RFC 1288 : The Finger User Information Protocol

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 24 septembre 2013

Date de publication du RFC : Décembre 1991

https://www.bortzmeyer.org/1288.html

Il y a bien longtemps, dans l'Internet, toutes les machines Unix connectées au réseau avaient un serveur finger. Ce protocole permettait d'obtenir des informations sur les utilisateurs de cette machine, à la fois de l'information statique (leur numéro de téléphone, le numéro de leur bureau à l'université...) mais aussi de l'information dynamique (est-ce qu'ils sont connectés ou pas, actifs ou pas, quand ont-il lu leur courrier pour la dernière fois, ...) Aujourd'hui, on utiliserait Facebook pour cela mais, dans les années 80 et 90, pour se renseigner sur un collègue ou un confrère, on se servait de finger. L'importance croissante donnée à la vie privée a petit à petit conduit au démantelement de cet utile service (remplacé, on l'a vu, par de gros silos de données qui sont bien pires, notamment parce que l'information n'est pas sous le contrôle de l'utilisateur). Finger, normalisé dans ce RFC, n'a plus aujourd'hui qu'un intérêt historique.

Je ne connais pas aujourd'hui beaucoup de serveurs finger encore en activité (mais je suis sûr que les lecteurs de mon blog vont m'en trouver plusieurs). Alors, j'en ai créé un, par nostalgie, `blog@www.bortzmeyer.org`. Vous pouvez utiliser un client finger comme l'outil Unix du même nom pour l'interroger. Il ne donne pas de renseignements sur une personne mais sur un service, comme le faisaient un certain nombre de serveurs finger. Ainsi, il existait un `quake@geophys.washington.edu` qui donnait de l'information sur les derniers tremblements de terre détectés, et un `nasanews@space.mit.edu` qui servait les informations de la NASA sur ses vols spatiaux. Ces deux-là ont disparu, mais les toilettes du MIT diffusent toujours des informations :

```
% finger @bathroom.mit.edu
Random Hall Bathroom Server v2.1

                        Bonfire Kitchen: vacant for 28 min
                        Bonfire  Lounge: vacant for 3 min
                         Pecker  Lounge: vacant for 2 hr
                         Pecker Kitchen: *IN*USE* for 16 min
   K 282  L  290 K         Clam Kitchen: vacant for 43 min
   ... ... ... ...         Clam  Lounge: *IN*USE* for 33 min
  | o : o | o : x |          BMF  Lounge: vacant for 3 min
  | o : x | o : o |          BMF Kitchen: vacant for 52 min
```

```
  | o : o | o : o |         Loop Kitchen: vacant for 75 min
  | o : x | - : o |         Loop  Lounge: vacant for 3 min
 ~~~~~~~~~~~~~~~~~~  Black Hole  Lounge: vacant for 42 min
                     Black Hole Kitchen: vacant for 33 min
      o = vacant!       Destiny Kitchen: vacant for 2 min
      x = in use         Destiny Lounge: *IN*USE* for 33 min
                                   Foo: vacant for 17 min

For more information finger help@bathroom.mit.edu

(2146055)
```

Plus sérieux, le projet FreeBSD a toujours un serveur finger actif `@freebsd.org` donc vous pouvez vous informer sur n'importe quel *"commiter"* :

```
%  finger roberto@freebsd.org
Login: roberto         Name: Ollivier Robert
Directory: /home/roberto             Shell: /usr/local/bin/zsh
Office: Bretigny s/Orge FRANCE Office Phone: +33-1-69887290
No Mail.
Plan:
Home address: <roberto@keltia.net>
Work address: <ollivier.robert@eurocontrol.int>

Twitter: @Keltounet

pub  1024D/7DCAE9D3 1997-08-21 Ollivier Robert <roberto@keltia.net>
uid                            Ollivier Robert <roberto@keltia.freenix.fr>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.1 (FreeBSD)

mQGiBDP8ElkRBADQrtDGMaOawsdVCxQTrtCoa+VFeSebgBgIdfrgduTTOteV+bqz
RYa94GBx3Df/LCxEtb8bgcL6DlD/B5nCjzMfwzW+JkqXDz7g96oVWW48jUIeNrYL
qBZruvcopUEmH3iBZU8ig3lpJ5/XbN+IYGP474E2rnTNfaY26E0iWkWqtQCg/wIY
...
```

Autre serveur finger utile, celui de météorologie <`https://0p.no/2014/12/13/graph_no___weather_forecast_via_finger.html`> qui permet d'obtenir un rapport sur n'importe quelle ville du monde (ici, Budva) :

```
% date
Tue Feb 16 08:31:00 CET 2016

% finger budva@graph.no
              -= Meteogram for montenegro/budva/budva~3203106 =-
 'C                                                                  Rain (mm)
 23
 21   ---                                                 |          9 mm
 19                                                       |          8 mm
 17---   ^^^      ^^^^^^                                  |          7 mm
 15         ======         =====|                         |          6 mm
 13                       ^^^    |^^^===   ^^^         ==|          5 mm
 11                              |      =--   ---=-----  |         ^^^ 4 mm
  9                              |     |                 |=-----      3 mm
  7              |           |  |     |                 |      ^^^   2 mm
  5           |  |           |  |     |              |  |            1 mm
   _10_13 19 01_07_13 19 01_07_13 19 01_07_13 19 01_07_13 19 01_07_13 Hour

    NE SE NE  S  S SW NW  N NE  E NE NW NW  W NW NE NE SW NW NW NW SW Wind dir.
```

https://www.bortzmeyer.org/1288.html

```
     2  2  2  4  2  0  1  2  2  1  2  1  2  3  2  1  1  2  2  2  1  2 Wind(mps)

Legend left axis:   - Sunny   ^ Scattered   = Clouded   =V= Thunder   # Fog
Legend right axis:  | Rain    ! Sleet       * Snow
Mail a "thank you" to finger@falkp.no if you like the service.
```

Si vous voulez installer, vous aussi, un serveur finger, je vous mets en garde : c'est un service sensible et qui peut ouvrir des failles de sécurité. Après tout, son rôle est de distribuer de l'information. Et l'information peut être utile à un méchant.

Comment fonctionne ce protocole ? Difficile de faire plus simple (section 2.1). À l'époque, le moyen le plus courant de faire un protocole d'accès à l'information était de réserver un port TCP (ici, 79), de dire au client de se connecter à ce port et d'envoyer sa question sous forme d'une simple chaîne de caractères, et de dire au serveur d'écouter sur ce port, et de répondre à question par... ce qu'il veut. De nombreux protocoles avaient été conçus ainsi, comme whois (port 43, RFC 3912 [1]) ou comme daytime (port 13, RFC 867) qui était encore plus simple (même pas de question, de la part du client). Écrire un logiciel client pour ces protocoles est trivial (exercice de première heure de la première année en réseaux) et on peut même se contenter d'un simple telnet :

```
% telnet www.bortzmeyer.org finger
Trying 2605:4500:2:245b::42...
Connected to www.bortzmeyer.org.
Escape character is '^]'.
blog
Debian GNU/Linux      Copyright (C) 1993-1999 Software in the Public Interest
...
Connection closed by foreign host.
```

Notez bien (section 2.2) que la fin de la ligne « question » doit être composée de deux caractères, CR et LF. Certains serveurs finger sont plus tolérants que d'autres si vous violez cette règle. Si on utilise netcat comme client, on va préciser ces deux caractères :

```
% echo -n -e "blog\r\n" | nc www.bortzmeyer.org finger
```

On peut aussi utiliser un client whois puisque les deux protocoles sont très proches :

```
% whois -h www.bortzmeyer.org -p finger blog
```

Et le format de la réponse ? Il n'est pas du tout défini par ce RFC (section 2.5) qui dit juste que c'est du **texte libre**, conçu pour être analysé par un humain et pas par un programme. (C'est exactement la même chose avec whois.)

Le RFC donne des indications sur les informations utiles à distribuer. Mais il précise aussi que la liste effective est entièrement sous le contrôle de l'administrateur système. Dans le reste de cet article, je donnerai des exemples de configuration empruntés à deux serveurs finger distribués en logiciel libre, efingerd <http://kassiopeia.juls.savba.sk/~garabik/software/efingerd.html> et cfingerd <http://www.infodrom.org/projects/cfingerd/>. Par exemple, le RFC cite le numéro de téléphone, l'adresse, le nombre de minutes depuis la dernière activité sur cet ordinateur. Avec cfingerd, on peut décider d'afficher (signe plus) ou au contraire de ne pas activer (signe moins) l'envoi de ces informations, et on peut le faire pour les utilisateurs locaux à la machine (le second booléen) ou distants (le premier booléen). Un exemple dans le `cfingerd.conf` est :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3912.txt`

```
CONFIG finger_display  = {
...
        +REAL_NAME              = [FALSE, TRUE],
        +DIRECTORY              = [FALSE, TRUE],
        +SHELL                  = [FALSE, TRUE],
        +ROOM_NUMBER            = [FALSE, TRUE],
        +WORK_NUMBER            = [FALSE, TRUE],
        +HOME_NUMBER            = [FALSE, TRUE],
        +LAST_TIME_ON           = [FALSE, TRUE],
        +IF_ONLINE              = [FALSE, TRUE],
        +TIME_MAIL_READ         = [FALSE, TRUE],
        +DAY_MAIL_READ          = [FALSE, TRUE],
        +ORIGINATION            = [FALSE, TRUE],
        +PLAN                   = [TRUE, TRUE],
        +PROJECT                = [TRUE, TRUE],
        +PGP                    = [TRUE, TRUE],
...
```

Avec efingerd, cela se configure en éditant le script shell `luser` qui, par défaut, utilise la commande locale finger qui peut être très bavarde (certains serveurs finger affichaient même l'**expéditeur** du dernier courrier reçu !). Voici un exemple :

```
Login: stephane         Name:
Directory: /home/stephane            Shell: /usr/bin/zsh
On since Tue Sep 24 06:39 (UTC) on pts/0 from central.example.net
   1 minute 54 seconds idle
Last login Tue Sep 24 06:39 (UTC) on pts/2 from central.example.net
Mail forwarded to "| procmail -d"
New mail received Tue Sep 24 05:41 2013 (UTC)
     Unread since Tue Sep 24 13:28 2013 (UTC)
No Plan.
```

Notez que ces deux logiciels serveurs permettent de ne **pas** envoyer d'information sur un utilisateur particulier (bien que le RFC se demande à quoi cela sert de faire tourner un serveur finger dans ces conditions). Avec efingerd, c'est en éditant le script `luser`.

Parfois, le nom passé au serveur finger n'est pas celui d'un utilisateur humain mais celui d'un service (comme `blog` plus haut). La section 2.5.5 discute du cas où le service est un distributeur automatique et demande que finger renvoie la liste des produits actuellement disponibles dans la machine...

Une technique courante était de permettre aux utilisateurs de mettre un fichier nommé `.plan` dans leur répertoire maison. Ce fichier était alors envoyé après la réponse finger et contenait des informations supplémentaires comme une clé PGP. Les serveurs modernes permettent également aux utilisateurs de mettre un programme (et pas un simple fichier texte) dont le résultat sera envoyé au client finger. Inutile de décrire les risques de sécurité que cela entraine (section 3.2.5)...

finger offrait une autre possibilité : avoir la liste de tous les utilisateurs actuellement connectés, en indiquant une question vide. Donc, la commande `finger @machine-distante` (sans nom devant le @) donnait la liste de qui était branché. Particulièrement indiscrète, cette possibilité (qui n'avait en outre de sens que pour les grosses machines multi-utilisateurs) a été la première à disparaître (éditer le script `nouser` pour efingerd et option `SYSTEM_LIST` pour cfingerd).

À noter qu'il existe d'autres serveurs finger que ceux cités, par exemple IcculusFinger `<http://icculus.org/IcculusFinger/>` qui en prime est disponible sur le site de l'auteur :

```
% finger ryan@icculus.org
IcculusFinger v2.1.24


                              /Nothing to report./

------------------------------------------------------------------------
.plan archives for this user: finger ryan?listarchives=1@icculus.org
Powered by IcculusFinger v2.1.24 (http://icculus.org/IcculusFinger/)
Stick it in the camel and go.
```

Et la sécurité ? Absente du premier RFC, le RFC 742, elle est au contraire très présente dans ce RFC 1288, notamment en section 3. Il y a plusieurs aspects. D'abord, la sécurité de la mise en œuvre du serveur finger, qui fut illustrée par le ver de Morris. Notons que ce n'était pas un problème spécifique au protocole finger : tout serveur réseau devrait être programmé de manière défensive, de façon à ne pas être vulnérable lorsque le client envoie des données inattendues (très longues, par exemple). Ne pas avoir de serveur finger aujourd'hui, à cause d'une faille de sécurité d'un logiciel qui n'est plus utilisé depuis longtemps, est donc idiot.

Mais finger a évidemment un autre problème, déjà cité, vis-à-vis de la vie privée. On n'a pas forcément envie <https://www.bortzmeyer.org/finger-vie-privee.html> que n'importe qui sur l'Internet soit au courant de ses heures de présence au bureau ou du rythme auquel on reçoit du courrier. C'est pourquoi le RFC exige qu'on puisse ajuster l'information présentée (cfingerd fournit un contrôle très fin dans son fichier de configuration, efingerd vous laisse programmer ce que vous voulez dans les scripts qu'il exécute).

Si vous avez un serveur finger et que vous voulez le superviser depuis Icinga <https://www.bortzmeyer.org/icinga.html>, cela peut se faire avec la configuration :

```
define service{
        use   generic-service
        host_name            MonServeur
        service_description       Finger
        check_command             check_finger!utilisateur!chaîne à chercher
}
```

Elle lèvera une alarme si le texte `chaîne à chercher` n'apparait pas dans la réponse du serveur. La commande `check_finger` est définie ainsi :

```
define command {
        command_name    check_finger
        command_line    /usr/local/share/nagios/libexec/check_finger $HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$
        }
```

Et le script `check_finger` (inspiré d'un équivalent pour whois <http://jon.netdork.net/2009/03/09/nagios-and-monitoring-whois/>) est disponible ici (en ligne sur `https://www.bortzmeyer.org/files/check_finger.sh`).

Notre RFC succède au RFC 1196 et le premier RFC sur finger était le RFC 742. Comme beaucoup de protocoles Internet, finger a d'abord été mis en œuvre (parfois sous le nom de « Name » et pas « Finger »), puis normalisé. La section 1.2 du RFC résume une longue et glorieuse histoire. Le RFC note que la norme actuelle est fondée avant tout sur le comportement du serveur finger d'Unix BSD. Par rapport à son prédécesseur, le RFC 1196, on note surtout une définition plus rigoureuse : le RFC 1196 était souvent très vague. Aujourd'hui, on l'a vu, finger est une survivance du passé, mais le standard IETF de récupération d'information sur une entité, WebFinger (RFC 7033), lui rend hommage par son nom. Autre idée inspirée de finger (et l'utilisant), le réseau social expérimental Thimbl <http://www.thimbl.net/>. Comme le dit un contributeur d'Une contre-histoire des Internets <http://lesinternets.arte.tv/contribution/433/>, « Je me souviens de finger, seul service interrogeable qui indiquait les services accessibles sur un host »...