Docs.Framasoft.org : un site pour apprendre à utiliser tous nos services !

Mine de rien, entre les services Dégooglisons Internet et les projets Framasoft, nous maintenons près d'une cinquantaine de sites/projets ouverts au public.

C'est bien joli, mais si on n'accompagne pas ces sites des savoir-faire et outils pour mieux vous aider à vous en emparer… c'est triste, non ?

Un peu de cathédrale dans notre joli bazar…

Depuis près de trois ans que nous Dégooglisons Internet, nous n'aurions rien pu faire sans votre aide. Nous savons que proposer des outils c'est bien, et que cela ne suffit pas. Il faut aussi les présenter, donner des tutoriels, des outils pour les comprendre et les prendre en main.

Bien entendu, ces logiciels sont déjà souvent soutenus par leurs propres communautés, qui proposent leur propre documentation dont chacun·e peut bénéficier. Il nous fallait, néanmoins, un endroit où rassembler tout cela.

Et depuis trois ans, nombre d'entre nous (contributeurs et contributrices, bénévoles et salarié·e·s…) ont apporté leur petite pierre à l'édifice. Il fallait nous voir, à chaque nouvelle contribution, nous émerveiller :

> *« Chouette ! Arpinux a fait une vidéo de prise en main de Framapic, pour mieux héberger ses images ! »*
>
> *« Ah ! je me suis bien marré devant la présentation de*

Framapack que Pyves vient de nous proposer. J'espère que de plus en plus de windowsiens l'utiliseront pour télécharger des logiciels libres… »

« Attend, en plus de coder des fonctionnalités à Nexcloud pour ouvrir Framagenda, Tcit il s'est fadé d'écrire une jolie documentation pour synchroniser ses rendez-vous et ses contacts… GG ! »

« Sérieusement, le groupe Framalang s'est encore surpassé en traduisant la doc de Mattermost… Ça va bien aider à ce que les gens s'emparent de Frameteam pour abandonner leurs groupes Facebook. »

« Oh, tu as vu la vidéo de SVTux pour découvrir Framapad ? En deux minutes, on voit que le libre peut faire aussi bien que GoogleDocs. »

« Pouhiou a encore trippé sur sa présentation de Framanotes. J'espère que Turtl aura autant de succès qu'Evernotes… »

« Franchement, les tutos de Cartocité pour utiliser Umap et Framacartes sont excellents… Si ça pouvait libérer les gens de Google Maps… »

(L'est-y pas belle, la vidéo Framalistes de Nicolas Geiger pour le site Colecti.cc ?)

Au départ, nous avons essayé de mettre les liens vers ces outils de documentation dans chaque page d'accueil de chacun de nos projets, afin que vous ayez tout sous la main dès que vous commencez à vous y intéresser… Mais souvent, une fois que vous êtes dans le service, vous n'allez plus voir la page d'accueil. On le sait, parce que nous, on fait pareil.

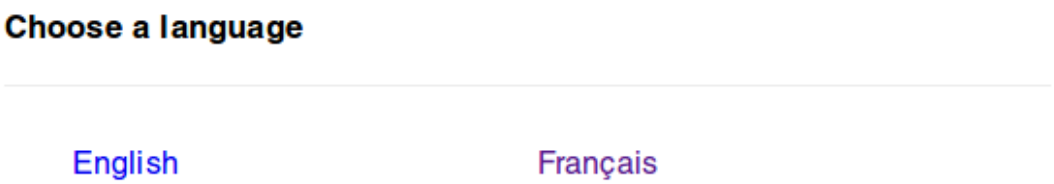
Alors nous avons lancé le défi à JosephK de mettre un peu de cathédrale dans ce merveilleux bazar, et de rassembler nos

documentations en un seul et même endroit. N'écoutant que les clapotis de son clavier, ce dernier a décidé de collecter, d'organiser et de présenter tout cela sous la forme d'un gitbook, afin d'avoir un outil que l'on puisse modifier, amender et mettre à jour de façon collaborative (et pas trop ardue).

Demandez la doc !

Le principe est simple : vous cherchez comment utiliser un de nos services ? Pourquoi choisir tel Frama-bidule ? À quoi sert tel Framachin ? rendez-vous sur docs.framasoft.org.

Vous y serez accueillies par un choix de langue (parce qu'un jour, peut-être, on pourrait avoir des versions en anglais, breton ou espéranto).



C'est sommaire, mais éloquent.

Puis sur la page d'accueil, vous verrez une barre latérale qui vous permet de vous guider dans l'ensemble de notre documentation (elle s'adapte selon la rubrique dans laquelle vous vous trouvez). C'est dans la colonne principale, à droite, que se trouvent l'accès aux informations. Tout en haut, vous y trouverez des guides pratiques.

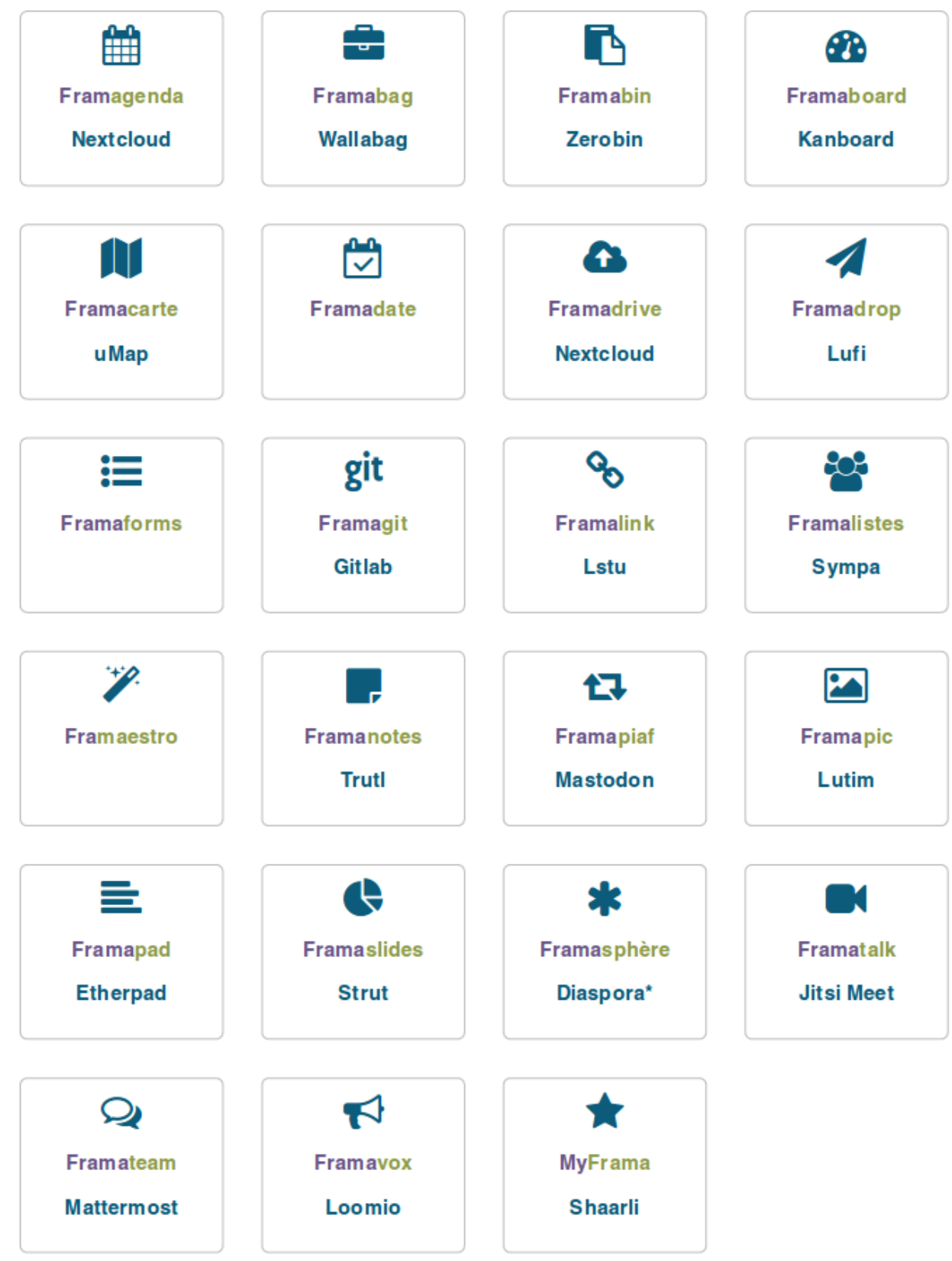
Les deux premiers guides disponibles à ce jour.

Ce sont des guides à destination du grand public, regorgeant d'informations aussi pratiques qu'indispensables. Pour l'instant, nous y avons inclus :

- Libertés Numériques, de C. Masutti, déjà paru chez Framabook
- L'auto-hébergement facile, de X. Cartron

(si vous voulez nous proposer le vôtre, rendez-vous dans la prochaine partie de cet article)

Ensuite, toujours sur cette page, vous y trouverez une liste des services Framasoft.



Tous les services n’y sont pas (encore) présents… alors proposez vos contributions !

Ce sont l'ensemble des services Dégooglisons Internet sur lesquels nous avons une documentation en Français et (plus ou moins ^^) à jour à proposer.

Il vous suffit de cliquer sur le service qui vous intéresse pour découvrir les outils que nous avons pu récolter à son sujet.

Bien entendu, si vous ne trouvez pas votre service préféré et/ou que vous souhaitez proposer un élément de documentation, nous sommes preneurs (voir plus bas).

Enfin, toujours sur cette page, vous y verrez une rubrique « Culture et Logiciels Libres »

Les premiers Frama-Projets à bénéficier de leur documentation !

Ici, vous aurez des savoirs et savoir-faire sur les projets Framasoft qui ne sont pas des services Dégooglisons, qui tendent à promouvoir le logiciel libre et sa culture.

Libre à vous de cliquer et de consulter ce que bon vous semble, et de faire passer les liens à vos ami·e·s, collaborateurs et collaboratrices !

Une documentation qui n'attend que

vous

Bien entendu, l'ensemble de ces documents sont libres. Par défaut, la licence utilisée pour les productions Framasoft est la CC-BY-SA, mais prenez soin de vérifier pour chaque outil de documentation, car leurs auteurs et autrices peuvent tout à fait les avoir placé sous une autre licence libre ^^ !

C'est néanmoins une des grandes forces du Libre : n'importe qui peut y participer.

Vous cherchez à soutenir le (logiciel) libre sans forcément savoir coder ? Présentez votre service ou projet favori avec une petite vidéo, une présentation animée, un texte avec captures d'images… Nous vous l'assurons, cela aidera énormément de monde à passer le pas et à adopter du libre dans ses habitudes numériques.

Pour participer, deux cas de figure :

- Vous connaissez le *git*, les *push* et *pull request* ne vous font pas peur ? : rendez-vous sur la *forge* de notre *gitbook* pour proposer vos *commits* afin que l'on *merge* tout cela ensemble.
- Vous n'avez rien compris à la phrase ci dessus ? (ne vous inquiétez pas, celui qui l'a écrite est comme vous !) Rendez-vous sur notre forum des bénévoles, partie tutoriels, pour proposer vos tutos, vidéos, et autres trucs en -os !

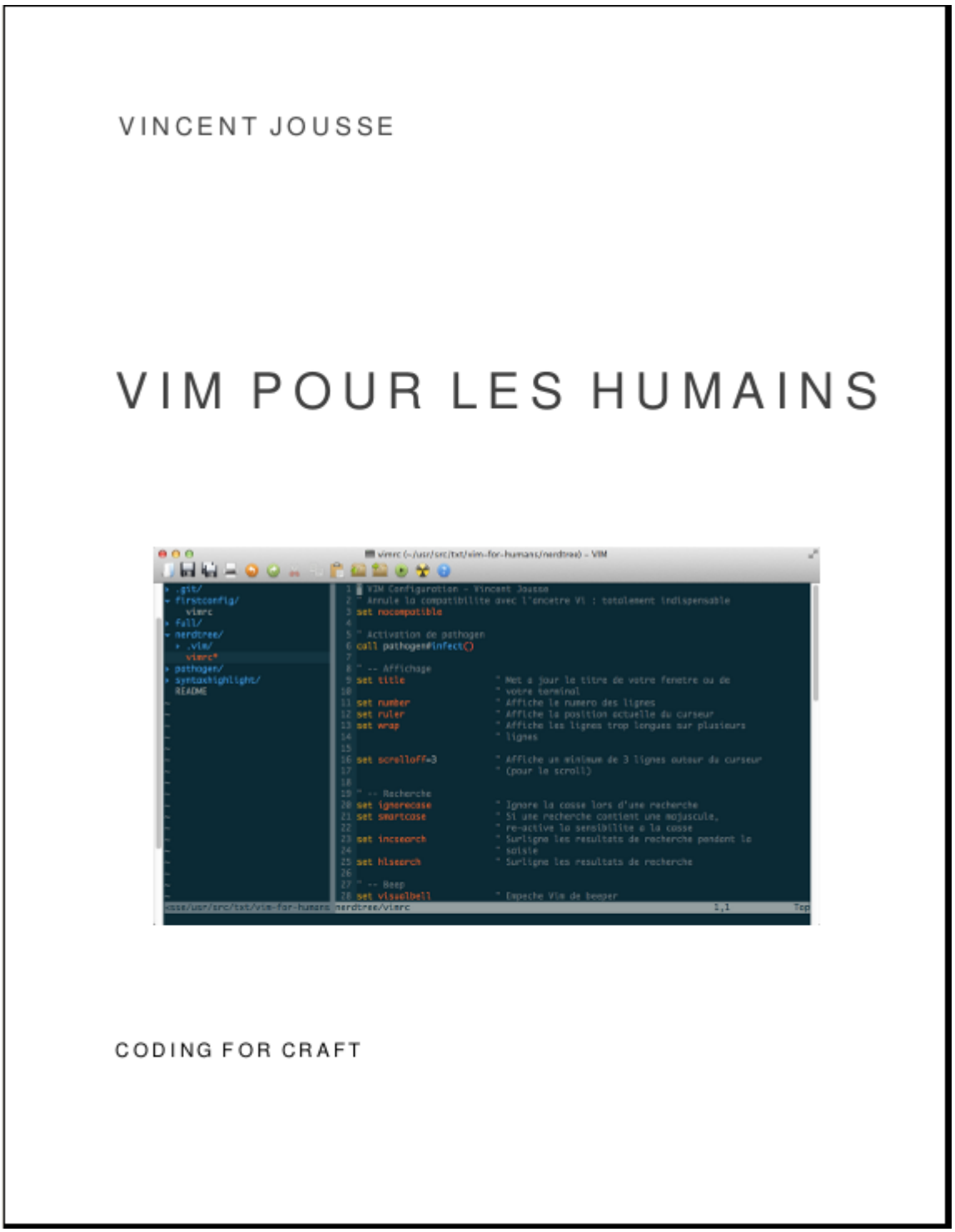
Enfin, une manière toute simple de participer, c'est simplement d'aller lire ces petits bouts de savoirs qui aident à mieux se dégoogliser… et de les partager avec son entourage !

Livre à prix libre : Vim pour les humains, de Vincent Jousse

Connaissez-vous et surtout utilisez-vous l'éditeur de texte libre Vim ?

Non ? C'est parce que vous n'avez pas encore lu l'interview ci-dessous et surtout n'avez pas encore parcouru le livre d'initiation « Vim pour les humains » mis librement à notre disposition par Vincent Jousse

Un livre électronique vendu à prix libre dont l'auteur a décidé d'en redistribuer 20% à Framasoft, merci à lui.



Bonjour Vincent, peux-tu te présenter succinctement à nos lecteurs ?

Vincent Jousse, 33 ans, libre et heureux ! J'écris régulièrement sur http://viserlalune.com au sujet de la vie en général et de tout ce qui peut la rendre encore plus sympathique. Je suis développeur informatique de formation et actuellement enseignant-chercheur à mi-temps à l'Université du

Maine où je travaille sur la reconnaissance de la parole. Je suis aussi gérant d'une entreprise dans le domaine : http://voxolab.com.

Je suis contributeur open source depuis pas mal d'années, d'abord en PHP (Symfony) et puis maintenant en Python. Je publie de temps en temps des articles un peu plus techniques sur http://vincent.jousse.org/.

Tous mes écrits sont placés sous licence CC, mon code source étant généralement publié sous licence MIT.

Alors le projet « Vim pour les humains » c'est quoi exactement ?

C'est tout d'abord l'envie d'essayer « autre chose ». À l'époque (déjà 2 ans !), j'avais envie de faire autre chose que passer mes journées à programmer. Comme j'aimais bien écrire, j'ai cherché un sujet sur lequel je pourrais apporter de la valeur. Je n'étais pas une star de Vim (et ne le suis toujours pas), mais je me rappelais de la difficulté que j'avais eue à franchir le premier pas, tout ça à cause d'une documentation réservée aux initiés.

Je trouvais ça dommage que, après des décennies d'existence, apprendre Vim (et surtout comprendre son intérêt) soit toujours aussi compliqué. L'objectif du projet est donc d'aider les utilisateurs à aller plus loin que la sempiternelle question : « mais comment on fait pour quitter cette m*** ?! ».

On peut lire ceci en ouverture du site : « Apprendre Vim est le meilleur investissement que j'aie jamais fait. Que ce soit en tant qu'écrivain, professeur ou programmeur : on l'apprend une fois, il nous suit partout, et pour toujours. » Ah oui, quand même !

Ça fait rêver non ? Le pire, c'est que c'est vrai. Tout ce que je produis l'est toujours à partir d'un fichier au format

texte : ma thèse en LaTeX, mes présentations en Markdown (via http://bartaz.github.io/impress.js/#/bored), mes articles de blog en Markdown, mon code source en Python, le livre « Vim pour les humains » en reStructuredText via Sphinx, j'en passe et des meilleurs.

Vim est le seul outil que j'utilise à longueur de journée, partout, tout le temps. Même les réponses de cette interview ont été écrites dans Vim.

Ton livre, c'est pour que Vim ne soit pas uniquement l'apanage des geeks barbus ?

C'est exactement ça. C'est peut-être mon côté utopiste, mais j'aime à croire que beaucoup de choses dans le monde du libre seraient plus accessibles si on prenait le temps de les expliquer et si on adoptait un état d'esprit plus pragmatique. Par exemple, tout le monde ne peut pas perdre un mois de productivité en se mettant à Vim. J'ai donc pris le parti de commencer mon livre en expliquant comment rendre Vim utilisable comme un éditeur classique.

Ça a fait grincer quelques dents du style : « c'est pas la bonne manière d'apprendre Vim », « faire croire que Vim est facile est une mauvaise idée », … Foutaises. Mon avis est que si les personnes l'utilisent encore une semaine après avoir commencé à l'apprendre, c'est gagné. Et pour ça, il faut leur faciliter la transition avec leur ancien éditeur.

Question troll : Alors, c'est sûr vi est meilleur qu'Emacs ?

Ahah, c'est même plus du troll à ce niveau là. Disons qu'avant « Vim pour les humains » ils étaient à égalité, maintenant Vim a clairement pris le dessus

Vi, Emacs, TeX… comment expliques-tu que ces éditeurs, conçus dans les années 1970, soient toujours utilisés aujourd'hui ?

C'est malheureux à dire mais, en ce qui concerne Vim et Emacs,

on n'a pas encore fait mieux. Ce sont des environnements qui ont été pensés avant l'arrivée de la souris et des interfaces graphiques. La majorité des efforts a donc porté sur la maximisation de l'efficacité au clavier, et il faut dire que Vim et Emacs sont redoutables dans ce domaine (c'est d'ailleurs ce qui les rend compliqués à apprendre à l'ère actuelle des clickodromes et autres interfaces graphiques).

Quand on édite du texte à longueur de journée (que ça soit du code ou autre chose), on cherche à être le plus rapide possible. Quitter les mains du clavier pour bouger la souris et retourner au clavier est aussi inefficace que mauvais pour vos mains (http://fr.wikipedia.org/wiki/Troubles_musculosquelettiques). À part Vim ou Emacs, aucune alternative sérieuse n'existe à ma connaissance pour l'instant.

En ce qui concerne Tex, c'est un peu pareil. Si l'on veut éditer un document en se focalisant sur le contenu et non la présentation, les langages de balises comme (La)Tex, Markdown, rst sont super adaptés. Si en plus on veut écrire un document avec une belle biblio générée automatiquement et des formules mathématiques avec un rendu impeccable c'est simple : LaTeX est la seule solution.

Ok, c'est pas libre, mais que penses-tu du succès actuel d'un éditeur comme Sublime Text ? Fait-il de l'ombre et du tort à Vim ?

En fait, peu importe qu'il fasse du tort ou de l'ombre à Vim, s'il répond à un besoin et que les personnes en sont contentes, parfait !

Je serai très content que Vim puisse être remplacé par quelque chose qui tire entièrement parti des spécificités des environnements graphiques et je pense qu'en effet, Sublime Text est un bon pas dans cette direction.

Mais même si Sublime Text venait à être utilisé partout et par

tout le monde, il resterait toujours un problème à résoudre : comment faire lorsque l'on n'a pas d'interface graphique ? « Je connais un truc sympa si tu veux, ça prend rien en ressources, tu peux l'utiliser en ligne de commande, Vim que ça s'appelle »

Pourquoi avoir opté pour la très libre licence CC-By ?

Parce que je fais une réaction épidermique à la connerie ambiante qui consiste à se « défendre des autres » coûte que coûte. Quelqu'un a une meilleure idée que moi pour valoriser ce travail ? Mais qu'il fasse donc, au contraire !

Je me suis basé sur Vim et sur toutes les contributions de la communauté open source pour écrire ce livre. Ce livre ne m'appartient pas, il appartient au bien commun, comme toutes les sources d'inspirations qui m'ont permis de l'écrire.

Le choix du « prix libre », c'est symbolique ou tu crois au devenir de ce modèle économique ? (et que penses-tu des « passagers clandestins » qui vont payer 0 euro pour télécharger le livre ?)

Non ce n'est pas symbolique, j'ai envie d'y croire. Le livre était tout d'abord téléchargeable au prix fixe de 9,99€. En 1 an et demi, j'en ai vendu 90 environ, pas si mal. Mais en faisant ça, j'allais contre mes valeurs, je limitais l'accès à la connaissance à cause de ce prix fixe.

J'ai donc décidé de le mettre sous un « prix libre », au risque en effet de ne plus rien gagner. Je suis prêt à prendre ce risque car, utopiste ou non, je pense que tout est dans l'art de demander. Faire la différence entre « gratuit » et « payant mais sous un prix libre » est important pour moi.

Les personnes qui payent 0 euro pour télécharger le livre et ne donnent rien en échange (même pas un petit mail) le feront en leur âme et conscience. J'ai ma conscience pour moi

Pourquoi uniquement une version électronique du livre ? C'est pour ne pas tuer d'arbres ?

Du tout, c'est juste par méconnaissance du monde du livre papier, tout simplement. Ça pourrait d'ailleurs être une bonne idée de monétisation. Des amateurs ?

Une dernière question qui nous flatte et nous t'en remercions, pourquoi avoir choisi de redistribuer 20% des gains à Framasoft ?

Pour être honnête, j'ai fais ça au feeling. Je ne vous connais pas très très bien, mais j'ai tendance à vous voir un peu partout autour des projets et personnes que j'apprécie : Ploum, Wallabag, Geektionnerd, … J'aime les personnes qui aident à faire bouger les choses et vous en faites partie, alors merci ! « La route est longue mais la voie est libre… »

Framapad : 3 tutoriels vidéos en situation d'éducation

Il y a un peu moins d'un an nous faisions le constat que Framapad était de plus en plus souvent utilisé dans l'éducation. En illustrant cela avec, entre autres, un billet sur l'expérience de Frédéric Véron, professeur de SVT dans l'Académie de Créteil.

Il nous propose ici, merci à lui, 3 tutoriels vidéos récemment mis en ligne.

Découverte et utilisation de Framapad

(pads publics)

Comparaison et avantages de la création d’un compte (pads privés)

Attention : La création de compte risque d’être bientôt suspendue parce que non maintenue, nous vous invitons plutôt à utiliser les pads publics.

Exemple de travail collaboratif avec des élèves de Sixième

---

13 points que les gens détestent sur la documentation de votre projet libre

Qu’il s’agisse de son code ou de son utilisation, la faiblesse de la documentation d’un logiciel libre est souvent montrée du doigt.

Voici, selon Andy Lester, 13 défauts ou lacunes communément rencontrés, qui sont autant d’écueils que l’on peut contourner avec un minimum d’efforts aujourd’hui pour gagner demain un

temps précieux.

13 choses que les gens détestent sur vos documentations open source

13 Things People Hate about Your Open Source Docs

Andy Lester – 10 janvier 2013 – SmartBear Blog
(Traduction : Lamessen, calou, Shanx, sinma, Asta + anonymes)

La plupart des développeurs open source aiment penser à la qualité du logiciel qu'ils développent, mais la qualité de la documentation est souvent laissée de côté. Il est rare de voir vanter la documentation d'un projet, et pourtant elle a un impact direct sur sa réussite. Sans une bonne documentation, les utilisateurs n'utiliseront pas votre projet, ou ils n'y prendront pas de plaisir. Les utilisateurs comblés sont ceux qui diffusent des infos à propos de votre projet – ce qu'ils ne font qu'après avoir compris comment il fonctionne. Et ils

apprennent cela à partir de la documentation du projet.

Malgré tout, de trop nombreux projets ont une documentation décevante. Et cela peut être décevant de plusieurs manières.

Les exemples que je donne ci-dessous sont purement arbitraires, je ne veux pas cibler un projet en particulier. Ce sont seulement ceux que j'ai utilisés récemment, cela ne veut pas dire qu'ils représentent les pires atrocités. Chaque projet a commis au moins quelques-uns de ces péchés. Que vous soyez utilisateur ou développeur, à vous d'évaluer à quel point votre logiciel préféré est ou non coupable, et comment vous pouvez aider à y remédier le cas échéant.

1. Le manque d'une bonne introduction ou d'un README/LISEZ-MOI

Le README/LISEZ-MOI est la première impression que les utilisateurs potentiels ont de votre projet. Si le projet est sur GitHub, le README/LISEZ-MOI est automatiquement affiché sur la page d'accueil du projet. Si vous l'avez mal rédigé, ils peuvent ne jamais revenir.

Vous voulez capter l'attention du lecteur et l'encourager à continuer la découverte de votre projet ? Le README/LISEZ-MOI devrait alors au moins expliquer :

- ce que le projet fait
- pour qui il est fait
- sur quel matériel ou plateforme il tourne
- toutes les dépendances majeures, comme « Requiert Python 2.6 et libxml »
- comment l'installer, ou un accompagnement de chaque étape à la suivante.

Tout cela doit pouvoir être compris par quelqu'un qui n'a jamais entendu parler de votre projet, et peut-être même jamais imaginé un projet pouvant s'en rapprocher. Si le projet possède un module calculant la distance de Levenshtein, ne

partez pas du principe que n'importe qui lisant votre README/LISEZ-MOI sait ce que c'est. Expliquez que la distance de Levenshtein est utilisée pour comparer deux chaînes de caractères, et ajoutez quelques renvois vers des explications plus poussées pour celui qui aimerait approfondir le sujet.

Ne décrivez pas votre projet par rapport à un autre projet, comme « NumberDoodle est comme BongoCalc, mais meilleur ! » Ça n'est d'aucune aide pour quelqu'un qui n'a jamais entendu parlé de BongoCalc.

2. La documentation non disponible en ligne

Bien que je n'ai pas lu d'études à ce sujet, je serais prêt à parier que 90% des recherches de documentation sont faites avec Google et un navigateur sur Internet. La documentation de votre projet doit être en ligne, et disponible. Partant de là, il serait embarrassant que la documentation de mon propre projet, ack, ne soit pas disponible à l'endroit où la majorité des gens vont la chercher. Mon hypothèse est basée sur ma propre expérience, à savoir que si je veux connaître le fonctionnement d'un outil en ligne de commande, je vais vérifier sa page man.

Comment je m'en suis aperçu ? Les utilisateurs m'écrivaient pour me poser des questions dont les réponses se trouvaient dans la FAQ. Ce qui m'a ennuyé : ils ne lisaient pas ma FAQ. Il se trouve qu'ils avaient cherché sur le site internet, mais je n'avais pas mis la FAQ à cet endroit. C'est une erreur facile à faire. Je suis proche du projet et je n'ai jamais eu besoin d'utiliser moi-même la FAQ, je n'avais donc pas remarqué qu'elle n'était pas présente en ligne. Beaucoup de problèmes sont dus à ce piège : les auteurs ne se mettent pas à la place des utilisateurs.

3. La documentation disponible uniquement en ligne

Le revers de ce problème est d'avoir la documentation disponible uniquement en ligne. Certains projets ne

distribuent pas la documentation avec les livrables du projet, ou incluent une version médiocre de la documentation.

Le moteur de recherche Solr, par exemple, a un excellent wiki qui sert à la documentation du projet. Malheureusement, la documentation liée au téléchargement comporte 2200 pages de Javadoc d'API auto-générées. Au final, la seule documentation pour l'utilisateur est une unique page de tutoriel.

Le langage PHP n'est distribué avec aucune documentation. Si vous voulez la documentation, vous devez aller sur une page séparée pour les obtenir. Pire, seule la documentation du cœur est disponible au téléchargement, sans les annotations utiles des utilisateurs (voir « Ne pas accepter les remarques des utilisateurs » plus bas), et ce n'est pas le même format facile à parcourir que celui qui est disponible en ligne.

Les projets open source ne peuvent pas supposer que les utilisateurs ont accès à Internet quand ils ont besoin de la documentation. Le mode avion existe toujours. De toute façon, vous ne souhaitez pas que l'utilisateur dépende uniquement du fait que votre site web est disponible ou non. Au moins à deux reprises durant les derniers mois, le wiki de Solr était indisponible au beau milieu de ma journée de travail alors que je recherchais des informations sur un problème de configuration épineux.

Un projet qui fait les choses bien est Perl et son dépôt de module CPAN. La documentation pour chaque module est disponible soit à search.cpan.org ou metacpan.org dans un format hypertexte facile à lire. Pour la consultation hors-ligne, la documentation de chaque module est intégrée dans le code lui-même, et quand le module est installé sur le système d'un utilisateur, la documentation locale est créée sous forme de pages man. Les utilisateurs peuvent aussi utiliser « perldoc Module::Name » pour obtenir la documentation depuis le shell. En ligne ou hors-ligne : c'est votre choix.

4. La documentation non installée avec le paquet

Ce problème est généralement une erreur des paquageurs, pas des auteurs du projet. Par exemple, sur Ubuntu Linux, la documentation du langage Perl est séparée, ce sont des paquets optionnels pour le langage lui-même. L'utilisateur doit savoir qu'il doit explicitement installer la documentation de la même façon que le langage principal ou il n'y aura pas accès quand il en aura besoin. Ce compromis de quelques mégabites d'espace disque au détriment de la documentation à portée de main de l'utilisateur dessert tout le monde.

5. Le manque de captures d'écran

Il n'y a pas de meilleur moyen d'obtenir l'attention potentielle d'un utilisateur, ou d'illustrer un usage correct, qu'avec des captures d'écran judicieuses. Une image vaut mieux qu'un long discours, c'est encore plus important sur Internet parce que vous ne pouvez obtenir d'un lecteur de lire plus de quelques centaines de mots en tout.

Les captures d'écran accompagnant le texte sont inestimables pour guider l'utilisateur voulant faire les choses au mieux. Une capture d'écran lui permet de comparer visuellement ses résultats à ceux de la documentation et va le rassurer d'avoir exécutée la tâche correctement ou l'aidera à trouver facilement ce qui ne va pas.

Il est de plus en plus commun de trouver des vidéos sur le site internet d'un projet pour en donner un aperçu, et c'est génial. Tout autant que le fait d'avoir une vidéo pour chaque étape d'un processus complexe. Le projet Plone, par exemple, a un site entier dédié aux tutoriels vidéos. Cependant, les vidéos ne peuvent pas remplacer les captures d'écran. Un utilisateur veut voir rapidement l'allure des captures d'écran sans s'arrêter devant une vidéo. Les vidéos n'apparaissent également pas dans une recherche *Google Image*, à l'inverse des captures d'écran.

6. Le manque d'exemples réalistes

Pour les projets basés sur du code, l'analogue des captures d'écran sont de bons et solides exemples du code en action. Ces exemples ne devraient pas être abstraits, mais directement issus du monde réel. Ne créez pas d'exemples bateaux plein de « nom de la démo ici » et lorem ipsum. Prenez le temps de créer des exemples signifiants avec une histoire d'utilisateur qui représente la façon dont votre logiciel résout un problème.

Il y a de bonnes raisons de vous embêter avec des problèmes de maths en classe. Ils permettent d'appliquer ce que vous avez appris.

Disons que j'ai écrit un module d'un robot Web, et que j'explique la méthode follow_link. Je pourrais montrer la définition de la méthode ainsi :

```
$mech->follow_link( text_regex => $regex_object, n => $link_index );
```

Mais admirez à quel point cela devient évident en ajoutant de la réalité dans l'exemple.

```
# Suit le 2e lien où la chaîne de caractères « download » apparait
$mech->follow_link( text_regex => qr/download/, n => 2 );
```

Les noms des variables $regex_object et $link_index sont maintenant compréhensibles par le lecteur.

Bien entendu, vos exemples ne doivent pas être aussi brefs. Comme Rich Bowen du projet Apache le souligne, « Un exemple correct, fonctionnel, testé et commenté l'emporte sur une page de prose, à chaque fois. »

Montrez autant que possible. L'espace n'est pas cher. Créez une section dédiée aux exemples dans la documentation, ou même un livre de cuisine. Demandez aux utilisateurs d'envoyer du

code qui fonctionne, et publiez leurs meilleurs exemples.

7. Liens et références inadéquats

Vous avez les hyperliens. Utilisez-les.

Ne pensez pas, parce que quelque chose est expliquée dans une certaine partie de la documentation, que le lecteur a déjà lu cette partie, ou bien qu'il sait où elle se trouve. Ne vous contentez pas de signaler que cette partie du code manipule des objets frobbitz. Expliquez brièvement lors du premier usage de ce terme ce qu'est un objet frobbitz, ou donnez le lien vers la section du manuel l'expliquant. Encore mieux, faites les deux !

8. Oublier les nouveaux utilisateurs

Il arrive trop souvent que l'écriture de la documentation soit rédigée à partir du point de vue de son auteur, alors que es nouveaux utilisateurs ont besoin de documentation d'introduction pour les aider.

L'introduction devrait être une page séparée de la documentation, idéalement avec des exemples qui permettent à l'utilisateur de réussir quelques manipulations avec le logiciel. Pensez à l'excitation que vous ressentez quand vous commencez à jouer avec un nouveau logiciel et qu'il vous permet de faire quelque chose de cool. Faites que ça arrive aux nouveaux utilisateurs également.

Par exemple, un package graphique pourrait présenter une série de captures d'écran qui montrent comment ajouter des données dans un fichier, comment faire intervenir le grapheur, et ensuite montrer les graphes obtenus. Une bibliothèque de codes pourrait montrer quelques exemples d'appels à la bibliothèque, et montrer le résultat obtenu. Pensez simplicité. Offrez une victoire facile. Le texte devrait introduire les termes aux endroits appropriés, avec des liens vers une documentation plus détaillée sur le long terme.

Un document de démarrage séparé donne à l'utilisateur une compréhension rapide du logiciel. Il garde aussi les explications d'introduction en dehors de la partie principale de votre documentation.

9. Ne pas écouter les utilisateurs

Les développeurs doivent écouter les utilisateurs de la documentation. La chose évidente est d'écouter les suggestions et requêtes des personne qui utilisent activement votre logiciel. Quand un utilisateur prend le temps d'écrire un mail ou de poster quelque chose comme « ça aurait pu m'aider à mieux installer le programme s'il y avait eu une explication ou des liens au sujet des pilotes de la base de données », prenez ce message au sérieux. Pour chaque utilisateur vous envoyant un mail pour un problème, vous devez vous attendre à ce que dix utilisateurs silencieux aient le même problème.

Il est important d'écouter les problèmes des utilisateurs et d'en chercher les causes. S'ils ont souvent des problèmes pour effectuer des mises à jour groupées de bases de données, la première chose à faire est d'ajouter une question à la FAQ (vous avez bien une FAQ, n'est-ce pas ?) qui traite de ces questions-là. Cependant, la question peut aussi indiquer que la section traitant des mises à jour de base de données n'est pas assez claire. Ou peut-être qu'il n'y a pas de référence à cette section depuis la vue d'ensemble introductive du document, avec pour conséquence que vos utilisateurs ne pensent jamais à lire le reste du manuel.

En plus d'aider plus de gens à découvrir à quel point votre projet est utile, ça diminuera aussi la frustration de la communauté déjà existante. Si votre liste de diffusion, forum ou canal IRC est remplie de personnes qui posent toutes les mêmes questions idiotes (ou pas si idiotes) au point que tout le monde devient lassé d'y répondre, sachez reconnaître que ce sont des questions récurrents pour la FAQ, et mettre les réponses à un endroit facile à trouver aidera tout le monde à

se concentrer sur des choses plus amusantes.

Gardez aussi un œil sur les questions des forums externes. Consultez les sites comme StackOverflow régulièrement, et placez une Google Alert sur votre nom de projet pour être maintenu au courant des discussions concernant votre projet sur Internet.

10. Ne pas accepter les entrées des utilisateurs

Si votre projet a une base d'utilisateur assez grande, il peut être judicieux d'incorporer les commentaires des utilisateurs directement dans la documentation. Le meilleur exemple que j'ai pu voir est celui donné par PHP. Chaque page de la documentation permet aux utilisateurs authentifiés d'ajouter des commentaires sur la page, aidant ainsi à clarifier certains points ou ajoutant des exemples qui ne sont pas dans la documentation principale. L'équipe PHP laisse aussi le choix au lecteur de lire la doc avec ou sans les commentaires des autres utilisateurs.

Aussi pratique cela soit-il, cela nécessite de la maintenance. Les commentaires doivent être éliminés de temps en temps pour éviter la prolifération. Par exemple, la page de la documentation PHP sur comment lancer PHP depuis la ligne de commande inclut 43 commentaires d'utilisateurs qui remontent à 2001. Les commentaires écrasent la documentation principale. Les commentaires devraient être archivés ou supprimés, tout en incluant les points les plus importants dans la documentation principale.

Un wiki est également une bonne approche. Cependant, si votre wiki ne permet pas à l'utilisateur de télécharger toutes les pages en une seule grosse archive (cs. point n°3 ci-dessus), alors vos utilisateurs sont à la merci de votre connexion internet et du serveur hébergeant le projet.

11. Impossibilité de voir ce que fait le logiciel sans l'installer

Au minimum, chaque projet de logiciel nécessite une liste de fonctionnalités et une page de captures d'écran pour permettre au potentiel utilisateur intéressé de savoir pourquoi il devrait l'essayer. Aidez l'utilisateur, comparant les différents paquets à utiliser, à voir pourquoi cela vaut la peine de prendre le temps de le télécharger et de l'installer.

Les images sont un bon moyen de faire cela. Votre projet devrait avoir une page « Captures d'écran » qui montre des exemples de l'outil en action (cf. point n°5 ci-dessus). Si votre projet se résume uniquement à du code, comme une librairie, alors il devrait y avoir une page d'exemples montrant ce code utilisant le projet.

12. S'appuyer sur la technologie pour votre rédaction

Trop souvent, les auteurs de logiciels utilisent des systèmes de documentation automatisés pour faire leur travail. Ce système automatisé rend les choses plus facile à maintenir, mais il ne supprime pas la nécessité d'un travail d'écriture humain.

Le pire des cas concerne le changelog, qui n'est rien de plus qu'un dump des messages de commit du système de gestion de version, mais sans un résumé qui l'explique. Un changelog devrait lister les nouvelles fonctionnalités, les problèmes résolus et les incompatibilités potentielles. Sa cible est l'utilisateur final. Un log de commit est pratique et simple à générer pour les personnes travaillant sur le projet, mais ce n'est pas ce dont l'utilisateur a besoin.

Jetez un œil à cette page de la documentation de Solarium, une interface PHP pour le moteur de recherche Solr. Tout d'abord, l'avertisemment prend la moitié supérieure de l'écran, ne donnant aucune information au lecteur. Ensuite, il n'y a

vraiment rien de véritablement descriptif sur la page que la liste des noms de fonctions. Il n'y a aucune explication sur les différentes méthodes, ni de liens indiquant où trouver l'explication. Les pages générées automatiquement sont jolies, et elles pourraient ressembler à de la documentation, mais elles n'en sont pas.

13. Arrogance et hostilité vis-à-vis de l'utilisateur

L'attitude du type RTFM (Read The Freaking Manual) est mauvaise pour votre projet et votre documentation.

C'est le summum de l'arrogance que de croire que tous les problèmes qui ont trait au fait que quelqu'un ne sache pas utiliser votre logiciel sont de la faute de l'utilisateur.

Même s'il est probablement vrai que les utilisateurs peuvent trouver leurs réponses dans votre documentation mais ne le font pas, il est stupide de penser que c'est la faute de l'utilisateur. Peut-être votre documentation est-elle mal écrite, ou difficile à lire, ou présente mal à l'écran. Peut-être avez-vous besoin d'améliorer la section « Mise en route » (lien #8 ci-dessus) qui explique ce que le logiciel a pour but de faire. Peut-être que certaines parties d'information nécessitent d'être répétées à de multiples endroits de la documentation.

N'oubliez pas que les nouveaux utilisateurs de votre logiciel peuvent arriver sur votre projet sans rien n'en savoir. Votre documentation doit faire de son mieux pour s'assurer que cette ignorance soit facilement résolue.

Synthèse

Je suis sûr que vous avez déjà eu affaire à quelques-uns de ces problèmes listés ci-dessous, et peut-être que pour d'autres vous n'y avez pas pensé. Faites-nous connaître les problèmes que vous avez rencontrés dans les commentaires ci-

dessous, sachant qu'il ne s'agit pas de pointer du doigt certains projets en particulier.

Surtout, j'espère que si vous reconnaissez un problème dans la documentation de vos projets, vous prendrez la peine d'améliorer la situation. Heureusement, améliorer la documentation est une manière idéale de faire participer les nouveaux arrivants dans votre projet. On me demande souvent : « Comment puis-je commencer dans l'open source », et je recommande des améliorations dans la documentation comme une bonne manière de commencer.

Faites-en sorte que ce soit aussi facile que possible, pour les novices comme les plus anciens, de savoir où il est nécessaire de travailler la documentation. Créez une liste des tâches, par exemple dans votre système de suivi des bogues, qui explique ce qui a besoin d'être amélioré. Soyez précis dans ce que sont vos besoins. Ne vous contentez pas de dire que vous avez besoin d'exemples, sans plus de précision. Créez des tâches spécifiques, comme « ajoutez un code d'exemple sur le fonctionnement de la tâche X », « ajouter une capture d'écran du générateur de rapports » ou « ajouter des informations de dépendances au fichier README/LISEZ-MOI ». Les contributeurs souhaitent aider mais trop souvent ils ne savent pas par où commencer.

La documentation n'est pas la partie la plus glamour d'un projet open source, et pour la plupart d'entre nous ce n'est pas amusant. Mais sans une bonne documentation, les utilisateurs ne sont pas servis comme ils pourraient l'être, et votre projet en souffrira sur le long terme.

Crédit photo : Rosalux Stiftung (Creative Commons By)

Apprendre à déléguer (Libres conseils 19/42)

Chaque jeudi à 21h, rendez-vous sur le framapad de traduction, le travail collaboratif sera ensuite publié ici même.

Traduction Framalang : Nyx, lamessen, Sphinx, peupleLà, lerouge, Sky, Julius22, Astalaseven, Alpha, HgO, michel, Sputnik, goofy, HanX, KoS

Ne vous inquiétez pas, faites confiance

Shaun McCance

Shaun McCance est impliqué dans la documentation de GNOME depuis 2003 en tant que rédacteur, chef de la communauté et développeur d'outils. Il a passé la plupart de ce temps à se demander comment inciter davantage de personnes à écrire une documentation de meilleure qualité, avec un certain succès à long terme. Il propose son expérience de la documentation communautaire à travers sa société de conseil, Syllogist.

Alors que je m'apprêtais à écrire cet article, il s'est passé quelque chose d'énorme : GNOME 3 est sorti. C'était la première version majeure de GNOME depuis neuf ans. Tout était différent et toute la documentation existante devait être réécrite. Au même moment, nous changions notre façon de l'écrire. Nous avions jeté nos vieux manuels et étions repartis sur une nouvelle base, avec un système d'aide dynamique par sujet, en utilisant Mallard.

Quelques semaines avant la sortie, une partie d'entre nous s'est réunie pour élaborer la documentation. Nous passions nos journées à travailler, à planifier, à écrire et à réviser.

Nous avons écrit des centaines de pages malgré les changements incessants liés aux ultimes modifications du logiciel. Nous avions des contributeurs en ligne qui proposaient de nouvelles pages et corrigeaient ce qui existait déjà. Je n'avais jamais vu notre équipe de documentation aussi productive.

À quoi avons-nous finalement abouti ? Beaucoup de facteurs sont entrés en jeu, et je pourrais écrire un livre entier sur les nuances de la documentation *open source*. Mais ce que j'ai fait de plus important a été de m'effacer et de laisser les autres faire le travail. J'ai appris à déléguer ; et à déléguer dans les règles de l'art.

Revenons huit ans en arrière. J'ai commencé à m'impliquer dans la documentation de GNOME en 2003. Je n'avais pas vraiment d'expérience en tant que rédacteur technique à cette époque. Mon emploi m'amenait à travailler sur des outils de publication et j'ai commencé à travailler sur les outils et sur le visualiseur d'aide utilisés par la documentation de GNOME. Peu de temps après, je me suis retrouvé à la rédaction de la documentation.

En ce temps-là, la majeure partie de notre documentation était entre les mains de rédacteurs techniques professionnels de chez Sun. Ils s'occupaient d'un manuel, l'écrivaient, le relisaient et l'envoyaient sur notre dépôt CVS. Après quoi nous pouvions tous le regarder, y apprendre quelque chose et lui apporter des corrections. Mais il n'existait pas d'efforts concertés pour impliquer les gens dans le processus d'écriture.

Ce n'est pas que les rédacteurs de Sun essayaient de protéger ou de cacher quoi que ce soit. Ils étaient avant tout rédacteurs techniques. Ils connaissaient leur travail et le faisaient bien. D'autres personnes auraient pu les remplacer pour d'autres manuels mais ils auraient écrit leurs travaux d'une manière habituelle. Utiliser un groupe de collaborateurs novices, aussi enthousiastes soient-ils, pour chaque page,

revient à perdre un temps inimaginable sur des détails. C'est tout simplement contre-productif.

De manière inévitable, le vent a tourné chez Sun et leurs rédacteurs techniques ont été affectés à d'autres projets. Cela nous a laissés sans nos rédacteurs les plus prolifiques, ceux qui disposaient des meilleures connaissances. Pire que cela, nous étions laissés sans communauté et personne n'était là pour ramasser les morceaux.

Il y a des idées et des processus standards dans le monde de l'entreprise. J'ai travaillé dans le monde de l'entreprise. Je ne crois pas que quiconque remette ces idées en cause. Les gens font leur travail. Ils choisissent des missions et les terminent. Ils demandent aux autres de faire une relecture, mais ils n'ouvrent pas leur travail aux nouveaux venus et aux rédacteurs moins expérimentés. Les meilleurs rédacteurs écriront sans doute le plus.

Ces idées sont d'une plate évidence, mais elles échouent lamentablement dans un projet communautaire. Vous ne développerez jamais une communauté de contributeurs si vous faites tout vous-même. Dans un projet de logiciel, vous pouvez avoir des contributeurs compétents et suffisamment impliqués pour contribuer régulièrement. Dans la documentation, cela n'arrive presque jamais.

La plupart des gens qui s'essayent à la documentation ne le font pas parce qu'ils veulent être rédacteur technique ni même parce qu'ils aiment écrire. Ils le font parce qu'ils veulent contribuer. Et la documentation est la seule manière qu'ils trouvent accessible. Ils ne savent pas coder. Ils ne sont artistiquement pas doués. Ils ne maîtrisent pas assez une autre langue pour faire de la traduction. Mais ils savent écrire.

C'est là que les rédacteurs professionnels lèvent les yeux au ciel. Le fait que vous soyez instruit ne signifie pas que vous

puissiez écrire une bonne documentation pour l'utilisateur. Il ne s'agit pas simplement de poser des mots sur le papier. Vous devez comprendre vos utilisateurs, ce qu'ils savent, ce qu'ils veulent, les endroits où ils cherchent. Vous avez besoin de savoir comment présenter l'information de façon compréhensible et savoir où la mettre pour qu'ils puissent la trouver.

Les rédacteurs techniques vous diront que la rédaction technique n'est pas à la portée de tous. Ils ont raison. Et c'est exactement pourquoi la chose la plus importante que les rédacteurs professionnels puissent faire pour la communauté est de ne pas écrire.

La clé pour construire une communauté efficace autour de la documentation, c'est de laisser les autres prendre les décisions, faire le travail et en récolter eux-mêmes les fruits. Il ne suffit pas de leur donner du travail en continu. La seule solution pour qu'ils s'intéressent suffisamment et s'accrochent au projet, c'est qu'ils se sentent investis personnellement. Le sentiment de faire partie intégrante d'un projet est une source puissante de motivation.

Mais si vous ne travaillez qu'avec des rédacteurs débutants et que vous leur donnez tout le travail à faire, comment pouvez-vous avoir l'assurance que la documentation ainsi créée sera de qualité ? Une participation massive mais incontrôlée n'aboutit pas à de bons résultats. Le rôle d'un rédacteur expérimenté au sein de la communauté est d'être un professeur et un mentor. Vous devez leur apprendre comment rédiger.

Commencez par impliquer les gens tôt dans le planning. Planifiez toujours du bas vers le haut. La planification du haut vers le bas n'incite pas à la collaboration. Il est difficile d'impliquer les gens dans la réalisation d'une vue d'ensemble de haut niveau si tous n'ont pas la même perception de cette vue d'ensemble. Mais les gens sont capables de travailler sur des segments. Ils peuvent réfléchir à des sujets particuliers d'écriture, à des tâches que les gens

réalisent, à des questions que les gens peuvent se poser. Ils peuvent regarder les forums de discussion et les listes de diffusion afin de voir ce que les utilisateurs demandent.

Écrivez vous-même quelques pages. Cela donne un exemple à imiter. Il faut ensuite répartir tout le reste du travail. Laissez à d'autres la responsabilité de rubriques ou de chapitres entiers. Précisez-leur clairement quelles informations ils doivent fournir, mais laissez-les écrire. C'est en forgeant qu'on devient forgeron.

Soyez constamment disponible pour les aider ou répondre aux questions. Au moins la moitié de mon temps consacré à la documentation est passée à répondre à des questions afin que les autres puissent effectuer leur travail. Quand des brouillons sont soumis, relisez-les et discutez des critiques et des corrections avec leurs auteurs. Ne vous contentez pas de corriger vous-même.

Cela vous laisse tout de même le gros du travail à faire. Les gens complètent les pièces du puzzle, mais c'est toujours vous qui les assemblez. Au fur et à mesure qu'ils acquièrent de l'expérience, les gens s'occuperont de pièces de plus en plus grandes. Encouragez-les à s'impliquer davantage. Donnez-leur davantage de travail. Faites en sorte qu'ils vous aident à aider plus de rédacteurs. La communauté fonctionnera toute seule.

Huit ans plus tard, GNOME a réussi à créer une équipe de documentation qui se gère elle-même, résout les problèmes, prend des décisions, produit une bonne documentation et accueille constamment de nouveaux contributeurs. N'importe qui peut la rejoindre et y jouer un rôle. Telle est la clé du succès pour une communauté *open source*.

Documenter c'est s'enrichir (Libres conseils 18/42)

Chaque jeudi à 21h, rendez-vous sur le framapad de traduction, le travail collaboratif sera ensuite publié ici même.

Traduction Framalang : lamessen, lerouge, Kalupa, Sky, Astalaseven, Alpha, LuD-up, CoudCoud, peupleLa, goofy

La documentation et moi, avant et après

Anne Gentle

Anne Gentle est une rédactrice technique acharnée et la coordinatrice de la documentation communautaire à Rackspace pour OpenStack, un projet open source d'informatique dans le nuage. Avant de rejoindre OpenStack, Anne travaillait en tant que consultante de publication communautaire, en donnant une direction stratégique aux rédacteurs professionnels qui veulent produire du contenu en ligne sur des wikis contenant des pages et des commentaires générés par les utilisateurs. Sa passion pour les méthodes communautaires de documentation l'a amenée à écrire un livre sur les techniques de publication collaborative pour la documentation technique intitulé Conversation et communauté : la documentation dans le web social. Elle s'occupe aussi bénévolement de la maintenance de la documentation pour les manuels FLOSS qui proposent de la documentation open source pour les projets open source.

Voilà une prémisse bien étrange : vider mes tripes sur ce que j'aurais voulu savoir de l'*open source* et de la documentation. Plutôt que de vous dire ce que je veux que vous sachiez sur l'*open source* et la documentation, je dois vous dire ce que

j'aurais aimé que mon moi antérieur sache. Cette demande suscite un sentiment de nostalgie ou de remords voire cette horrible énigme : « à quoi pouvais-je bien penser ? ».

En ce qui me concerne, avec juste cinq ans de moins, mon moi antérieur était une trentenaire bien installée dans sa vie professionnelle. D'autres, au contraire, se souviennent qu'ils n'étaient qu'adolescents lors de leurs premières expériences *open source*. Jono Bacon dans son livre *L'art de la Communauté*, raconte comment il s'est tenu devant la porte d'un appartement, le cœur battant, alors qu'il était sur le point de rencontrer quelqu'un à qui il n'avait jamais parlé que sur le réseau, par le biais d'une communauté *open source*. J'ai moi aussi fait cette expérience de la première rencontre physique de gens que j'avais découverts en ligne, mais ma première incursion dans le monde de la documentation *open source* s'est produite lorsque j'ai répondu à une demande d'aide par courriel.

Le courriel provenait d'un ancien collègue qui me demandait de l'aide pour la documentation de l'ordinateur portable XO, le projet fondateur de l'organisation *One Laptop Per Child* (un portable pour chaque enfant). J'ai réfléchi à ce que je pensais être une proposition intéressante, j'en ai parlé à mes amis et à mon époux en me demandant si ce n'était pas une bonne occasion d'expérimenter de nouvelles techniques de documentation et d'essayer une chose que je n'avais jamais faite auparavant : une documentation basée sur un wiki. Depuis cette première expérience, j'ai rejoint *OpenStack*, un projet *open source* sur une solution d'informatique dans le nuage, et je travaille à plein temps sur la documentation et le support communautaires.

Je pense immédiatement aux nombreuses contradictions que j'ai rencontrées tout au long de la route. J'ai découvert que pour chaque observation il existe des champs et contre-champs surprenants. Par exemple, la documentation est absolument indispensable pour l'aide aux utilisateurs, l'éducation, la

possibilité de choisir ou bien la documentation promotionnelle qui amène à adopter un logiciel. Pourtant, une communauté *open source* pourra continuer d'avancer malgré le manque de documentation ou de la doc complètement défectueuse. Voici une autre collision apparente de valeurs : la documentation pourrait être une très bonne tâche pour démarrer, un point de départ pour les nouveaux volontaires, pourtant les nouveaux membres de la communauté savent si peu de choses qu'il ne leur est pas possible d'écrire ni même d'éditer de manière efficace. En outre les petits nouveaux ne sont pas bien familiers des différents publics auxquels doit servir la documentation.

Ces derniers temps on entend dire un peu partout : « les développeurs devraient écrire la doc de développement » parce qu'ils connaissent bien ce public et par conséquent cela lui serait aussi utile qu'à eux-mêmes. D'après mon expérience, un regard frais et neuf est toujours le bienvenu sur un projet et certaines personnes ont la capacité d'écrire et de partager avec d'autres ce regard frais et empathique. Mais vous n'allez sûrement pas vous mettre à créer une culture « réservée aux novices » autour de la doc parce qu'il est important que des membres essentiels de la communauté technique apportent leur contribution aux efforts de documentation, et qu'ils encouragent aussi les autres à y participer.

Une partie du vilain petit secret sur la documentation des projets *open source* est qu'il n'existe qu'une frontière pour le moins floue entre leur documentation officielle et leur documentation officieuse. Si seulement j'avais su que les efforts de documentation devraient être sans cesse renouvelés et que de nouveaux sites web pourraient apparaître là où il n'y en avait pas… Une documentation extensive n'est pas le moyen le plus efficace pour s'initier à des projets ou des logiciels mais un parcours sinueux dans les méandres de la documentation sur le Web peut s'avérer plus instructif pour ceux qui veulent lire entre les lignes et avoir ainsi une idée

de ce qui se passe dans la communauté grâce à la documentation. Avoir beaucoup de forks(1) et des publics variés peut indiquer que le produit est complexe et qu'il est très suivi. Cela peut aussi signifier que la communauté n'a mis en place aucune pratique quant à la documentation de référence ou que les efforts désorganisés sont la norme.

À mes débuts, j'aurais aimé avoir la capacité de ressentir la « température conviviale » d'une communauté en ligne. Quand vous entrez dans un restaurant rempli de tables nappées de blanc, de couples qui dînent et de conversations feutrées, l'information visuelle et auditive que vous recevez détermine l'ambiance et vous donne quelques indices sur ce que vous vous apprêtez à vivre lors de votre repas. Vous pouvez tout à fait transposer ce concept de température conviviale à une communauté en ligne.

Une communauté *open source* vous donne quelques indices dans ses listes de discussion, par exemple. Une page de présentation de la liste qui commence par de nombreuses règles et conventions sur la manière de poster indique une gouvernance très stricte. Une liste de discussion qui contient de nombreuses publications mettant l'accent sur le fait qu'il « n'y a pas de question bête » est plus accueillante pour de nouveaux rédacteurs de documentation.

J'aurais aussi aimé connaître un moyen de faire non seulement de l'audit de contenu, c'est-à-dire lister le contenu à disposition pour le projet open source, mais aussi de l'audit de communauté, donc lister les membres influents de la communauté open source, qu'il s'agisse ou non de contributeurs.

Pour terminer, une observation à propos de l'*open source* et de la documentation que j'ai pu vérifier avec plaisir, c'est l'idée que la rédaction de la documentation peut s'effectuer via des « sprints » – grâce à des de brusques dépenses d'énergie avec un public ciblé et un but précis, pour aboutir

à un ensemble documentaire de référence.

J'ai été très contente d'entendre lors d'une conférence à SXSW Interactive que les sprints sont tout à fait acceptables pour la collaboration en ligne, qu'il faut s'attendre à des fluctuations du niveau d'énergie de chacun et que c'est OK. La documentation des logiciels est souvent faite à l'arrache dans les moments d'accalmie d'un cycle de release(2) et ça ne pose aucun problème dans la documentation *open source* qui est basée sur la communauté. Vous pouvez adopter une approche stratégique et coordonnée et continuer tout de même de proposer des évènements de grande intensité autour de la documentation. Ce sont des moments exaltants dans l'*open source* et mon ancien moi les a pleinement ressentis. C'est une bonne chose que vous continuiez à passer de votre moi antérieur à votre moi actuel avec un paquet de conseils en poche.

(1) Un *fork* est un nouveau logiciel créé à partir du code source d'un logiciel existant.

(2) La *release* est la publication d'un logiciel.

RTFM ? — mais où est-il, votre manuel ? (Libres conseils 16/42)

Chaque jeudi à 21h, rendez-vous sur le framapad de traduction, le travail collaboratif sera ensuite publié ici même.

Traduction Framalang : lamessen, Sputnik, lerouge, RavageJo, Sky, Astalaseven, goofy, KoS, peupleLa + Julius22

Une bonne documentation change la vie des débutants

Atul Jha

Atul Jha utilise des logiciels libres depuis 2002. Il travaille en tant que spécialiste des applications au CSS Corp, à Chennai en Inde. Il aime parcourir les universités, rencontrer des étudiants et propager la bonne parole du logiciel libre.

En 2002, on naviguait sur Internet dans des cybercafés en raison du coût important des accès par lignes commutées. À l'époque, la messagerie instantanée de Yahoo était très populaire et j'avais pris l'habitude de discuter sur le canal #hackers. Il y avait quelques fous furieux là-dedans qui prétendaient qu'ils pouvaient pirater mon mot de passe. J'étais très impatient d'en savoir plus sur le piratage et de devenir l'un d'eux. Le lendemain, je suis retourné au cybercafé et j'ai tapé « comment devenir un hacker » sur le moteur de recherche Yahoo. La toute première URL dirigeait sur le livre d'Eric S. Raymond. J'étais fou de joie à l'idée d'entrer dans le cercle des initiés.

J'ai commencé à lire le livre et à ma grande surprise la définition de hacker était « quelqu'un qui aime résoudre les problèmes et dépasser les limites ». Il y est également écrit : « les hackers construisent des choses, les casseurs les brisent »(1). Hélas, je cherchais le côté obscur, celui des crackers, et ce livre m'a mené de l'autre côté de la force, celui des hackers. J'ai continué à lire le livre et à rencontrer divers termes nouveaux tels que GNU/Linux, liste de diffusion, groupe d'utilisateur Linux, IRC, Python et bien d'autres encore.

En poursuivant mes recherches, j'ai pu trouver un groupe

d’utilisateurs de Linux à Delhi et j’ai eu l’opportunité de rencontrer de vrais hackers. J’avais l’impression d’être dans un autre monde quand ils parlaient de Perl, RMS, du noyau, des pilotes de périphériques, de compilation et d’autres choses qui me passaient bien au-dessus de la tête.

J’étais dans un autre monde. J’ai préféré rentrer à la maison et trouver une distribution Linux quelque part. J’avais bien trop peur pour leur en demander une. J’étais loin de leur niveau, un débutant totalement idiot. J’ai réussi à en obtenir une en payant 1 000 Roupies indiennes [NdT : environ 13,92 €] à un gars qui en faisait le commerce. J’en ai essayé beaucoup mais je n’arrivais pas à faire fonctionner le son. Cette fois-là, je décidai de consulter un canal IRC depuis le cybercafé. Je trouvai #linux-india et lançai : « g ok1 son ». Des injonctions fusèrent alors : « pas de langage SMS » et « RTFM ». Ça m’a fait encore plus peur et j’ai mis du temps à faire la relation entre « RTFM » et « read the fucking manual » [NdT : « lis le putain de manuel » dans la langue de Molière].

J’étais terrifié et j’ai préféré rester à l’écart de l’IRC pendant quelques semaines.

Un beau jour, j’ai reçu un courriel qui annonçait une réunion mensuelle de groupes d’utilisateurs Linux. J’avais besoin de réponses à mes nombreuses questions. C’est là que j’ai rencontré Karunakar. Il m’a demandé d’apporter mon ordinateur à son bureau, où il avait l’intégralité du dépôt de Debian. C’était nouveau pour moi, mais j’étais satisfait à l’idée de pouvoir enfin écouter de la musique sur Linux. Le lendemain, j’étais dans son bureau après avoir fait le trajet avec mon ordinateur dans un bus bondé, c’était génial. En quelques heures, Debian était opérationnel sur mon système. Il m’a aussi donné quelques livres pour débutants et une liste des commandes de base.

Le lendemain, j’étais à nouveau au cybercafé, et je lisais un

autre essai d'Eric S. Raymond, intitulé *Comment poser les questions de manière intelligente*. Je continuais de fréquenter le canal #hackers sur Yahoo chat où je m'étais fait un très bon ami, Krish, qui m'a suggéré d'acheter le livre intitulé *Commandes de références sous Linux*. Après avoir passé quelque temps avec ces livres et en utilisant tldp.org (The Linux Documentation Project) comme support, j'étais devenu un utilisateur débutant sous Linux. Je n'ai jamais regardé en arrière. J'ai aussi assisté à une conférence sur Linux où j'ai rencontré quelques hackers de Yahoo ; écouter leurs conférences m'a beaucoup inspiré. Quelques années plus tard, j'ai eu la chance de rencontrer Richard Stallman qui est une sorte de dieu pour beaucoup de gens dans la communauté du logiciel libre.

Je dois reconnaître que la documentation d'Eric S. Raymond a changé ma vie et sûrement celle de beaucoup d'autres. Après toutes ces années passées dans la communauté du logiciel libre, je me suis rendu compte que la documentation est la clé pour amener des débutants à participer à cette extraordinaire communauté *open source*. Mon conseil à deux balles à tous les développeurs serait : s'il vous plaît, documentez votre travail, même le plus petit, car le monde est plein de débutants qui aimeraient le comprendre. Mon blog propose un large éventail de publications, qui vont des plus simples comme l'activation de la vérification orthographique dans OpenOffice à celles, plus complexes, traitant de l'installation de Django dans un environnement virtuel.

[1] NdT : Un *hacker* sait où et comment bidouiller un programme pour effectuer des tâches autres que celles prévues par ses concepteurs alors qu'un *cracker* est un pirate informatique spécialisé dans le cassage des protections dites de sécurité.