Le protocole Gemini, revenir à du simple et sûr pour distribuer l'information en ligne ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 novembre 2020. Dernière mise à jour le 2 décembre 2020

`https://www.bortzmeyer.org/gemini.html`

Comme beaucoup de gens, je trouve que le Web, tout en étant un immense succès, souffre de plusieurs défauts. Tous ne proviennent pas de la technique, mais certains peuvent quand même être reliés à des choix lors de la conception des protocoles et formats. C'est par exemple le cas des problèmes de vie privée, car HTTP et HTML fournissent trop de moyens de pister les utilisateurs, et des problèmes stratégiques autour des navigateurs : aujourd'hui, écrire un navigateur Web complet, disons l'équivalent de Firefox, est une tâche colossale, réservée à de très grosses équipes très pointues. Cela limite la concurrence : seuls trois ou quatre navigateurs vraiment différents existent et encore moins de moteurs de rendu. (Et je n'ai pas encore mentionné le désir de davantage de sobriété numérique et d'empreinte environnementale réduite.) Le projet Gemini <https://gemini.circumlunar.space/> s'attaque à ce problème en définissant un nouveau protocole et un nouveau format, délibérement très simples et non extensibles, afin de rendre plus difficile l'apparition des mêmes problèmes que dans le Web.

Gemini est donc très simple. Il utilise :

— Un protocole à lui, qui ressemble un peu à la version 0.9 de HTTP <https://www.w3.org/Protocols/HTTP/AsImplemented.html> (en mieux, quand même),
— Un format à lui, une sorte de version simplifiée et uniformisée de Markdown,
— Les URL que nous connaissons,
— Le TLS que nous connaissons.

Un point important et délibéré de Gemini est son absence d'extensibilité. Le but est d'éviter ce qui est arrivé au Web, où un protocole simple et sûr du début a évolué en un gros machin compliqué et dangereux. Ainsi, Gemini n'a pas d'en-têtes de requêtes ou de réponses. Y ajouter un outil de flicage comme les "*cookies*" sera donc difficile et c'est exprès.

Si vous êtes intéressé par ce projet, consultez le site Web <https://gemini.circumlunar.space/> pour avoir une première idée. Si vous avez des questions (notamment si vous en êtes au stade « ah, les cons, pourquoi ils n'ont pas tout simplement utilisé X et/ou fait comme Y ? »), il est probable que la FAQ très détaillée <https://gemini.circumlunar.space/docs/faq.html> y répondra. Si vous êtes d'humeur à lire l'actuelle spécification, elle est aussi sur le Web <https://gemini.circumlunar.

space/docs/specification.html>. Mais beaucoup d'autres ressources à propos de Gemini ne sont évidemment pas sur le Web, mais dans le « "*geminispace*" ».

Comme ce « "*geminispace*" » ne peut pas se visiter avec un navigateur Web classique, il va falloir un client Gemini. Il en existe beaucoup. La plupart sont assez sommaires mais les lectrices et lecteurs de mon blog sont des gens avertis et qui savent qu'il ne faut pas juger un système de publication à ses interfaces actuelles. Déjà, je vais commencer par supposer que vous ne voulez pas installer un Nième logiciel sur votre machine mais que vous voudriez quand même jeter un coup d'œil. Ça tombe bien, plusieurs clients Gemini sont disponible sur un serveur SSH public. Connectez-vous en SSH à kiosk@gemini.circumlunar.space et essayez le client de votre choix. Personnellement, j'utilise Amfora. Vous pouvez visiter les liens de la page d'accueil en tapant espace puis leur numéro. À la place d'un numéro, vous pouvez aussi indiquer un URL Gemini comme gemini://gemini.bortzmeyer.org/. Si vous voulez tester plus loin, vous pouvez installer Amfora <https://github.com/makeworld-the-better-c amfora> ou bien un des autres clients <https://gemini.circumlunar.space/clients.html>. Sur Arch Linux, c'est aussi simple que de taper pacman -S amfora. Comme vous vous en doutez, il n'y a pas encore beaucoup de contenu disponible mais ça grossit tous les jours.

Comme dit plus haut, il y a de nombreux autres clients Gemini <https://gemini.circumlunar.space/clients.html>. Par exemple, les fans d'Emacs vont apprécier Elpher <https://thelambdalab.xyz/elpher/>, un client Gopher et Gemini pour Emacs. Voici son apparence :

Et, ici, avec le client graphique Lagrange <https://github.com/skyjake/lagrange> :

En plus sommaire, et si on aime la ligne de commande, récupérer un fichier avec Gemini est aussi simple que :

```
% echo -n  "gemini://purexo.mom/\r\n" | gnutls-cli -p 1965  purexo.mom
```

(gnutls-cli fait partie de GnuTLS.)

Et si vous voulez un serveur, pour distribuer vos idées géniales au monde entier ? Il existe plusieurs serveurs en logiciel libre, mais pas de page Web pour les présenter, il faut aller dans le "*geminispace*" en gemini://gemini.circumlunar.space/software/. J'ai choisi de commencer avec Agate <https://github.com/mbrubeck/agate>. Agate est écrit en Rust donc on l'installe en général par la méthode Rust habituelle, cargo install agate. Ensuite, il nous faut un certificat car Gemini impose TLS (notez qu'il semble que pas mal de serveurs Gemini n'aient qu'un certificat auto-signé). Demandons à Let's Encrypt :

```
# certbot certonly -n --standalone --domain gemini.bortzmeyer.org
```

Et je copie certificat et clé privée dans un répertoire à moi, pour ne pas exécuter Agate en étant root. (Pour le renouvellement du certificat, dans trois mois, il faudra que je trouve une solution plus pérenne.) Ensuite, je lance le serveur :

```
% ~/.cargo/bin/agate gemini.bortzmeyer.org:1965 /var/gemini fullchain.pem privkey.pem gemini.bortzmeyer.org
```

(Pourquoi le port 1965 est-il le port par défaut ? Je vous laisse chercher un peu.)

Mais il reste à ajouter du contenu, dans le répertoire `/var/gemini` qu'on a indiqué au serveur. Ce contenu est dans un format spécifique à Gemini, qui ressemble à Markdown, en plus limité. Je vous le dis tout de suite : il n'y a pas d'images ! Finies, les photos de chats mignons ! Voici le source de la page d'accueil de mon serveur :

```
% cat index.gmi
# First Gemini test

No actual content yet

=> https://www.afnic.fr/ AFNIC Web site
```

Vous noterez :

— Le titre (équivalent du `<h1>` de HTML) s'écrit comme en Markdown.
— Le texte s'écrit sans marques particulières.
— Les liens sont forcément sur une ligne. On ne peut pas faire d'hypertexte. (En prime, j'ai mis un lien Web, ce qui n'est pas très géministe et ne marchera pas avec les clients Gemini non-Web.)

Voilà, je vais essayer de mettre un peu plus de contenu que ce premier exemple mais ne comptez pas sur une publication de mon blog en Gemini : l'absence d'hypertexte nécessiterait de réécrire sérieusement les articles.

Je reviens un peu au choix des serveurs. On a vu qu'il y avait une liste en `gemini://gemini.circumlunar.space/s` mais elle n'est pas éditée, c'est juste un vrac tous les serveurs possibles, alors que certains sont très sommaires, voire déjà abandonnés. Voici donc une liste personnelle de ceux que j'ai testés, en commençant (oui, c'est subjectif), par ceux que je trouve les plus « prêts pour la prod' » (avec des liens Web à chaque fois, pour faciliter la vie de celles et ceux qui n'ont pas encore de client Gemini) :

— Molly-brown `<https://tildegit.org/solderpunk/molly-brown>` : le serveur de référence. Parfait en tout, mais n'a pas de "*virtual hosts*".
— Gemserv `<https://git.sr.ht/~int80h/gemserv>` : très bon serveur également, et il dispose de "*virtual hosts*".
— Agate `<https://github.com/mbrubeck/agate/>` : simple mais marche bien.
— Twins `<https://gitlab.com/tslocum/twins/>` : sans doute le plus riche en fonctions.
— geminid `<https://github.com/jovoro/geminid/>` : écrit en C, donc seulement si vous n'avez pas peur des débordements. Mais il marche bien.
— Net-gemini `<https://github.com/jackdoe/net-gemini>` : pas très documenté, mais j'ai l'impression que c'est plutôt une bibliothèque pour développer des serveurs adaptés à un usage particulier, par exemple la génération de pages dynamiques.
— Northstar `<https://github.com/panicbit/northstar>` : même remarque.
— GeGoBi `<https://tildegit.org/solderpunk/gegobi>` : plutôt pour les gens qui ont déjà du contenu Gopher et veulent le servir rapidement sur Gemini.
— Pollux `<https://git.sr.ht/~julienxx/pollux>` : très limité, il sert un seul fichier, et ne semble plus développé.
— Shavit `<https://sr.ht/~yotam/Shavit/>` : pour l'instant, il ne peut communiquer qu'avec des clients locaux à la machine.
— Blizanci `<https://github.com/mk270/blizanci>` : aucun développement depuis six mois.

Conclusion ? Ne me demandez pas si Gemini sera un succès ou pas, je suis mauvais pour les pronostics. Je dis juste que je suis content de voir que des gens ne se résignent pas à déplorer les problèmes du Web mais qu'ils essaient de les résoudre.

Autres articles en français sur Gemini :

— Gemini, le protocole du slow web `<https://ploum.net/gemini-le-protocole-du-slow-web/>`
— Le protocole Gemini `<https://toutetrien.lithio.fr/article/le-protocole-gemini>`
— Gemini et Solid, deux alternatives au Web (qu'il faut qu'on m'explique) `<https://linuxfr.org/users/hellpe/journaux/gemini-et-solid-deux-alternatives-au-web-qu-il-faut-qu-on-`