par Jean−Etienne Poirrier
(homepage)

*L´auteur:*

Jean−Etienne Poirrier est un doctorant à l'Université de Liège, en Belgique. Il étudie la consolidation de la mémoire chez les animaux. Il s'intéresse aussi aux logiciels libres et à l'Open Access.

# Automatiser la création de graphiques avec Graphviz

*Résumé*:

Dans cet article, nous apprendrons comment créer quelques graphes simples avec Graphviz, un logiciel open source en ligne de commande pour la visualisation de graphes.

*Traduit en Français par:*
Jean−Etienne Poirrier
(homepage)

## Introduction

Nous avons tous voulu, un jour, dessiner un simple graphe liant deux concepts, un diagramme représentant un processus, une structure de base de données, un réseau d'ordinateurs ou tout autre relation. Graphviz (http://graphviz.org) [1] est un logiciel open source de visualisation de graphes : il va vous aider à représenter de tels informations structurées. C'est un outil puissant, travaillant avec des algorithmes spéciaux afin de créer le meilleur graphe possible.

Un autre avantage est que c'est un outil en ligne de commande. Vous pourrez générer un graphique à partir d'une simple commande, d'un énorme fichier texte ou, automatiquement, à partir de données générées par un autre processus. Graphiviz peut être utilisé avec des outils interactifs pour générer ces graphiques mais son but n'est pas de ressembler à MS−Visio ou Smartdraw.

Graphviz est composé de deux outils principaux : *dot* et *neato*. Chacun a sa spécialité, comme nous le verrons plus bas... Ce petit article va se focaliser sur ces deux outils en ligne de commande avec des données que vous lui fournirez. Mais vous pouvez utiliser ces outils à partir d'un serveur web, d'un autre logiciel, etc.

## Installation

L'installation de Graphviz est très simple car de nombreuses distributions GNU/Linux l'incluent, d'une manière ou d'une autre. Par exemple, si vous utilisez Debian, vous pouvez l'installer avec cette commande :

```
# apt-get install graphviz-doc
```

Si vous utiliser une Fedora Core 3 (comme moi), vous pouvez ajoutez le dépôt Extra [2] et, ensuite, taper cette commande :

```
# yum install graphviz-2.2-3
```

Si vous avez une autre distribution, vous devriez chercher Graphviz dans son dépôt de logiciels.

Enfin, vous pouvez toujours installer Graphviz à partir des sources après les avoir téléchargées [3]. Après avoir accepté la licence, vous pourrez télécharger le code source ou des paquets exécutables pour Apple, MS−Windows, Sun Solaris, etc.

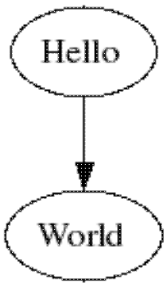
Un dernier mot à propos de la licence : pour les versions au−dessus de la 1.3, c'est la Common Public License Version 1.0 [4], une licence de logiciel libre (mais non compatible avec la GNU GPL).

## « Hello World! »

Que vous ayez ou non appris un langage de programmation, vous devez probablement savoir qu'une des premières choses qu'on apprend est d'afficher la chaîne « Hello World! ». Nous n'allons pas déroger à cette bonne vieille habitude, voici une manière de l'afficher avec l'outil *dot* :

```
echo "digraph G {Hello->World}" | dot −Tpng >hello.png
```

Le résultat est :




C'est la première façon de dessiner un graphe : afficher les commandes et les rediriger vers *dot*. L'option −Tpng indique à *dot* d'écrire le résultat dans un fichier PNG. Bien sûr, vous pouvez utiliser beaucoup de formats comme JPG, SVG, Postscript (et donc PDF), etc.

La commande est assez simple à comprendre. Avec *dot*, nous dessinons des graphes dirigés : ils ont une direction, nous avons des flèches entre les noeuds (*nodes*). Dans ce but, nous utilisons le mot clé « digraph » et nous nommons ce graphe « G ». Entre les crochets, nous mettons tout ce qui est inclus dans le graphe dirigé : deux noeuds (« Hello » et « World ») et une flèche (ou « bord », *edge*) allant de « Hello » à « World ». Simple, n'est−ce pas ?

# La hiérarchie de LinuxFocus

La seconde manière de dessiner un graphe est en utilisant un fichier texte et la ligne de commande. Pour comprendre l'avantage d'écrire tout dans un fichier texte, prenons un autre exemple...

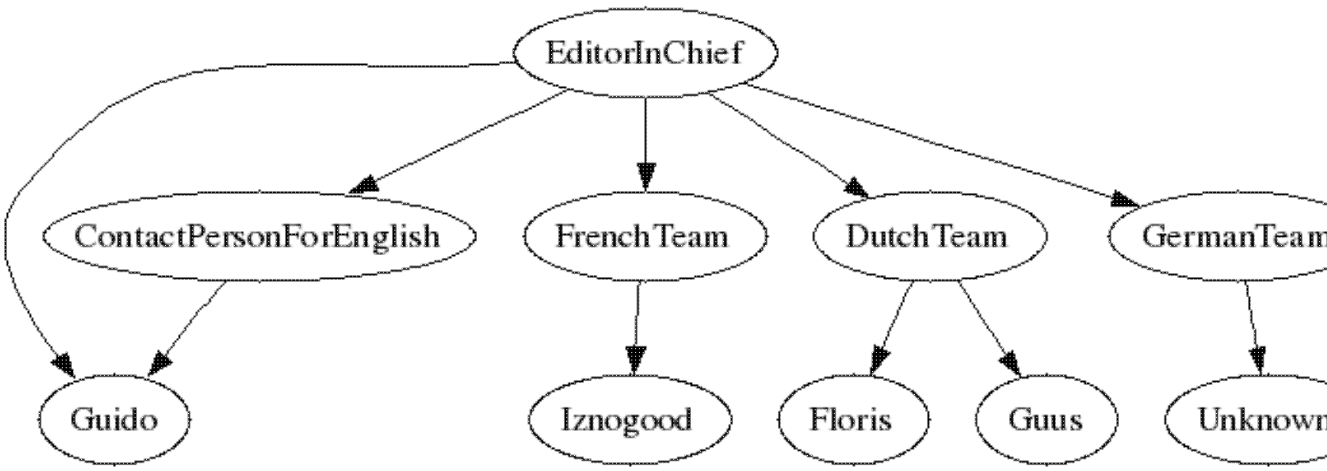
Si nous jetons un oeil à la liste de contact pour LinuxFocus [5], nous voyons qu'il y a un Editeur en chef et plusieurs personnes responsables d'une équipe d'une langue. Voici un résumé des données dont nous aurons besoin pour créer notre diagramme (j'ai seulement utilisé les prénoms pour la simplicité) :

| Equipe/Rôle | Personne |
|---|---|
| Editor−in−Chief | Guido |
| French Team | Iznogood |
| Dutch Team | Floris et Guus |
| Contact Person for English | Guido |
| German Team | ? |

En utilisant un point virgule pour séparer les noeuds liés, je peux écrire un simple fichier texte résumant les relations :

```
digraph G {
  EditorInChief -> Guido;
  EditorInChief -> FrenchTeam;
  EditorInChief -> DutchTeam;
  EditorInChief -> ContactPersonForEnglish;
  EditorInChief -> GermanTeam;
  FrenchTeam -> Iznogood;
  DutchTeam -> Floris;
  DutchTeam -> Guus;
  ContactPersonForEnglish -> Guido;
  GermanTeam -> Unknown;
}
```

Nous le sauvons sous le nom « lf−hierarchy1.dot », lançons la commande « dot −Tpng lf−hierarchy1.dot −o lf−hierarchy1.png » et nous obtenons ce merveilleux graphe :

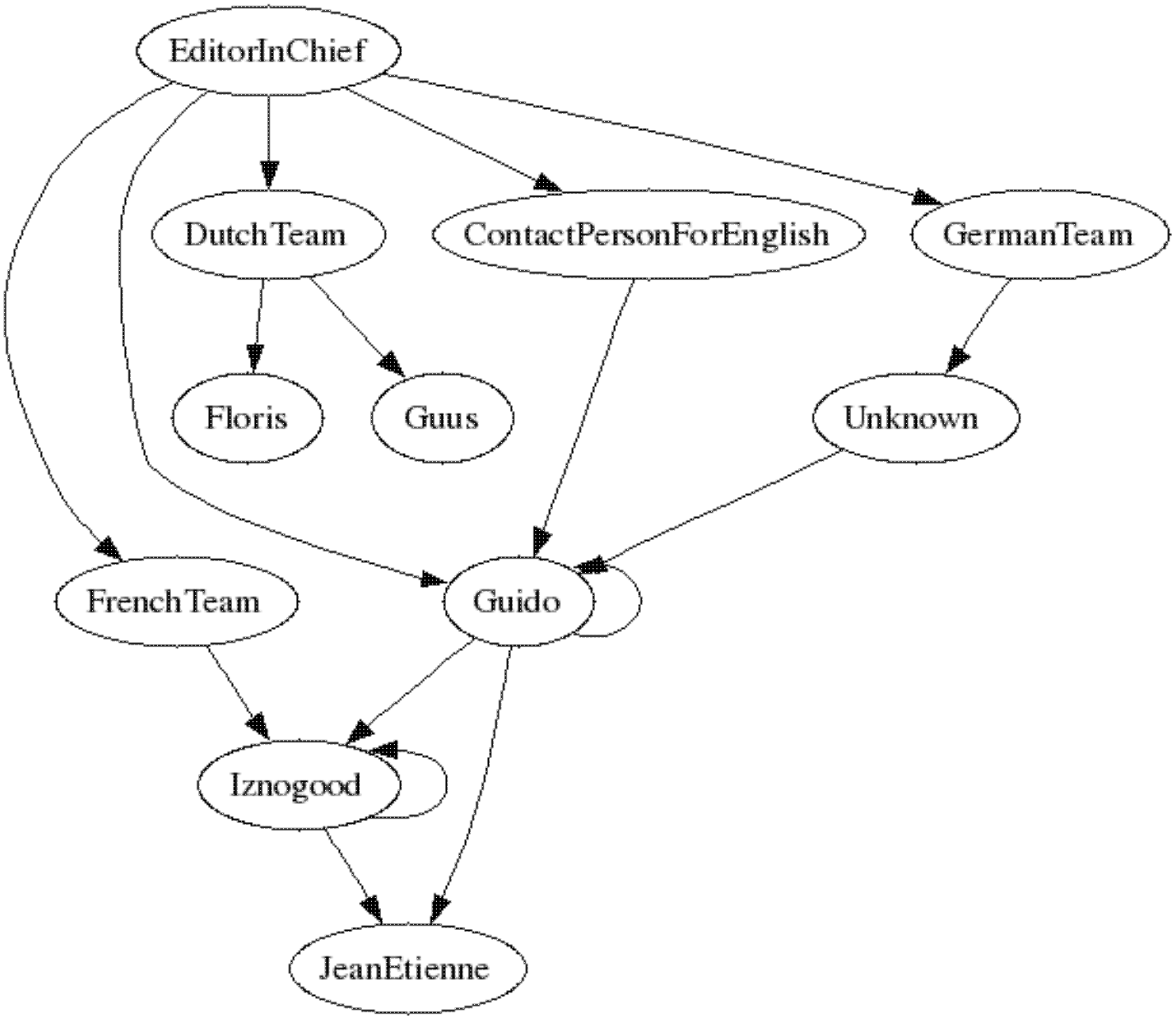



Comme vous pouvez le voir, *dot* a aligné les noms des équipes (*Team*) sur une ligne et les prénoms sur une autre ligne. Il a automatiquement donné deux flèches en−dessous du DutchTeam, pour les deux prénoms. Et il a automatiquement montré les statuts de Guido, bien que les relations sont assez éloignées dans le fichier texte. C'est un bon point de départ.

Maintenant, nous allons ajouter trois auteurs : Guido, Iznogood et moi−même (JeanEtienne). Assumons que Guido écrive en anglais et en allemand, qu'Iznogood écrive en anglais et en français et que j'écrive en anglais et en français. Supposons aussi que les auteurs soient dirigés par les chefs d'équipe. Nous pouvons continuer à ajouter des relations dans le fichier .dot et voyons ce qui se passe :

```
digraph G {
  EditorInChief -> Guido;
  EditorInChief -> FrenchTeam;
  EditorInChief -> DutchTeam;
  EditorInChief -> ContactPersonForEnglish;
  EditorInChief -> GermanTeam;
  FrenchTeam -> Iznogood;
  DutchTeam -> Floris;
  DutchTeam -> Guus;
  ContactPersonForEnglish -> Guido;
  GermanTeam -> Unknown;
  Guido -> Guido;
  Unknown -> Guido;
  Guido -> Iznogood;
  Iznogood -> Iznogood;
  Guido -> JeanEtienne;
  Iznogood -> JeanEtienne;
}
```

Nous le sauvons sous le nom « lf−hierarchy2.dot », lançons la commande « dot −Tpng lf−hierarchy2.dot −o lf−hierarchy2.png » et nous obtenons ce « graphe » :




Voyons les choses en face : bien que *dot* ait respecté toutes les relation que nous avons écrites, le graphe semble étrange, nous avons perdu l'alignement des fonctions et, des bulles partout, c'est assez ennuyant.

## Raffinement du graphe

Raffiner quelques éléments du graphe (noeud, bord, ...) est très simple : chaque élément personnalisé est suivi par ces crochets droits (« [ » et « ] ») contenant des paires nom − valeurs de chaînes de caractères. Par exemple, nous pouvons écrire « élément [option = value]; ». Nous allons utiliser quelques options. Ceux qui sont intéressés par plus d'options peuvent lire le *dot* user guide [6].

Tout d'abord, nous pouvons donner différentes formes aux différents rôles. Gardons les ellipses pour les rôles abstraits comme « EditorInChief », « DutchTeam », ... mais nous utiliserons une boîte pour les gens réels. Cela est fait par l'utilisation de cette chaîne après les prénoms : « [shape = box] », seul sur une ligne. Par exemple, nous écrirons « Guido [shape = box]; » avant toute utilisation de son noeud.

Ensuite, nous pouvons utiliser différents types de flèches (*edge*), selon que la personne est un auteur ou un chef d'équipe. Par exemple, nous pouvons demander des lignes grasses entre l'EditorInChief et chaque équipe. Une des relations sera alors « EditorInChief−>GermanTeam [style = bold]; ».

Nous pouvons aussi demander des couleurs : par exemple, rouge pour les chefs d'équipe et vert pour les auteurs. Cela clarifiera les différentes relations. Comme vous pouvez le deviner, l'option est, par exemple, « [color = red] ». Nous pouvons évidemment écrire « FrenchTeam –> Iznogood [color = red]; » mais nous serions alors obligé de l'écrire pour toutes les relations. Nous pouvons alors demander que toutes les futures flèches soient en rouge en écrivant « edge [color = red]; ».

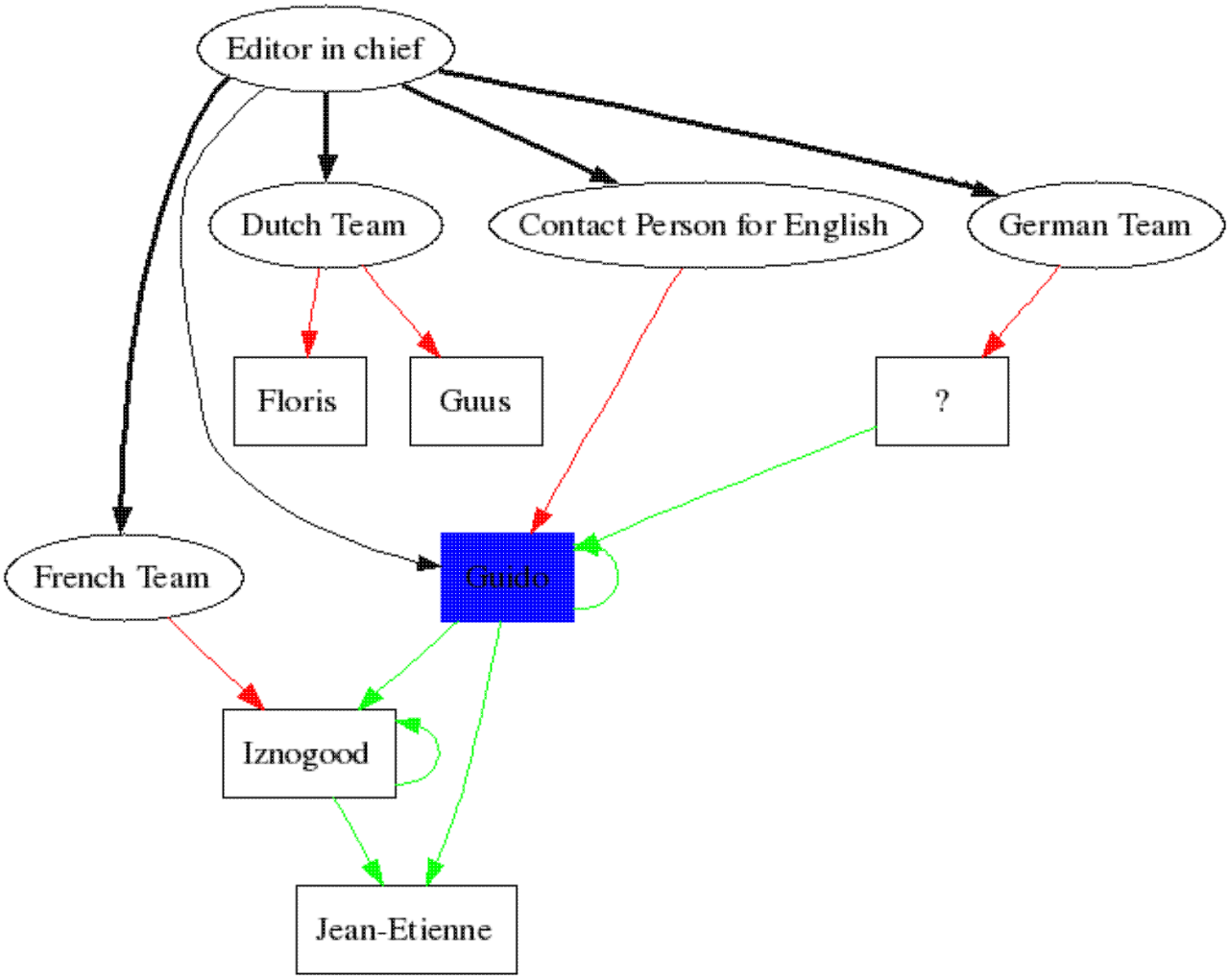
Bien que nous sachions ce que « EditorInChief » devrait signifier, il serait mieux de l'écrire correctement, avec des espaces. C'est ici que nous avons besoin de l'option « label ». « EditorInChief » deviendra « EditorInChief [label = "Editor in chief"]; ». Et nous pouvons changer « Unknown » en un « ? » plus visuel.

Enfin, si nous voulons mettre en évidence l'Editeur en chef, vous pouvez remplir son ellipse avec l'option suivante : « Guido [style = filled, color = blue]; » (par exemple).

Si on suit tous ces raffinements, nous nous retrouvons avec ce fichier texte :

```
digraph G {
  Guido [shape = box, style = filled, color = blue];
  Iznogood [shape = box];
  Floris [shape = box];
  Guus [shape = box];
  Unknown [shape = box, label = "?"];
  JeanEtienne [shape = box, label = "Jean-Etienne"];
  EditorInChief [label = "Editor in chief"];
  FrenchTeam [label = "French Team"];
  DutchTeam [label = "Dutch Team"];
  ContactPersonForEnglish [label = "Contact Person for English"];
  GermanTeam [label = "German Team"];

  EditorInChief -> Guido;
  EditorInChief -> FrenchTeam [style = bold];
  EditorInChief -> DutchTeam [style = bold];
  EditorInChief -> ContactPersonForEnglish [style = bold];
  EditorInChief -> GermanTeam [style = bold];

  edge [color = red]; /* defines the team leaders */
  FrenchTeam -> Iznogood;
  DutchTeam -> Floris;
  DutchTeam -> Guus;
  ContactPersonForEnglish -> Guido;
  GermanTeam -> Unknown;

  edge [color = green]; /* defines the authors */
  Guido -> Guido;
  Unknown -> Guido;
  Guido -> Iznogood;
  Iznogood -> Iznogood;
  Guido -> JeanEtienne;
  Iznogood -> JeanEtienne;
}
```

Nous le sauvons sous le nom « lf–hierarchy3.dot », lançons la commande « dot –Tpng lf–hierarchy3.dot –o lf–hierarchy3.png » et nous obtenons ce graphe fantaisiste :

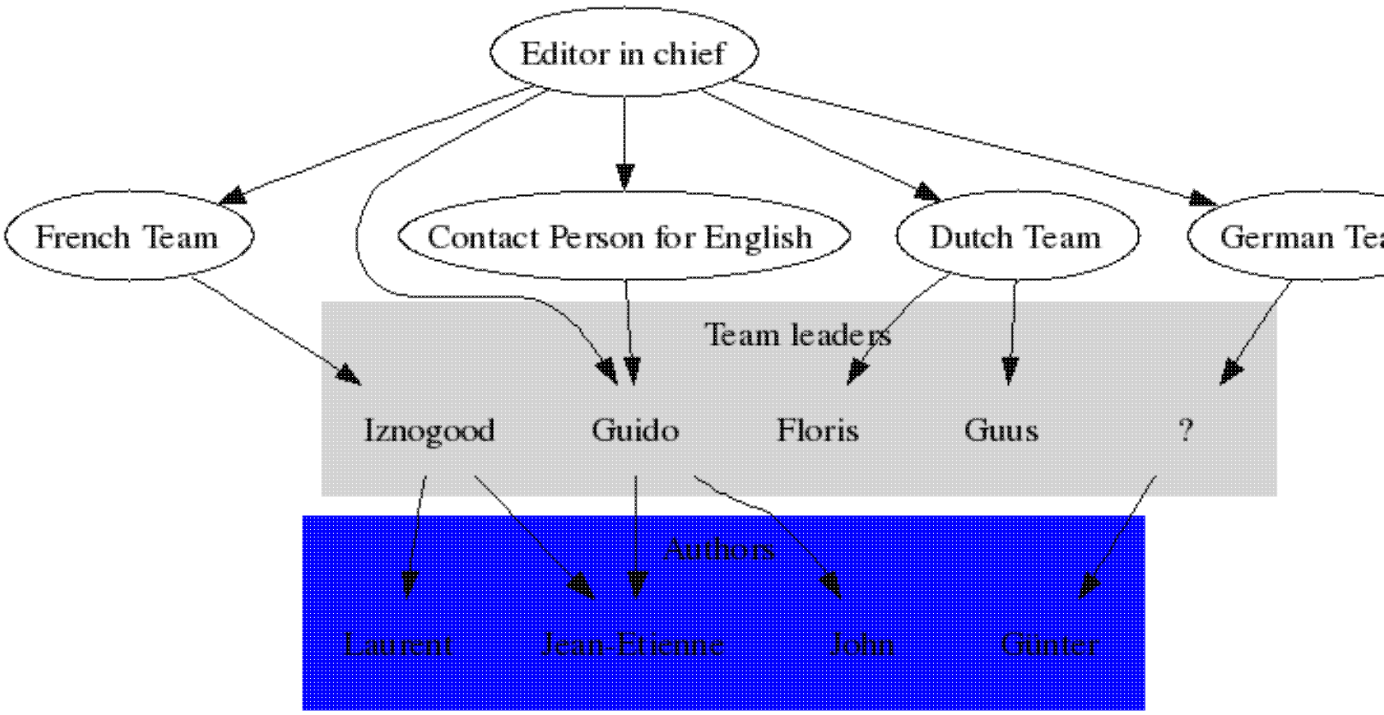



Revenons à notre premier graphe (celui sans les auteurs). Nous allons maintenant ajouter quelques auteurs mais nous supposons qu'un chef d'équipe ne peut pas être un auteur. Avec cela, nous pouvons grouper tous les chefs d'équipe dans un groupe et tous les auteurs dans un autre. Cela se fait avec la commande « subgraph » qui fonctionne un peu comme « digraph » (il a besoin d'un nom et tout est inclu entre des crochets). C'est mieux avec un exemple (j'ai aussi enlevé les formes autour des prénoms) :

```
digraph G {
  subgraph cluster_team_leaders {
          style = filled; /* defines the cluster */
          color = lightgrey;
          label = "Team leaders";

          Guido [shape = plaintext]; /* defines the elements in the cluster */
          Iznogood [shape = plaintext];
          Floris [shape = plaintext];
          Guus [shape = plaintext];
          Unknown [shape = plaintext, label = "?"];
  }

  subgraph cluster_authors {
          style = filled;
          color = blue;
          label = "Authors";

          JeanEtienne [shape = plaintext, label = "Jean-Etienne"];
          John [shape = plaintext];
```

```
        Gunter [shape = plaintext, label = "Günter"];
        Laurent [shape = plaintext];
  }

  EditorInChief [label = "Editor in chief"];
  FrenchTeam [label = "French Team"];
  DutchTeam [label = "Dutch Team"];
  ContactPersonForEnglish [label = "Contact Person for English"];
  GermanTeam [label = "German Team"];

  EditorInChief -> Guido;
  EditorInChief -> FrenchTeam;
  EditorInChief -> DutchTeam;
  EditorInChief -> ContactPersonForEnglish;
  EditorInChief -> GermanTeam;

  FrenchTeam -> Iznogood;
  DutchTeam -> Floris;
  DutchTeam -> Guus;
  ContactPersonForEnglish -> Guido;
  GermanTeam -> Unknown;

  Guido -> John;
  Unknown -> Gunter;
  Iznogood -> Laurent;
  Guido -> JeanEtienne;
  Iznogood -> JeanEtienne;
}
```

Nous le sauvons sous le nom « lf−hierarchy4.dot », lançons la commande « dot −Tpng lf−hierarchy4.dot −o lf−hierarchy4.png » et nous obtenons ce graphe :




Je pense que c'est assez pour éditer des graphes de base avec *dot*. Si vous êtes intéressés par d'autres options, jetez un oeil au *dot* user guide [6].

# Réseaux

Comme vous l'avez vu dans les paragraphes précédents, les liens (edges) entre les noeuds sont tous des flèches. C'est pourquoi ils sont appelés « graphes dirigés ». Maintenant, si je veux représenter un réseau de gens sans hiérarchie, je n'ai pas besoin de ces flèches. Mais j'ai besoin d'un autre outil : *neato*. *Neato* suit une syntaxe similaire à *dot* et son guide [7] est aussi très clair et facile à lire. Nous allons voir ici quelques concepts de base des graphes non dirigés.

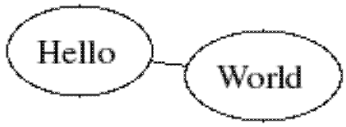
Si nous revenons à notre exemple « Hello World! », voici comment le réaliser avec *neato*:

```
echo "graph G {Hello--World}" | neato -Tpng
>hello2.png
```

Les différences principales avec cette ligne de commande sont :

- nous utilisons *neato* à la place de *dot*;
- maintenant, c'est un « graph » et pas un « digraph » (nous avons perdu la « direction »);
- et nous avons également perdu la flèche (nous avons une simple ligne à la place).

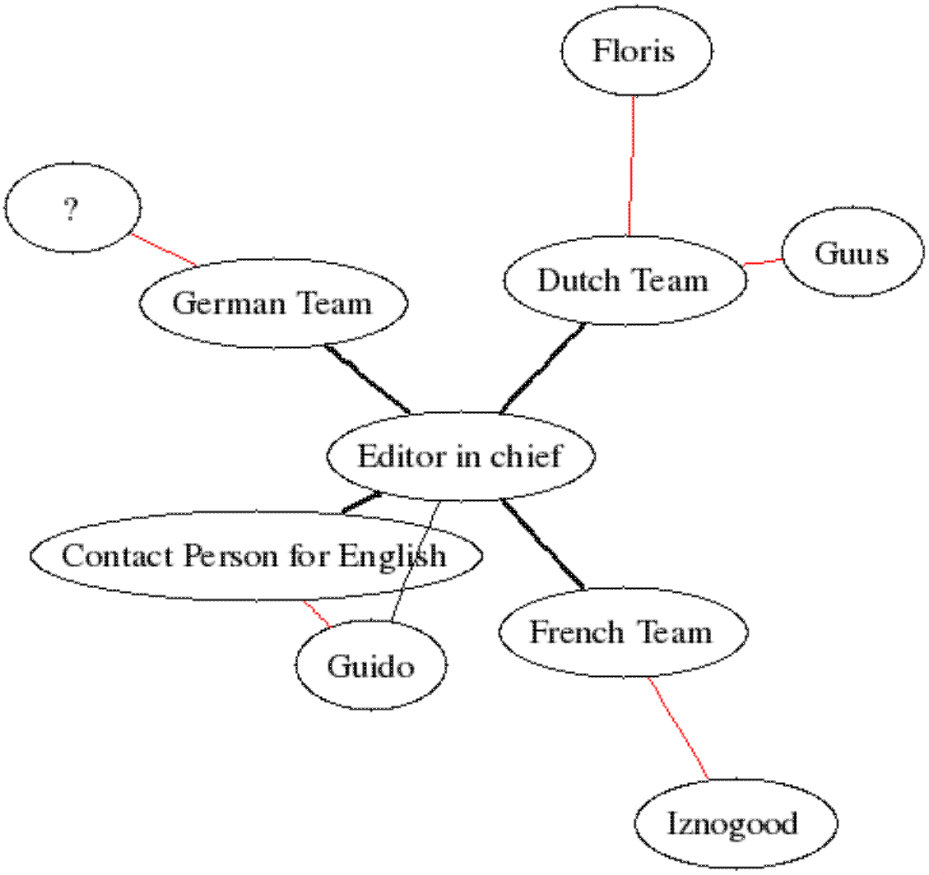
Comme prévu, le résultat ressemble un peu au premier « Hello World » que nous avons eu :




En fait, la syntaxe pour *neato* est très semblable à celle utilisée dans *dot* (c'est la même syntaxe de base, à l'exception d'options spécifiques pour chaque outil). Si nous prenons la première hiérarchie de LinuxFocus (lf–hierarchy1.dot) et l'adaptons pour *neato* (càd. remplaçons « digraph » par « graph » et « –> » par « –– »), nous avons maintenant « lf–hierarchy5.dot » :

```
graph G {
  Unknown [label = "?"];
  EditorInChief [label = "Editor in chief"];
  FrenchTeam [label = "French Team"];
  DutchTeam [label = "Dutch Team"];
  ContactPersonForEnglish [label = "Contact Person for English"];
  GermanTeam [label = "German Team"];

  EditorInChief -- Guido;
  EditorInChief -- FrenchTeam [style = bold];
  EditorInChief -- DutchTeam [style = bold];
  EditorInChief -- ContactPersonForEnglish [style = bold];
  EditorInChief -- GermanTeam [style = bold];

  edge [color = red]; /* defines the team leaders */
  FrenchTeam -- Iznogood;
  DutchTeam -- Floris;
  DutchTeam -- Guus;
  ContactPersonForEnglish -- Guido;
  GermanTeam -- Unknown;
}
```

Si nous lançons la commande « neato –Tpng lf–hierarchy5.dot –o lf–hierarchy5.png », nous obtenons ce graphe :

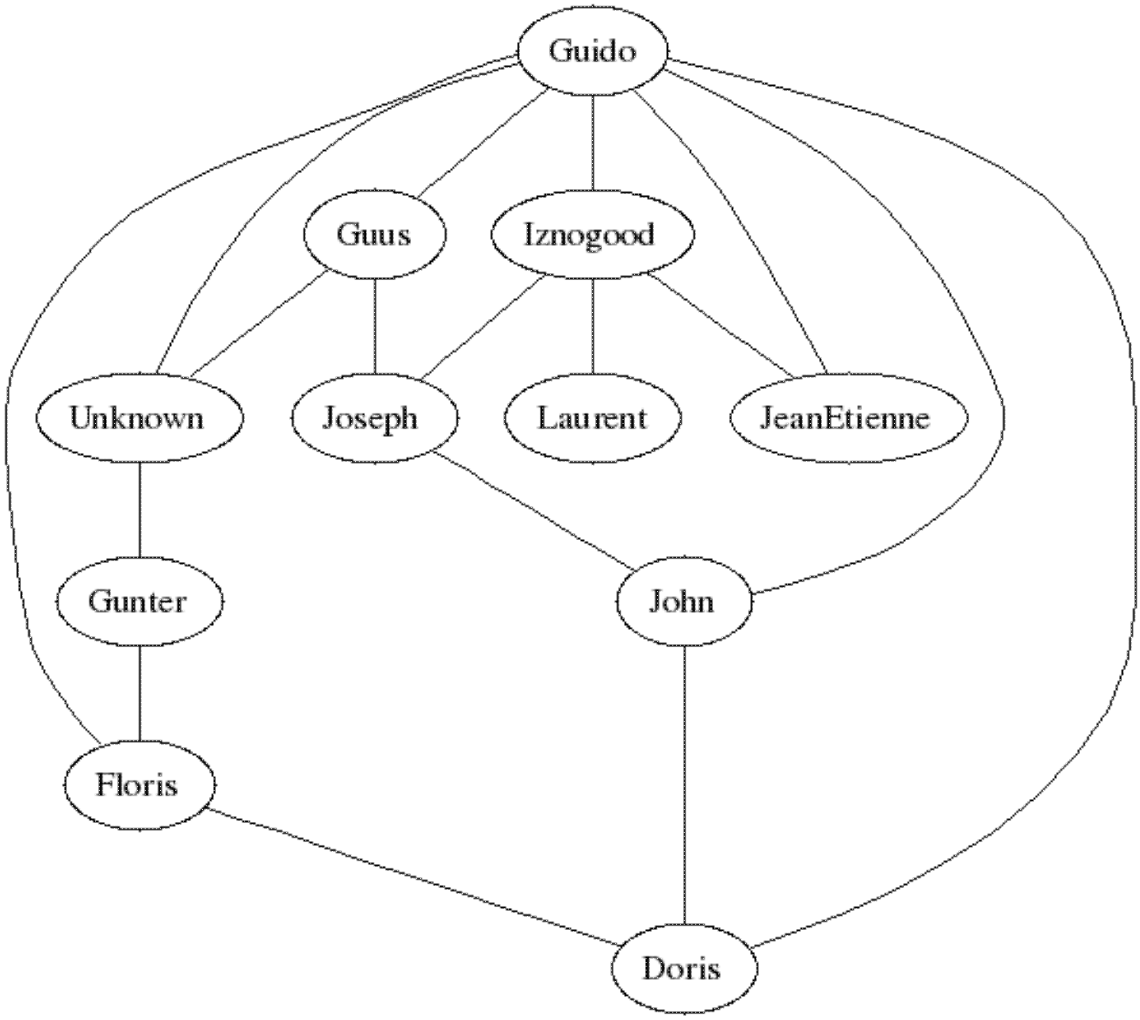



Comme nous pouvons le voir, ce graphe ressemble un peu à la premi&eagrave;re hiérarchie. Mais *neato* a essayé d'étendre tous les éléments sans aucune direction. Maintenant, si nous reprenons les auteurs, mélangeons tout le monde et souhaitons afficher seulement les relations entre les gens, je peux afficher un graphe similaire à la première hiérarchie, sans les flèches :

```
graph G {
  Guido -- Iznogood;
  Guido -- Floris;
  Guido -- Guus;
  Guido -- Unknown;
  Iznogood -- Laurent;
  Iznogood -- JeanEtienne;
  Guido -- JeanEtienne;
  Guido -- John;
  Unknown -- Gunter;
  Guido -- Doris;
  Iznogood -- Joseph;
  Guus -- Joseph;
  Guus -- Unknown;
  John -- Doris;
  Joseph -- John;
  Gunter -- Floris;
  Floris -- Doris;
}
```

Nous sauvons le tout sous le nom « lf–people.dot » et lançons la commande « dot –Tpng lf–people.dot –o

lf−people.png ». Voici le résultat :




Maintenant, je pense que vous en avez appris assez pour construire un réseau d'ordinateurs, par exemple. Et rappelez−vous que si vous avez besoin d'aide, le *neato* user guide [7] est très utile.

# Conclusion

Dans cet article, vous avez appris comment créer quelques graphes simples avec graphviz, un outil open source de visualisation de graphes en ligne de commande. Si vous voulez aller plus loin et en apprendre plus sur les outils et options de Graphviz, alors rendez−vous sur la page web de Graphviz [1]. Vous découvrirez d'autres types de graphes que vous pouvez réaliser dans la Gallerie [8] et, si vous êtes intéressés, vous pouvez trouvez de nombreux liens sur la théorie derrière la visualisation de graphes [9]. Enfin, vous pouvez aussi télécharger tous les scripts and graphs utilisés dans cet article [10].

Un grand merci à l'équipe Graphviz chez AT&T Research pour ces outils !

# Références


- [1] Page web de Graphviz : http://graphviz.org/
- [2] Ajouter le dépôt Extra à la Fedora Core 3: http://www.fedoraproject.org/wiki/Extras/UsingExtras
- [3] Télécharger le code source de Graphviz : http://graphviz.org/Download.php
- [4] Graphviz'Common Public Licence Version 1.0 : http://graphviz.org/License.php
- [5] Personnes à contacter pour LinuxFocus : http://linuxfocus.org/common/lfteam.html
- [6] *dot* user guide (PDF) : http://graphviz.org/Documentation/dotguide.pdf
- [7] *neato* user guide (PDF) : http://graphviz.org/Documentation/neatoguide.pdf
- [8] Gallerie Graphviz : http://graphviz.org/Gallery.php
- [9] Théorie derrière la visualisation des graphes : http://graphviz.org/Theory.php
- [10] Téléchargez tous les scripts et graphes de cet article : répertoire de téléchargement pour graphviz. Vous pouvez juste prendre le fichier lf−graphviz.tar.gz qui inclus tous les fichiers et images.


---



2005−08−26, generated by lfparser_pdf version 2.51