

Les commandes de réseau sous UNIX et GNU/Linux

Principales commandes : `ifconfig`, `ping`, `arp`, `rarp`, `route`, `ssh` et `traceroute`

Les commandes principales de réseau sous le système GNU/Linux sont des outils qui doivent être maîtrisés par tous les administrateurs en informatique. Ils permettent de configurer l'infrastructure des réseaux locaux (lan) et du réseau internet (wan) fonctionnant avec TCP/IP.

`ifconfig`, *InterFace CONFIGuration*, permet de configurer les interfaces réseau de la machine. Il faut savoir qu'il existe plusieurs types d'interfaces réseau.

Les plus courants sont les trois types d'interfaces suivants :

- l'interface `loopback`, qui représente le réseau virtuel de la machine, et qui permet aux applications réseau d'une même machine de communiquer entre elles avec ou sans carte réseau ;
- les interfaces des cartes réseau (que ce soient des cartes Ethernet, TokenRing ou autres) ;
- les interfaces `ppp`, `plip` ou `slip`, qui sont des interfaces permettant d'utiliser les connexions sérieuses, parallèles ou téléphoniques comme des réseaux.

La configuration d'une interface comprend l'initialisation des pilotes nécessaires à son fonctionnement et l'affectation d'une adresse IP à cette interface. La syntaxe générale que vous devrez utiliser est la suivante :

ex : `ifconfig interface adresse netmask masque up` où `interface` est le nom de l'interface réseau que vous voulez configurer, `adresse` est l'adresse IP que cette interface gèrera, et `netmask` est le masque de sous-réseau que vous utilisez. La commande `ifconfig` désigne les interfaces Ethernet en utilisant les noms `eth0`, `eth1`, etc... . Si vous désirez configurer l'interface `loopback`, vous devrez utiliser le nom d'interface `lo`.

Le paramètre `up` donné à `ifconfig` lui indique que l'interface doit être activée. Cela signifie que dès que la commande `ifconfig` s'achèvera, votre interface réseau sera active et fonctionnelle. Il existe le paramètre inverse : `down`. Ce paramètre s'utilise tout simplement dans la commande `ifconfig` :

ex : `ifconfig interface down` où `interface` est toujours le nom de l'interface.

`ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up` la configuration classique

(*) Note : lorsque vous écrivez vos adresses IP, ne pas rajouter de 0 supplémentaire devant les nombres qui la constituent. Les nombres préfixés d'un 0 sont codés en octal, c'est-à-dire en base 8 et on n'utilise pas les mêmes nombres que lorsqu'elle est exprimée en décimal (non fonctionnement de votre réseau si vous faisiez cette erreur !).

Le noyau utilisera par défaut le nombre 255 pour les adresses de `broadcast` dans les composantes de l'adresse IP qui ne fait pas partie de l'adresse de sous-réseau. Si vous désirez utiliser une autre adresse (l'alternative est de prendre l'adresse du sous-réseau), vous devrez utiliser l'option `broadcast` adresse dans la commande `ifconfig`. Cependant, le comportement par défaut convient à la plupart des réseaux. La commande de configuration donnée en exemple ci-dessus sera alors :

ex : `ifconfig eth0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.0.0 up`

Enfin, il est possible d'affecter plusieurs adresses IP à certaines interfaces réseau. C'est en particulier le cas pour toutes les interfaces réseau classiques. Cela n'est pas réalisable avec les interfaces de type point à point comme les interfaces des connexions ppp. Lorsqu'une interface dispose de plusieurs adresses, la première est considérée comme l'adresse principale de l'interface, et les suivantes comme des `alias`. Ces `alias` utilisent comme nom le nom de l'interface réseau principale et le numéro de l'`alias`, séparés par deux points (`:`) :

ex : si l'interface `eth0` dispose d'un `alias`, celui-ci sera nommé `eth0:0`.

Pour fixer l'adresse d'un `alias` d'une interface réseau , on utilisera :

ex : `ifconfig interface:numéro add adresse netmask masque` où `interface` est toujours le nom de l'interface, `numéro` est le n° de l'`alias`, `adresse` est l'IP à attribuer à cet `alias`, et `netmask` est le masque de sous-réseau de cette adresse.

La commande `ifconfig` est en général appelée dans les scripts d'initialisation du système, qui ont été générés par l'utilitaire de configuration réseau de votre distribution. Si vous effectuez un `grep` sur «`ifconfig`» dans le répertoire `/etc/rc.d/`, vous trouverez la commande de démarrage du réseau. Il se peut qu'une variable d'environnement soit utilisée à la place de l'adresse IP que vous avez choisie. Les lignes suivantes, qui permettent l'initialisation de l'interface `loopback` :

ex : `ifconfig lo 127.0.0.1 netmask 255.0.0.0 up`

`ping` envoie des datagrammes ICMP à des hôtes sur un réseau. Il permet de détecter bon nombre de problèmes concernant la configuration IP de l'interface Ethernet : adressage de votre carte réseau (adresse IP, adresse de réseau, routage, configuration de la résolution de nom). `ping` est un outil de diagnostic qui permet de s'assurer de la connexion entre deux machines sur un réseau. Attention, beaucoup de pare-feu interdisent explicitement les paquets ICMP :

ex : `ping 127.0.0.1` teste la configuration de l'interface `loopback`

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
```

```
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.093 ms
```

On peut tester la liaison avec un hôte distant en limitant le nombre de paquets à transmettre :

```
ping -c2 192.168.1.1
```

 où l'argument `-c` désigne le nombre de datagrammes

```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
```

```
64 bytes from 192.168.1.1: icmp_seq=0 ttl=254 time=0.640 ms premier (c=1)
```

```
64 bytes from 192.168.1.1: icmp_seq=1 ttl=254 time=0.627 ms second (c=2)
```

```
--- 192.168.1.1 ping statistics --- résumé des paquets transmis de la réponse
```

```
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
```

```
rtt min/avg/max/mdev = 0.627/0.633/0.640/0.026 ms, pipe 2
```

`arp`, *Address Resolution Protocol* manipule la table ARP du système : permet de visualiser et modifier la table ARP stockée en cache. La table ARP est une table de correspondance entre adresses IP et *adresses matérielles* (MAC). Elle est générée au fur et à mesure grâce au protocole ARP et stockée en cache. La commande permet de détecter 2 types de problèmes : le fonctionnement au niveau matériel (hardware) de la carte et la bonne correspondance entre une adresse IP et une carte réseau au moyen de son adresse MAC. Permet de voir entre autres s'il y a une double attribution d'une adresse IP sur un réseau, une mystification (spoofing) IP, ... :

ex : `/sbin/arp -e` affiche (tous) les hôtes : par défaut en style Linux

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.1.1	ether	00:A0:C5:50:0C:85	C		eth0
192.168.1.3	ether	00:10:4B:45:9C:87	C		eth0
markab.1.168.192.in-add	ether	00:D0:09:42:72:7F	C		eth0

`rarp`, *Reverse Address Resolution Protocol* manipule la table RARP du noyau de différentes façons. Les options de base permettent d'effacer une entrée de correspondance d'adresses et d'en redéfinir une manuellement. Pour des besoins de débogage, le programme `rarp` permet aussi de vider complètement la table RARP. Pour configurer RARP, il vous faudra connaître l'adresse Ethernet du client (c'est-à-dire, l'« adresse MAC ») en utilisant la commande `/sbin/ifconfig eth0`.

ex : `/sbin/rarp -s client-hostname client-enet-addr`

`/usr/sbin/arp -s client-ip client-enet-addr`

Si en retour vous obtenez : `SIOCSRARP: Invalid argument` vous devrez probablement charger le module `rarp` du noyau ou bien recompiler le noyau pour accepter RARP.

`route`, *Route* manipule la table de routage IP du noyau. Il est possible de définir plusieurs règles de routage actives simultanément. L'ensemble de ces règles constitue ce qu'on appelle la table de routage. La règle utilisée est sélectionnée par le noyau en fonction de l'adresse destination du paquet à router. Chaque règle indique donc un critère de sélection sur les adresses, et l'interface vers laquelle doivent être transférés les paquets dont l'adresse destination vérifie cette règle. Pour manipuler la table de routage nous exécuterons la commande `route` suivant la règle suivante :

ex : `route operation [-net | -host] address netmask mask interface`

Où `operation` est l'opération à effectuer sur la table de routage. L'opération la plus courante est simplement l'ajout d'une règle de routage, auquel cas `add` doit être utilisé. L'option suivante permet d'indiquer si le critère de sélection des paquets se fait sur l'adresse du réseau destination ou plus restrictivement sur l'adresse de la machine destination. En général, il est courant d'utiliser la sélection de toutes les adresses d'un même réseau et de les router vers une même interface. Dans tous les cas, `address` est l'adresse IP de la destination, que celle-ci soit un réseau ou une machine. Si la destination est un réseau, il faut indiquer le masque de sous-réseau `mask` à l'aide de l'option `netmask`. Enfin, `interface` est l'interface réseau vers laquelle doivent être envoyés les paquets qui vérifient les critères de sélection de cette règle.

La règle de routage à utiliser pour l'interface loopback est la suivante :

```
ex: route add -net 127.0.0.0 netmask 255.0.0.0 lo
```

Cette règle signifie que tous les paquets dont l'adresse de destination appartient au sous-réseau de classe A 127.0.0.0 doivent être transférés vers l'interface loopback. Cela implique en particulier que les paquets à destination de la machine d'adresse IP 127.0.0.1 (c'est-à-dire la machine locale) seront envoyés vers l'interface loopback (ils reviendront donc sur la machine locale).

Une autre règle de routage classique est la suivante :

```
ex: route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Elle permet d'envoyer tous les paquets à destination du réseau de classe C 192.168.0.0 vers la première interface Ethernet. C'est typiquement ce genre de règle qu'il faut utiliser pour faire fonctionner un réseau local.

(*) Note : une commande équivalente à `route` est `netstat -r`. La commande affiche la table de routage et effectue la résolution de nom dès que possible. Les commandes `route -n` ou `netstat -rn` affichent la table de routage sans résolution de nom.

`ssh`, *Secure SHell* est un programme qui permet de se connecter sur une machine distante, ou d'exécuter des commandes sur une machine distante. Il est supposé remplacer `rlogin` et `rsh`, et fournit des transmissions sécurisées et cryptées entre deux machines qui ne sont pas sûres, et ce à travers un réseau non sécurisé. On peut transférer des connexions X11 et des ports TCP/IP arbitraires à travers le tunnel sécurisé :

```
ex: ssh 192.168.1.3
```

```
The authenticity of host '192.168.1.3 (192.168.1.3)' can't be established.  
RSA key fingerprint is d0:e8:5a:d4:af:00:82:63:d2:71:4a:09:e1:fa:73:07.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.1.3' (RSA) to the list of known hosts.  
root@192.168.1.3's password:  
Last login: Sat Jan 15 20:03:46 2005
```

L'utilisateur peut prouver son identité sur la machine distante à l'aide de plusieurs méthodes, qui dépendent de la version du protocole SSH utilisée.

(*) Note : on préférera `ssh` par rapport à la commande `telnet` qui n'offre aucun protocole crypté (les informations sont transmises en clair sur le réseau : pas de sécurité). `ssh` est recommandé lorsqu'on effectue des manipulations sur une machine distante et plus particulièrement en mode super utilisateur sur la machine hôte. L'emploi et l'installation du serveur OpenSSH (version open source de `ssh`) est requis sur la machine distante. En qualité d'administrateur réseau, vous devez impérativement utiliser ce type de commande pour exécuter des manipulations à distance en mode sécurisé.

tracroute, *Traceroute* affiche les paquets routés sur le réseau depuis un hôte : permet d'atteindre une URL ou un poste donné... La commande `tracroute`, établit la route suivie par les paquets de données vers la destination. La route est constituée de tous les routeurs traversés pour arriver à destination :

```
ex : tracroute 192.168.1.3
```

```
tracroute to 192.168.1.3 (192.168.1.3), 30 hops max, 38 byte packets
1 192.168.1.3 (192.168.1.3) 1.077 ms 0.186 ms 0.155 ms
```

Il s'agit d'une machine locale c'est à dire qu'elle est située sur le même réseau, aucun routeur n'est traversé, l'adresse de destination est donc atteinte directement.

```
tracroute www.linuxfranch-county.org
```

```
tracroute to markab.linuxfranch-county.org (192.168.1.3),30 hops max,38 byte packets
1 markab.0.168.192.in-addr.arpa (192.168.1.3) 0.700 ms 0.197 ms 0.155 ms
```

Ici le site est hébergé par une machine locale (située sur le même réseau) appelé `markab` configurée avec des serveurs HTTP, DNS. Ces serveurs sont exploitables depuis l'extérieur.

```
tracroute www.trimaille.net
```

```
tracroute to www.trimaille.net (207.234.154.195), 30 hops max, 38 byte packets
1 192.168.1.1 (192.168.1.1) 0.768 ms 0.663 ms 0.646 ms
2 193-207-118-80.kaptech.net (80.118.207.193) 51.513 ms 36.237 ms 36.335 ms
3 117-193-118-80.kaptech.net (80.118.193.117) 37.666 ms 37.155 ms *
4 V4089.abv1-co-1.gaoland.net (62.39.148.21) 42.899 ms 46.545 ms 47.975 ms
5 telefonica-data.sfinx.tm.fr (194.68.129.179) 39.752 ms 264.266 ms 434.694 ms
6 P14-0-grtwaseq1.red.telefonica-wholesale.net (213.140.37.190) 532.176 ms 121.998 ms 124.247 ms
7 So5-2-0-0-grtmiana2.red.telefonica-wholesale.net (213.140.36.5) 150.977 ms 148.781 ms 147.571 ms
8 teusa-7-3-0-0-grtmiana2.red.telefonica-wholesale.net (213.140.39.50) 150.360 ms
9 66.119.71.190 (66.119.71.190) 148.943 ms 149.569 ms 148.956 ms
10 207.234.128.53 (207.234.128.53) 150.320 ms 146.850 ms 147.585 ms
11 ra.infohell.net (207.234.154.195) 151.695 ms 151.560 ms 150.974 ms
```

Ici on trace la route d'une adresse située à l'extérieur : tous les routeurs traversés sont indiqués et numérotés.

La commande `tracroute` peut-être utilisée comme un outil de diagnostic :

```
ex : tracroute 192.168.1.9
```

```
tracroute to 192.168.1.9 (192.168.1.9), 30 hops max, 38 byte packets
1 192.168.1.7 (192.168.1.7) 3000.058 ms !H 2999.590 ms !H 2999.903 ms !H
```

Ici la commande nous donne des informations différentes : il lui est impossible de trouver la route et il affiche directement la destination avec un des codes d'erreur suivant :

!H hôte introuvable (host unreachable)

!N réseau introuvable (network unreachable)

!P protocole introuvable (protocol unreachable)

Termes employés relatifs aux réseaux :

Base 10 : le système bien connu des nombres décimaux, représentant n'importe quelle valeur avec les chiffres 0-9.

Base 16 : habituellement utilisée dans les langages de programmation de bas et haut niveaux, connue encore en tant que système numérique hexadécimal, représentant les valeurs avec les chiffres 0-9 et les caractères A-F (insensible à la casse).

Base 85 : représentation d'une valeur grâce à 85 différents chiffres/caractères, cela permet des chaînes de caractères plus courtes mais jamais vue dans la pratique.

Bit : unité minimale de stockage, allumée (on)/vraie (1) ou éteinte (off)/fausse (0).

Byte : le plus souvent, une collection de 8 bits (mais ce n'est pas réellement une nécessité - regardez les systèmes des anciens ordinateurs).

Périphérique : matériel de connexion réseau, voir aussi NIC.

Hôte à double résidence : un hôte à double résidence est un noeud ayant deux interfaces réseau (physique ou virtuelle) sur deux liens différents, mais qui ne réalise pas de renvoi de paquets entre les interfaces.

Hôte : généralement, un hôte simple résident, présent sur un lien. Normalement, il n'a seulement qu'une interface réseau active, par exemple Ethernet ou (non pas et) PPP.

Interface : quasi-synonyme de "périphérique", voir aussi NIC.

En-tête IP : en-tête d'un paquet IP (chaque paquet réseau a un en-tête, son type dépendant de la couche réseau).

Lien : un lien est un médium de transport de paquet réseau de la couche 2, des exemples en sont Ethernet, PPP, SLIP, ATM, RNIS, Frame Relay, etc...

Noeud : un noeud est soit un hôte, soit un routeur.

Octet : une collection véritable de 8 bits, aujourd'hui synonyme de "Byte".

Port : information destinée au distributeur TCP/UDP (couche 4) afin de transporter l'information à la couche supérieure.

Protocole : chaque couche réseau contient la plupart du temps un champ protocole facilitant la distribution de l'information transportée à la couche supérieure, comme cela peut se voir dans la couche 2 (MAC) et 3 (IP)

Routeur : un routeur est un noeud possédant une ou plusieurs interface(s) réseau, capable d'envoyer les paquets entre ses interfaces.

Socket : une socket IP est définie par ses adresses source et destination, ses ports et (association)

Pile : une collection de couches relative au réseau.

Masque de sous-réseau : les réseaux IP utilisent un masque de bits afin de distinguer le réseau local de ceux qui sont distants.

Tunnel : un tunnel est typiquement une connexion point-à-point sur laquelle les paquets échangés transportent les données d'un autre protocole, un tunnel IPv6-in-IPv4 en est un exemple.