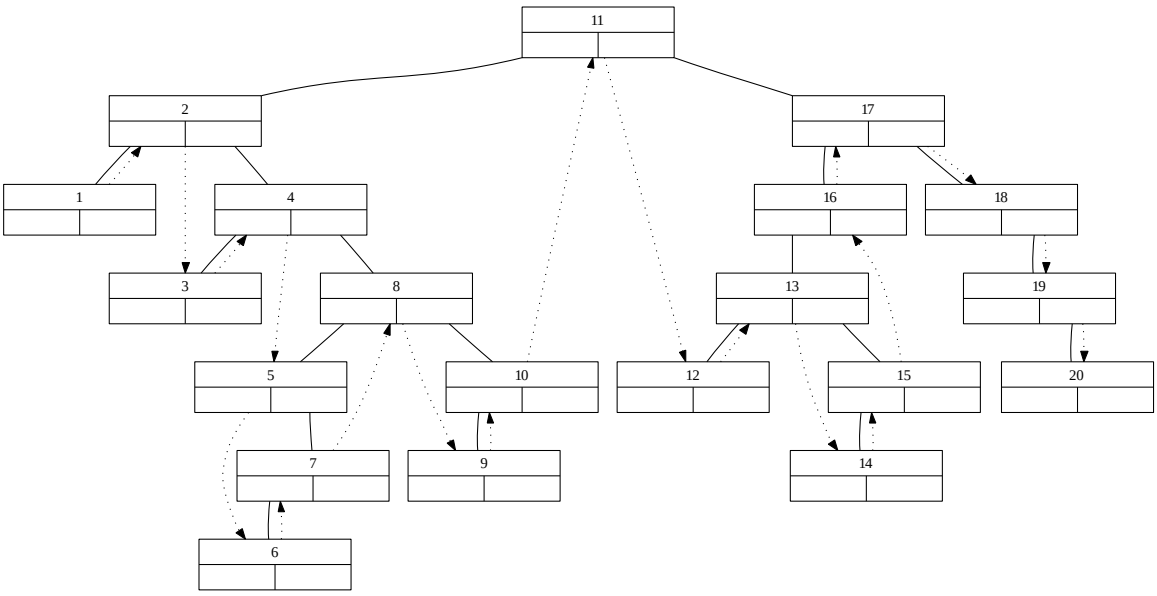

FIG. 3 – Arbre binaire obtenu après compilation et exécution du code FIG. 1.

```
1  #include <dotneato.h>
2  #include <sys/types.h>
3  #include <dirent.h>
4  #include <stdio.h>
5  #include <string.h>
6  Agraph_t * g;
7  void modifierAttr(void * obj, char type, char * attr, char * value) {
8    Agsym_t * a;
9    if(!(a = agfindattr(g->proto->n, attr)))
10     switch(type) {
11     case 'n': a = agnodeattr(g, attr, ""); break;
12     case 'e': a = agedgeattr(g, attr, ""); break;
13     default: break;
14     }
15   agxset(obj, a->index, value);
16 }
17 int recuDir(char * path, char * name, Agnode_t * parent) {
18   DIR * dir;
19   char buf[BUFSIZ];
20   struct dirent * dirent;
21   Agnode_t * n;
22   n = agnode(g, path);
23   modifierAttr(n, 'n', "label", name);
24   if(parent)
25     agedge(g, parent, n);
26   if((dir = opendir(path)) == NULL) return 0;
27   while((dirent = readdir(dir))) {
28     if(!strcmp(dirent->d_name, ".") || !strcmp(dirent->d_name, "..")) continue;
29     sprintf(buf, "%s/%s", path, dirent->d_name);
30     recuDir(buf, dirent->d_name, n);
31   }
32   closedir(dir);
33   return 1;
34 }
35 int main(int argc, char ** argv) {
36   GVC_t    * gvc;
37   if(argc < 2) {
38     fprintf(stderr, "usage : %s path [dot options]\n", argv[0]);
39     exit(1);
40   }
41   gvc = gvContext();
42   dotneato_initialize(gvc, argc, argv);
43   g = agopen("g", AGDIGRAPH);
44   recuDir(argv[1], ".", NULL);
45   gvBindContext(gvc, g);
46   dot_layout(g);
47   dotneato_write(gvc);
48   neato_cleanup(g);
49   agclose(g);
50   dotneato_terminate(gvc);
51   return 0;
52 }
```

FIG. 4 – Construction du graph `dot` représentant l'arborescence d'un sous répertoire «`repArbo.c`».

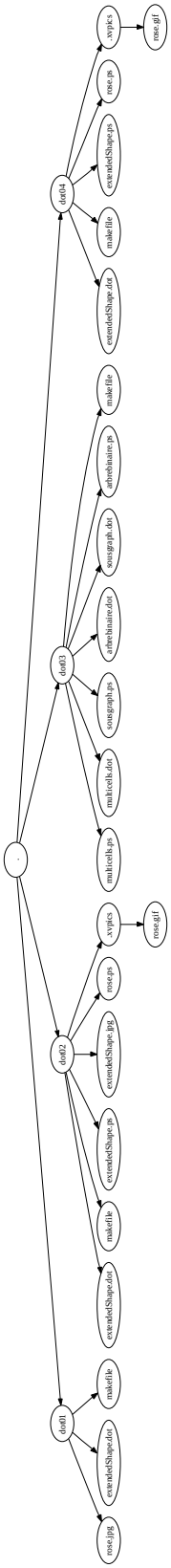

FIG. 5 – Arborescence d'un répertoire obtenue après compilation et exécution du code FIG. 4.