Utilitaire Socat

Table of Contents

Description

Socat est un utilitaire basé sur la ligne de commande qui établit deux flux d'octets bidirectionnels et transfère les données entre eux. Parce que les flux peuvent être construits à partir d'un grand ensemble de différents types de puits et de sources de données (voir les types d'adresse) et que de nombreuses options d'adresse peuvent être appliquées aux flux, socat peut être utilisé à différentes fins.

Filan est un utilitaire qui imprime des informations sur ses descripteurs de fichiers actifs à la sortie standard. Il a été écrit pour le débogage de socat , mais pourrait également être utile à d'autres fins. Utiliser l'option -h pour trouver plus d'informations.

Procan est un utilitaire qui imprime des informations sur les paramètres du processus à la sortie standard. Il a été écrit pour mieux comprendre certaines propriétés de processus UNIX et pour déboguer socat , mais pourrait être utile à d'autres fins également.

Cycle de vie

Le cycle de vie d'une instance de socat se compose généralement de quatre phases.

Dans la phase d' initialisation

les options de ligne de commande sont analysées et la journalisation est initialisée.

Pendant la phase ouverte

socat ouvre la première adresse et ensuite la deuxième adresse. Ces étapes bloquent généralement; Ainsi, en particulier pour les types d'adresses complexes comme les chaussettes, les demandes de connexion ou les boîtes de dialogue d'authentification doivent être terminées avant le démarrage de l'étape suivante.

Lors de la phase de transfert

socat surveille les descripteurs de fichiers en lecture et en écriture des deux flux via select () et, lorsque les données sont disponibles d'un côté et peuvent être écrites de l'autre côté, socat le lit et effectue des conversions de caractères si nécessaire. écrit les données dans le descripteur de fichier d'écriture de l'autre flux, puis continue d'attendre d'autres données dans les deux directions.

Lors de la phase de fermeture

Lorsque l'un des flux atteint effectivement la fin de vie, la phase de fermeture commence. Socat transfère la condition EOF à l'autre flux, c.-à-d. Tente d'arrêter uniquement son flux d'écriture, ce qui lui donne une chance de se terminer normalement. Pendant un temps défini, socat continue à transférer des données dans l'autre sens, puis ferme tous les canaux restants et se termine.

Options

Socat fournit des options de ligne de commande qui modifient le comportement du programme. Ils n'ont rien à voir avec les options d'adresse utilisées dans les spécifications d'adresses.

Option	Description
-V	Retourne la version et les informations sur les fonctionnalités disponibles pour la sortie standard et la sortie.
-h (-?)	Retourne un texte d'aide pour stdout décrivant les options de ligne de commande et les types d'adresses disponibles, puis quitte.
-hh (-??)	Comme -h, plus une liste des noms abrégés de toutes les options d'adresse disponibles. Certaines options dépendent de la plate-forme, cette sortie est donc utile pour vérifier l'implémentation particulière.
-hhh (-???)	Comme -hh, plus une liste de tous les noms d'options d'adresse disponibles.
-ré	Sans cette option, seuls les messages fatals et d'erreur sont générés. L'application de cette option imprime également des messages d'avertissement. Voir DIAGNOSTICS pour plus d'informations.
-d -d	Imprime les messages fatals, d'erreur, d'avertissement
-d -d -d	Imprime les messages fatals, d'erreur, d'avertissement, de notification et d'information.
-d -d -d -d	Imprime les messages fatals, d'erreur, d'avertissement, de notification, d'information et de débogage.
-RÉ	Consigne les informations sur les descripteurs de fichiers avant de commencer la phase de transfert.
-ly [<facility>]	Écrit des messages dans syslog au lieu de stderr; gravité comme défini avec l'option -d. Avec facultatif <facility>, le type syslog peut être sélectionné, par défaut "daemon". Les bibliothèques tierces peuvent ne pas obéir à cette option.
-lf <fichier journal>	Écrit des messages dans <fichier_journal> [nom_fichier] au lieu de stderr. Certaines bibliothèques tierces, en particulier libwrap, peuvent ne pas obéir à cette option.
-ls	Écrit des messages sur stderr (c'est la valeur par défaut). Certaines bibliothèques tierces peuvent ne pas obéir à cette option, en particulier libwrap semble se connecter uniquement à syslog.
-lp <nomprog>	Remplace le nom du programme imprimé dans les messages d'erreur et utilisé pour construire des noms de variables d'environnement.

Option	Description
-lu	Étend l'horodatage des messages d'erreur à la résolution microseconde. Ne fonctionne pas lors de la connexion à syslog.
-lm [<facility>]	Mode journal mixte. Pendant le démarrage, les messages sont imprimés sur stderr; lorsque socat démarre la boucle de phase de transfert ou le mode démon (c.-à-d. après avoir ouvert tous les flux et avant de commencer le transfert de données, ou avec les sockets d'écoute avec l'option fork, avant le premier appel), la journalisation passe à syslog. Avec facultatif <facility>, le type syslog peut être sélectionné, par défaut “daemon”.
-lh	Ajoute le nom d'hôte pour consigner les messages. Utilise la valeur de la variable d'environnement HOSTNAME ou la valeur extraite avec uname () si HOSTNAME n'est pas défini.
-v	Écrit les données transférées non seulement dans leurs flux cibles, mais également dans stderr. Le format de sortie est du texte avec certaines conversions pour la lisibilité, et préfixé par “>” ou “<” pour indiquer le sens du flux.
-X	Écrit les données transférées non seulement dans leurs flux cibles, mais également dans stderr. Le format de sortie est hexadécimal, préfixé par “>” ou “<” indiquant les directions du flux. Peut être combiné avec -v.
-b <taille>	Définit le bloc de transfert de données <size> [size_t]. Au plus <taille> octets sont transférés par étape. La valeur par défaut est 8192 octets.
-s	Par défaut, socat se termine lorsqu'une erreur s'est produite pour empêcher le processus de s'exécuter lorsqu'une option n'a pas pu être appliquée. Avec cette option, socat est bâclé avec des erreurs et essaie de continuer. Même avec cette option, socat quittera les fatals et annulera les tentatives de connexion en cas d'échec des contrôles de sécurité.
-t <timeout>	Lorsqu'un canal a atteint EOF, la partie écriture de l'autre canal est arrêtée. Ensuite, socat attend <timeout> [timeval] secondes avant de se terminer. La valeur par défaut est 0,5 seconde. Ce délai d'attente s'applique uniquement aux adresses où la partie écriture et lecture peut être fermée indépendamment. Lorsque, pendant l'intervalle de temporisation, la partie lecture donne EOF, socat se termine sans attendre le délai d'attente.
-T <timeout>	Délai d'inactivité total: lorsque socat est déjà dans la boucle de transfert et qu'il ne s'est rien passé pendant <timeout> [timeval] secondes (aucune donnée n'est arrivée, aucune interruption n'est survenue ...), alors elle se termine. Utile avec des protocoles comme UDP qui ne peuvent pas transférer EOF.
-u	Utilise le mode unidirectionnel. La première adresse n'est utilisée que pour la lecture et la seconde adresse est utilisée uniquement pour l'écriture (exemple).
-U	
-g	Lors de l'analyse de l'option d'adresse, ne vérifie pas si l'option est considérée comme utile dans l'environnement d'adresse indiqué. Utilisé si on souhaite, par exemple, forcer une option de socket sur un périphérique série.
-L <fichier de verrouillage>	Si le fichier de verrouillage existe, quitte avec erreur. Si le fichier de verrouillage n'existe pas, le crée et continue, dissocie le fichier de verrouillage à la sortie.
-W <fichier de verrouillage>	Si le fichier de verrouillage existe, attend qu'il disparaisse. Lorsque le fichier de verrouillage n'existe pas, le crée et continue, supprime le fichier de verrouillage à la sortie.
-4	Utilise la version IP 4 au cas où les adresses ne spécifient pas implicitement ou explicitement une version; c'est la valeur par défaut.
-6	Utilise la version IP 6 au cas où les adresses ne spécifient pas implicitement ou explicitement une version.

Spécifications d'adresse

Avec les arguments de ligne de commande d'adresse, l'utilisateur donne des instructions socat et les informations nécessaires pour établir les flux d'octets. Une spécification d'adresse consiste généralement en un mot-clé de type adresse, zéro ou plusieurs paramètres d'adresse requis séparés par ':' du mot-clé et les uns des autres, et zéro ou plusieurs options d'adresse séparées par ','.

Le mot-clé spécifie le type d'adresse (par exemple, TCP4, OPEN, EXEC). Pour certains mots-clés, il existe des synonymes ('-' pour STDIO, TCP pour TCP4). Les mots clés sont insensibles à la casse. Pour quelques types d'adresse spéciaux, le mot-clé peut être omis: Les spécifications d'adresse commençant par un nombre sont supposées être des adresses FD (descripteur de fichier brut); si un '/' est trouvé avant le premier ':' ou ',', GOPEN (fichier générique ouvert) est supposé.

Le nombre et le type requis de paramètres d'adresse dépendent du type d'adresse. Par exemple, TCP4 requiert une spécification de serveur (nom ou adresse) et une spécification de port (numéro ou nom de service). Zéro ou plusieurs options d'adresse peuvent être données avec chaque adresse. Ils influencent l'adresse à certains égards. Les options consistent en un mot-clé d'option ou un mot-clé d'option et une valeur, séparés par '='. Les mots-clés d'options sont insensibles à la casse. Pour filtrer les options utiles avec un type d'adresse, chaque option est membre d'un groupe d'options. Pour chaque type d'adresse, un ensemble de groupes d'options est autorisé. Seules les options appartenant à l'un de ces groupes d'adresses peuvent être utilisées (sauf avec l'option -g).

Les spécifications d'adresse qui suivent le schéma ci-dessus sont également appelées spécifications d'adresse unique . Deux adresses simples peuvent être combinées avec “!!” pour former une adresse de type double pour un canal. Ici, la première adresse est utilisée par socat pour lire les données et la seconde adresse pour écrire les données. Il est impossible de spécifier une option une seule fois pour l'appliquer aux deux adresses uniques.

Habituellement, les adresses sont ouvertes en mode lecture / écriture. Lorsqu'une adresse fait partie d'une spécification à double adresse ou lorsque l'option -u ou -U est utilisée, une adresse peut être utilisée uniquement pour la lecture ou l'écriture. Considérant cela est important avec certains types d'adresses. Avec socat version 1.5.0 et supérieure, l'analyse lexicale tente de gérer les guillemets et les parenthèses de manière significative et permet d'échapper des caractères spéciaux. Si l'un des caractères ({['est trouvé, le caractère de fermeture correspondant -)}]' - est recherché; ils peuvent aussi être imbriqués. Dans ces constructions, les caractères et les chaînes de caractères spéciaux de socats:, !! ne sont pas traités spécialement. Tous ces caractères et chaînes peuvent être échappés avec \ ou dans “”

Les sections suivantes décrivent les types d'adresses disponibles avec leurs mots-clés, paramètres et sémantique.

Les adresses IP

IP-SENDTO: <hôte>: <protocole>

Ouvre un socket IP brut. Selon la spécification de l'hôte ou l'option pf, le protocole IP version 4 ou 6 est utilisé. Il utilise <protocol> pour envoyer des paquets à <host> [adresse IP] et reçoit des paquets de l'hôte, ignore les paquets d'autres hôtes. Le protocole 255 utilise le socket brut avec l'en-tête IP faisant partie des données.

Groupes d'options: FD, SOCKET, IP4, IP6
Options utiles: pf, ttl
Voir aussi: IP4-SENDTO, IP6-SENDTO, IP-RECVFROM, IP-RECV, UDP-SENDTO, UNIX-SENDTO

IP4-SENDTO: <hôte>: <protocole>	Comme IP-SENDTO, mais utilise toujours IPv4. Groupes d'options: FD, SOCKET, IP4
IP6-SENDTO: <hôte>: <protocole>	Comme IP-SENDTO, mais utilise toujours IPv6.

Groupes d'options: FD, SOCKET, IP6

IP-DATAGRAM: <adresse>: <protocole>

Envoie des données sortantes à l'adresse spécifiée, qui peut notamment être une adresse de diffusion ou de multidiffusion. Les paquets arrivant sur le socket local sont vérifiés si leurs adresses source correspondent aux options RANGE ou TCPWRAP. Ce type d'adresse peut par exemple être utilisé pour mettre en œuvre des communications de diffusion ou de multidiffusion symétriques ou asymétriques.

Groupes d'options: FD, SOCKET, IP4, IP6, RANGE
Options utiles: bind, range, tcpwrap, broadcast, boucle ip multicast, ip-multicast-ttl, ip-multicast-if, ip-add-membership, ttl, tos, pf
Voir aussi: DATAGRAM IP4, DATAGRAM IP6, IP-SENDTO, IP-RECVFROM, IP-RECV, UDP-DATAGRAM

IP4-DATAGRAM: <hôte>: <protocole>	Comme IP-DATAGRAM, mais utilise toujours IPv4. (Exemple) `Groupes d'options: FD, SOCKET, IP4, RANGE`
IP6-DATAGRAM: <hôte>: <protocole>	Comme IP-DATAGRAM, mais utilise toujours IPv6. IPv6 ne connaît pas les émissions. `Groupes d'options: FD, SOCKET, IP6, RANGE`

IP-RECVFROM: <protocole>

Ouvre un socket IP brut de <protocole>. Selon l'option pf, le protocole IP version 4 ou 6 est utilisé. Il reçoit un paquet d'un pair non spécifié et peut envoyer un ou plusieurs paquets de réponse à ce pair. Ce mode est particulièrement utile avec l'option fork où chaque paquet - provenant de pairs arbitraires - est géré par son propre sous-processus. Cela permet un comportement similaire aux serveurs basés sur UDP typiques tels que ntpd ou named.

les paquets de réponse peuvent être récupérés en tant que trafic entrant lorsque l'adresse IP de l'expéditeur et du destinataire est identique car il n'y a pas de numéro de port pour distinguer les sockets.

Cette adresse fonctionne bien avec les homologues d'adresses IP-SENDTO (voir ci-dessus). Le protocole 255 utilise le socket brut avec l'en-tête IP faisant partie des données.

Groupes d'options: FD, SOCKET, IP4, IP6, CHILD, RANGE
Options utiles: pf, fork, range, ttl, broadcast
Voir aussi: IP4-RECVFROM, IP6-RECVFROM, IP-SENDTO, IP-RECV, UDP-RECVFROM, UNIX-RECVFROM

IP4-RECVFROM: <protocole>	Comme IP-RECVFROM, mais utilise toujours IPv4. `Groupes d'option: FD, SOCKET, IP4, CHILD, RANGE`
IP6-RECVFROM: <protocole>	Comme IP-RECVFROM, mais utilise toujours IPv6. `Groupes d'options: FD, SOCKET, IP6, CHILD, RANGE`

IP-RECV: <protocole>

Ouvre un socket IP brut de <protocole>. Selon l'option pf, le protocole IP version 4 ou 6 est utilisé. Il reçoit des paquets de plusieurs pairs non spécifiés et fusionne les données. Aucune réponse n'est possible. Cela peut être, par exemple, adressé par des homologues d'adresses IP-SENDTO. Le protocole 255 utilise le socket brut avec l'en-tête IP faisant partie des données.

Groupes d'options: FD, SOCKET, IP4, IP6, RANGE
Options utiles: pf, range
Voir aussi: IP4-RECV, IP6-RECV, IP-SENDTO, IP-RECVFROM, UDP-RECV, UNIX-RECV

IP4-RECV: <protocole>	Comme IP-RECV, mais utilise toujours IPv4. `Groupes d'options: FD, SOCKET, IP4, RANGE`
IP6-RECV: <protocole>	Comme IP-RECV, mais utilise toujours IPv6. `Groupes d'options: FD, SOCKET, IP6, RANGE`

Les adresses SCTP

SCTP est un protocole de transport, équivalent dans un certain sens au TCP ou à l'UDP. En effet, il fournit des services similaires à TCP, assurant optionnellement la fiabilité, la remise en ordre des séquences, et le contrôle de congestion.

SCTP-CONNECT: <hôte>: <port>

Établit une connexion de flux SCTP avec les <hôte> [adresse IP] et <port> [service TCP] spécifiés à l'aide de TCP / IP version 4 ou 6, selon la spécification d'adresse, la résolution de nom ou l'option pf.

Groupes d'options: FD, SOCKET, IP4, IP6, SCTP, CHILD, RETRY
Options utiles: lier, pf, timeout, tos, mtudiscover, sctp-maxseg, sctp-nodelay, nonblock, sourceport, retry, readbytes
Voir aussi: SCTP4-CONNECT, SCTP6-CONNECT, SCTP-LISTEN, TCP-CONNECT

SCTP4-CONNECT: <hôte>: <port>	Comme SCTP-CONNECT, mais ne supporte que le protocole IPv4. `Groupes d'options: FD, SOCKET, IP4, SCTP, CHILD, RETRY`
SCTP6-CONNECT: <hôte>: <port>	Comme SCTP-CONNECT, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, IP6, SCTP, CHILD, RETRY`

SCTP-LISTEN: <port>

Écoute sur <port> [service TCP] et accepte une connexion TCP / IP. La version IP est 4 ou celle spécifiée avec l'option d'adresse pf, l'option socat (-4, -6) ou la variable d'environnement SOCAT_DEFAULT_LISTEN_IP. L'ouverture de cette adresse est généralement bloquée jusqu'à ce qu'un client se connecte.

Groupes d'option: FD, SOCKET, LISTEN, CHILD, RANGE, IP4, IP6, SCTP, RETRY
Options utiles: crnl, fork, bind, range, tcpwrap, pf, max-enfants, backlog, sctp-maxseg, sctp-nodelay, su, reuseaddr, retry, cool-write
Voir aussi: SCTP4-LISTEN, SCTP6-LISTEN, TCP-LISTEN, SCTP-CONNECT

SCTP4-LISTEN: <port>	Comme SCTP-LISTEN, mais ne supporte que le protocole IPv4. `Groupes d'option: FD, SOCKET, LISTEN, CHILD, RANGE, IP4, SCTP, RETRY`
SCTP6-LISTEN: <port>	Comme SCTP-LISTEN, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, LISTEN, CHILD, RANGE, IP6, SCTP, RETRY`

Les adresses SOCKET

SOCKET-CONNECT: <domaine>: <protocole>: <adresse distante>

Crée un socket de flux en utilisant les premier et second paramètres de socket donnés et SOCK_STREAM (voir man socket \ (2)) et se connecte à l'adresse distante. Les deux paramètres de socket doivent être spécifiés par int numbers. Consulter la documentation de votre système d'exploitation et inclure les fichiers pour trouver les valeurs appropriées. L'adresse distante doit être la représentation de données d'une structure sockaddr sans composants sa_family et (BSD) sa_len.

On peut - au-delà des options des groupes spécifiés - utiliser également les options des protocoles de niveau supérieur lorsqu'on applique l'option socat -g.

Groupes d'options: FD, SOCKET, CHILD, RETRY
Options utiles: bind, setsockopt-int, setsockopt-bin, setsockopt-string
Voir aussi: TCP, UDP-CONNECT, UNIX-CONNECT, SOCKET-LISTEN, SOCKET-SENDTO

SOCKET-DATAGRAM: <domaine>: <type>: <protocole>: <adresse distante>

Crée un socket datagramme en utilisant les trois premiers paramètres de socket donnés (voir man socket \ (2)) et envoie les données sortantes à l'adresse distante. Les trois paramètres de socket doivent être spécifiés par int numbers. Inclure les fichiers pour trouver les valeurs appropriées. L'adresse distante doit être la représentation de données d'une structure sockaddr sans composants sa_family et (BSD) sa_len.

On peut - au-delà des options des groupes spécifiés - utiliser également les options des protocoles de niveau supérieur lorsqu'on applique l'option socat -g.

Groupes d'options: FD, SOCKET, RANGE
Options utiles: bind, range, setsockopt-int, setsockopt-bin, setsockopt-string
Voir aussi: DATAGRAM UDP, DATAGRAM IP, SOCKET-SENDTO, SOCKET-RECV, SOCKET-RECVFROM

SOCKET-LISTEN: <domaine>: <protocole>: <adresse-locale>

Crée un socket de flux en utilisant les premier et second paramètres de socket donnés et SOCK_STREAM (voir man socket \ (2)) et attend les connexions entrantes sur les adresses locales. Les deux paramètres de socket doivent être spécifiés par int numbers. Inclure les fichiers pour trouver les valeurs appropriées. L'adresse locale doit être la représentation de données d'une structure sockaddr sans composants sa_family et (BSD) sa_len.

On peut - au-delà des options des groupes spécifiés - utiliser également les options des protocoles de niveau supérieur lorsqu'on applique l'option socat -g.

Groupes d'options: FD, SOCKET, LISTEN, RANGE, CHILD, RETRY
Options utiles: setsockopt-int, setsockopt-bin, setsockopt-string
Voir aussi: TCP, UDP-CONNECT, UNIX-CONNECT, SOCKET-LISTEN, SOCKET-SENDTO, SOCKET-SENDTO

SOCKET-RECV: <domaine>: <type>: <protocole>: <adresse-locale>

Crée un socket en utilisant les trois paramètres de socket donnés (voir man socket \ (2)) et le lie à <local-address>. Reçoit les données à l'arrivée. Les trois paramètres doivent être spécifiés par int numbers. Inclure les fichiers pour trouver les valeurs appropriées. L'adresse locale doit être la représentation de données d'une structure sockaddr sans composants sa_family et (BSD) sa_len.

Groupes d'options: FD, SOCKET, RANGE
Options utiles: range, setsockopt-int, setsockopt-bin, setsockopt-string
Voir aussi: UDP-RECV, IP-RECV, UNIX-RECV, SOCKET-DATAGRAM, SOCKET-SENDTO, SOCKET-RECVFROM

SOCKET-RECVFROM: <domaine>: <type>: <protocole>: <adresse-locale>

Crée un socket en utilisant les trois paramètres de socket donnés (voir man socket \ (2)) et le lie à <local-address>. Reçoit les données à l'arrivée et renvoie les réponses à l'expéditeur. Les trois premiers paramètres doivent être spécifiés en tant que nombres int. Inclure les fichiers pour trouver les valeurs appropriées. L'adresse locale doit être la représentation de données d'une structure sockaddr sans composants sa_family et (BSD) sa_len.

Groupes d'options: FD, SOCKET, CHILD, RANGE
Options utiles: fork, range, setsockopt-int, setsockopt-bin, setsockopt-string
Voir aussi: UDP-RECVFROM, IP-RECVFROM, UNIX-RECVFROM, SOCKET-DATAGRAM, SOCKET-SENDTO, SOCKET-RECV

SOCKET-SENDTO: <domaine>: <type>: <protocole>: <adresse distante>

Crée un socket en utilisant les trois paramètres de socket donnés (voir man socket \ (2)). Envoie les données sortantes à l'adresse indiquée et reçoit les réponses. Les trois paramètres doivent être spécifiés en tant que nombres int. Inclure les fichiers pour trouver les valeurs appropriées. L'adresse distante doit être la représentation de données d'une structure sockaddr sans composants sa_family et (BSD) sa_len.

Groupes d'options: FD, SOCKET
Options utiles: bind, setsockopt-int, setsockopt-bin, setsockopt-string
Voir aussi: UDP-SENDTO, IP-SENDTO, UNIX-SENDTO, SOCKET-DATAGRAM, SOCKET-RECV SOCKET-RECVFROM

Les adresses TCP

TCP: <hôte>: <port>

Se connecte à <port> [service TCP] sur <hôte> [adresse IP] à l'aide de TCP / IP version 4 ou 6, selon la spécification d'adresse, la résolution de nom ou l'option pf.

Groupes d'options: FD, SOCKET, IP4, IP6, TCP, RETRY
Options utiles: crnl, bind, pf, connect-timeout, tos, mtudiscover, mss, nodelay, nonblock, sourceport, réessayer, readbytes
Voir aussi: TCP4, TCP6, TCP-LISTEN, UDP, SCTP-CONNECT, UNIX-CONNECT

TCP4: <hôte>: <port>	Comme TCP, mais ne supporte que le protocole IPv4 (exemple). `Groupes d'options: FD, SOCKET, IP4, TCP, RETRY`
TCP6: <hôte>: <port>	Comme TCP, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, IP6, TCP, RETRY`

TCP-LISTEN: <port>

Écoute sur <port> [service TCP] et accepte une connexion TCP / IP. La version IP est 4 ou celle spécifiée avec l'option d'adresse pf, l'option socat (-4, -6) ou la variable d'environnement SOCAT_DEFAULT_LISTEN_IP. L'ouverture de cette adresse est généralement bloquée jusqu'à ce qu'un client se connecte.

Groupes d'options: FD, SOCKET, LISTEN, CHILD, RANGE, IP4, IP6, TCP, RETRY
Options utiles: crnl, fork, bind, range, tcpwrap, pf, max-children, backlog, mss, su, reuseaddr, réessayer, cool-write
Voir aussi: TCP4-LISTEN, TCP6-LISTEN, UDP-LISTEN, SCTP-LISTEN, UNIX-LISTEN, OPENSSL-

LISTEN, TCP-CONNECT

TCP4-LISTEN: <port>	Comme TCP-LISTEN, mais ne supporte que le protocole IPv4 (exemple). `Groupes d'options: FD, SOCKET, LISTEN, CHILD, RANGE, IP4, TCP, RETRY`

TCP6-LISTEN: <port>	Comme TCP-LISTEN, mais ne supporte que le protocole IPv6.
	
	`Option utile supplémentaire: ipv6only`
	`Groupes d'options: FD, SOCKET, LISTEN, CHILD, RANGE, IP6, TCP, RETRY`

Les adresses UDP

UDP: <hôte>: <port>

Se connecte à <port> [service UDP] sur <hôte> [adresse IP] en utilisant UDP / IP version 4 ou 6 selon la spécification d'adresse, la résolution de nom ou l'option pf.

En raison des propriétés du protocole UDP, aucune connexion réelle n'est établie. les données doivent être envoyées pour la «connexion» au serveur, et aucune condition de fin de fichier ne peut être transportée.

Groupes d'options: FD, SOCKET, IP4, IP6
Options utiles: ttl, tos, bind, sourceport, pf
Voir aussi: UDP4, UDP6, UDP-LISTEN, TCP, IP

UDP4: <hôte>: <port>	Comme UDP, mais ne supporte que le protocole IPv4. `Groupes d'options: FD, SOCKET, IP4`
UDP6: <hôte>: <port>	Comme UDP, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, IP6`

UDP-DATAGRAM: <adresse>: <port>

Envoie des données sortantes à l'adresse spécifiée, qui peut notamment être une adresse de diffusion ou de multidiffusion. Les paquets arrivant sur le socket local sont vérifiés pour le port distant correct et si leurs adresses source correspondent aux options RANGE ou TCPWRAP. Ce type d'adresse peut par exemple être utilisé pour mettre en œuvre des communications de diffusion ou de multidiffusion symétriques ou asymétriques.

Groupes d'options: FD, SOCKET, IP4, IP6, RANGE
Options utiles: bind, range, tcpwrap, broadcast, boucle ip multicast, ip-multicast-ttl, ip-

multicast-if, ip-add-membership, ttl, tos, sourceport, pf
Voir aussi: UDP4-DATAGRAM, UDP6-DATAGRAM, UDP-SENDTO, UDP-RECVFROM, UDP-RECV, UDP-CONNECT, UDP-LISTEN, IP-DATAGRAM

UDP4-DATAGRAM: <adresse>: <port>	Comme UDP-DATAGRAM, mais ne supporte que le protocole IPv4 (exemple1, exemple2). `Groupes d'options: FD, SOCKET, IP4, RANGE`
UDP6-DATAGRAM: <adresse>: <port>	Comme UDP-DATAGRAM, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, IP6, RANGE`

UDP-LISTEN: <port>

Attend un paquet UDP / IP arrivant sur <port> [service UDP] et se connecte à l'expéditeur. La version IP acceptée est 4 ou celle spécifiée avec l'option pf. En raison des propriétés du protocole UDP, aucune connexion réelle n'est établie.les données doivent d'abord être fournies par l'homologue et aucune condition de fichier ne peuvent être transportées. Cette adresse est généralement bloquée jusqu'à ce qu'un client se connecte.

Groupes d'option: FD, SOCKET, LISTEN, CHILD, RANGE, IP4, IP6
Options utiles: fork, bind, range, pf
Voir aussi: UDP, UDP4-LISTEN, UDP6-LISTEN, TCP-LISTEN

UDP4-LISTEN: <port>	Comme UDP-LISTEN, mais ne supporte que le protocole IPv4. `Groupes d'option: FD, SOCKET, LISTEN, CHILD, RANGE, IP4`
UDP6-List: <port>	Comme UDP-LISTEN, mais ne supporte que le protocole IPv6. `Groupes d'option: FD, SOCKET, LISTEN, CHILD, RANGE, IP6`

UDP-SENDTO: <hôte>: <port>

Communique avec le socket homologue spécifié, défini par <port> [service UDP] sur <hôte> [adresse IP], en utilisant UDP / IP version 4 ou 6 selon la spécification d'adresse, la résolution de nom ou l'option pf. Il envoie des paquets à et reçoit des paquets de cette socket homologue uniquement. Cette adresse implémente efficacement un client de datagramme. Cela fonctionne bien avec les homologues d'adresse socat UDP-RECVFROM et UDP-RECV.

Groupes d'options: FD, SOCKET, IP4, IP6
Options utiles: ttl, tos, bind, sourceport, pf
Voir aussi: UDP4-SENDTO, UDP6-SENDTO, UDP-RECVFROM, UDP-RECV, UDP-CONNECT, UDP-LISTEN, IP-SENDTO

UDP4-SENDTO: <hôte>: <port>	Comme UDP-SENDTO, mais ne supporte que le protocole IPv4. `Groupes d'options: FD, SOCKET, IP4`
UDP6-SENDTO: <hôte>: <port>	Comme UDP-SENDTO, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, IP6`

UDP-RECVFROM: <port>

Crée un socket UDP sur <port> [service UDP] en utilisant UDP / IP version 4 ou 6 selon l'option pf. Il reçoit un paquet d'un pair non spécifié et peut envoyer un ou plusieurs paquets de réponse à ce pair. Ce mode est particulièrement utile avec l'option fork où chaque paquet - provenant de pairs arbitraires - est géré par son propre sous-processus. Cela permet un comportement similaire aux serveurs basés sur UDP typiques tels que ntpd ou named. Cette adresse fonctionne bien avec les homologues d'adresse UDP-SENDTO.

Groupes d'options: FD, SOCKET, IP4, IP6, CHILD, RANGE
Options utiles: fork, ttl, tos, bind, sourceport, pf
Voir aussi: UDP4-RECVFROM, UDP6-RECVFROM, UDP-SENDTO, UDP-RECV, UDP-CONNECT, UDP-LISTEN, IP-RECVFROM, UNIX-RECVFROM

UDP4-RECVFROM: <port>	Comme UDP-RECVFROM, mais ne supporte que le protocole IPv4. `Groupes d'option: FD, SOCKET, IP4, CHILD, RANGE`
UDP6-RECVFROM: <port>	Comme UDP-RECVFROM, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, IP6, CHILD, RANGE`

UDP-RECV: <port>

Crée un socket UDP sur <port> [service UDP] en utilisant UDP / IP version 4 ou 6 selon l'option pf. Il reçoit des paquets de plusieurs pairs non spécifiés et fusionne les données. Aucune réponse n'est possible. Cela fonctionne bien avec, par exemple, les homologues d'adresses UDP-SENDTO socat; il se comporte comme un serveur syslog.

Groupes d'options: FD, SOCKET, IP4, IP6, RANGE
Options utiles: fork, pf, bind, sourceport, ttl, tos
Voir aussi: UDP4-RECV, UDP6-RECV, UDP-SENDTO, UDP-RECVFROM, UDP-CONNECT, UDP-LISTEN, IP-RECV, UNIX-RECV

UDP4-RECV: <port>	Comme UDP-RECV, mais ne supporte que le protocole IPv4. `Groupes d'options: FD, SOCKET, IP4, RANGE`

UDP6-RECV: <port>	Comme UDP-RECV, mais ne supporte que le protocole IPv6. `Groupes d'options: FD, SOCKET, IP6, RANGE`

Les adresses UNIX

UNIX-CONNECT: <filename>

Se connecte à <filename> en supposant qu'il s'agit d'un socket de domaine UNIX. Si <filename> n'existe pas, il s'agit d'une erreur. si <filename> n'est pas un socket de domaine UNIX, il s'agit d'une erreur; Si <filename> est un socket de domaine UNIX, mais qu'aucun processus n'est à l'écoute, il s'agit d'une erreur.

Groupes d'options: FD, SOCKET, NAMED, RETRY, UNIX
Options utiles: bind
Voir aussi: UNIX-LISTEN, UNIX-SENDTO, TCP

UNIX-LISTEN: <filename>

Écoute <nomfichier> en utilisant un socket de domaine UNIX et accepte une connexion. Si <filename> existe et n'est pas un socket, il s'agit d'une erreur. Si <filename> existe et est une socket UNIX, la liaison à l'adresse échoue. Cette adresse est généralement bloquée jusqu'à ce qu'un client se connecte. Depuis la version 1.4.3, le système de fichiers est supprimé lorsque cette adresse est fermée (voir ci-dessous).

Groupes disponibles: FD, SOCKET, NAMED, LISTEN, CHILD, RETRY, UNIX
Options utiles: fork, umask, mode, user, group, unlink-early
Voir aussi: UNIX-CONNECT, UNIX-RECVFROM, UNIX-RECV, TCP-LISTEN

UNIX-SENDTO: <filename>

Communique avec la paire de socket spécifiée, définie par [<filename>] en supposant qu'il s'agit d'un socket de datagramme de domaine UNIX. Il envoie des paquets et reçoit des paquets de cette homologue uniquement. Il peut être nécessaire pour une socket locale à une adresse (par exemple / tmp / sock1, qui ne doit pas exister auparavant). Ce type d'adresse fonctionne bien avec les homologues de UNIX-RECVFROM et UNIX-RECV.

Groupes disponibles: FD, SOCKET, NAMED, UNIX
Options utiles: bind

Voir aussi: UNIX-RECVFROM, UNIX-RECV, UNIX-CONNECT, UDP-SENDTO, IP-SENDTO

UNIX-RECVFROM: <filename>

Un socket de datagramme de domaine UNIX [<nomfichier>]. Reçoit un paquet et peut envoyer un ou plusieurs paquets de réponses à cette paire. Ce mode est particulièrement utile avec une fork dans laquelle chaque paquet contient des arbitrages - est géré par son propre sous-processus. Cette adresse fonctionne bien avec les homologues de UNIX-SENDTO.

Options de groupes: FD, SOCKET, NAMED, CHILD, UNIX
Options utiles: fork
Voir aussi: UNIX-SENDTO, UNIX-RECV, UNIX-LISTEN, UDP-RECVFROM, IP-RECVFROM

UNIX-RECV: <filename>

Un socket de datagramme de domaine UNIX [<nomfichier>]. Reçoit les paquets de plusieurs homologues non spécifiés et fusionne les données. Aucune réponse n'est possible. Cela peut être, par exemple, adressé par les homologues de soc UNIX-SENDTO. Il existe un serveur Syslog. Groupes disponibles: FD, SOCKET, NAMED, UNIX

Voir aussi: UNIX-SENDTO, UNIX-RECVFROM, UNIX-LISTEN, UDP-RECV, IP-RECV

UNIX-CLIENT: <filename>

Communique avec le socket homologue spécifié, défini par [<filename>] en supposant qu'il s'agit d'un socket de domaine UNIX. Il essaie d'abord de se connecter et, en cas d'échec, suppose qu'il s'agit d'un socket datagramme, supportant ainsi les deux types.

Groupes d'options: FD, SOCKET, NAMED, UNIX
Options utiles: bind
Voir aussi: UNIX-CONNECT, UNIX-SENDTO, GOPEN

Les adresses ABSTRACT

Les adresses ABSTRACT sont presque identiques aux adresses UNIX associées, sauf si elles ne traitent

pas les sockets basées sur le système de fichiers mais un autre espace d'adressage de domaine UNIX. Pour archiver ceci, les chaînes de données sont préfixées avec “\ 0” en interne. This feature is available (only?) Sur Linux. Les groupes peuvent choisir parmi les adresses UNIX associées, sauf que les adresses ABSTRACT ne sont pas membres du groupe NAMED.

ABSTRACT-CONNECT: <chaîne>
ABSTRACT-LISTEN: <string>
ABSTRACT-SENDTO: <chaîne>
ABSTRACT-RECVFROM: <chaîne>
ABSTRACT-RECV: <chaîne>
ABSTRACT-CLIENT: <string>

Les autres types d'adresses

CREATE: <filename>

Ouvre <filename> avec creat() et utilise le descripteur de fichier pour écrire. Ce type d'adresse nécessite un contexte en écriture seule, car un fichier ouvert avec creat ne peut pas être lu.

Les drapeaux comme O_LARGEFILE ne peuvent pas être appliqués. Si on en a besoin, utiliser OPEN avec les options create, create.

<filename> doit être un chemin existant ou non existant. Si <filename> est un tube nommé, creat () pourrait bloquer; si <filename> fait référence à un socket, il s'agit d'une erreur.

Groupes d'options: FD, REG, NAMED
Options utiles: mode, user, group, unlink-early, unlink-late, append
Voir aussi: OPEN, GOPEN

EXEC: <ligne de commande>

fork un sous-processus qui établit la communication avec son processus parent et appelle le programme spécifié avec execvp (). <ligne de commande> est une commande simple avec des arguments séparés par des espaces simples. Si le nom du programme contient un '/', la partie après le dernier '/' est considérée comme ARGV [0]. Si le nom du programme est un chemin relatif, la sémantique execvp () pour trouver le programme via $ PATH s'applique. Après un démarrage réussi du programme, socat écrit les données dans stdin du processus et lit à partir de sa sortie standard via une socket de domaine UNIX générée par socketpair() par défaut. (Exemple)

Groupes d'options: FD, SOCKET, EXEC, FORK, TERMIOS
Options utiles: path, fdin, fdout, chroot, su, su-d, nofork, pty, stderr, ctty, setsid, pipes, login, sigint, sigquit Voir aussi: SYSTEM

FD: <fdnum>

Utilise le descripteur de fichier <fdnum>. Il doit déjà exister en tant que descripteur de fichier UNIX valide.

Groupes d'options: FD (TERMIOS, REG, SOCKET)
Voir aussi: STDIO, STDIN, STDOUT, STDERR

GOPEN: <filename>

(Generic open) Ce type d'adresse essaie de gérer toute entrée de système de fichiers à l'exception des répertoires utiles. <filename> peut être un chemin relatif ou absolu. S'il existe déjà, son type est vérifié. Dans le cas d'une socket de domaine UNIX, socat se connecte; Si la connexion échoue, socat suppose un socket datagramme et utilise les appels sendto (). Si l'entrée n'est pas un socket, socat l'ouvre en appliquant le drapeau O_APPEND. S'il n'existe pas, il est ouvert avec l'indicateur O_CREAT en tant que fichier normal (exemple).

Groupes d'options: FD, REG, SOCKET, NAMED, OPEN
Voir aussi: OPEN, CREATE, UNIX-CONNECT

Interface: <interface>

Communique avec un réseau connecté sur une interface en utilisant des paquets bruts, y compris des données de niveau de liaison. <interface> est le nom de l'interface réseau. Actuellement disponible uniquement sous Linux.

Groupes d'options: FD, SOCKET
Options utiles: pf, type
Voir aussi: ip-recv

OPEN: <filename>

Ouvre <nomfichier> à l'aide de l'appel système open () (exemple). Cette opération échoue sur les sockets de domaine UNIX.

Ce type d'adresse est rarement utile en mode bidirectionnel.

Groupes d'options: FD, REG, NAMED, OPEN
Options utiles: creat, excl, noatime, nofollow, append, rdonly, wronly, lock, readbytes, ignoreeof
Voir aussi: CREATE, GOPEN, UNIX-CONNECT

OPENSSL: <hôte>: <port>

Essaie d'établir une connexion SSL à <port> [service TCP] sur <hôte> [adresse IP] à l'aide de TCP / IP version 4 ou 6, selon la spécification d'adresse, la résolution de nom ou l'option pf.

Jusqu'à la version 1.7.2.4, la validité du certificat de serveur n'était vérifiée que par rapport au magasin de certificats système, à cafile ou à capath, mais pas au nom du serveur ou à son adresse IP. Depuis la version 1.7.3.0, socat vérifie que le certificat homologue correspond au paramètre <host> ou à la valeur de l'option openssl-commonname. Socat essaie de le faire correspondre aux objets du certificat commonName et à l'extension de certification subjectAltName. Les caractères génériques du certificat sont pris en charge.

Groupes d'options: FD, SOCKET, IP4, IP6, TCP, OPENSSL, RETRY
Options utiles: cipher, method, verify, commonname cafile, capath, certificate, key, compress, bind, pf, connect-timeout, sourceport, retry
Voir aussi: OPENSSL-LISTEN, TCP

OPENSSL-LISTEN: <port>

Ecoute tcp <port> [service TCP]. La version IP est 4 ou celle spécifiée avec pf. Lorsqu'une connexion est acceptée, cette adresse se comporte comme un serveur SSL.

Lorsqu'on veut utiliser l'option de certificat avec cette adresse. La validité du certificat client n'est vérifiée que pour cafile ou capath, mais pas pour le nom du client ou son adresse IP!

Groupes d'options: FD, SOCKET, IP4, IP6, TCP, LISTEN, OPENSSL, ENFANT, GAMME, RETRY
Options utiles: pf, cipher, method, verify, commonname cafile, capath, certificate, key, compress, fork, bind, range, tcpwrap, su, reuseaddr, retry

Voir aussi: OPENSSL, TCP-LISTEN

PIPE: <nomfichier>

Si <filename> existe déjà, il est ouvert. S'il n'existe pas, un canal nommé est créé et ouvert. À partir de la version 1.4.3 de socat, le canal nommé est supprimé lorsque l'adresse est fermée (mais voir l'option unlink-close)

Lorsqu'un tube est utilisé à la fois pour la lecture et l'écriture, il fonctionne comme un service d'écho.>

WRAP important>Lorsqu'un canal est utilisé à la fois pour la lecture et l'écriture et que socat essaie d'écrire plus d'octets que le tampon de canette (Linux 2.4: 2048 octets), socat peut bloquer. Envisager d'utiliser l'option socat, par exemple, -b 2048

Groupes d'options: FD, NAMED, OPEN
Options utiles: rdonly, nonblock, group, user, mode, unlink-early
Voir aussi: pipe anonyme

PIPE

Crée un canal sans nom et l'utilise pour la lecture et l'écriture. Cela fonctionne comme un écho, car tout ce qui lui est écrit apparaît immédiatement comme une donnée lue.

Lorsque socat essaie d'écrire plus d'octets que le tube ne peut mettre en file d'attente (Linux 2.4: 2048 octets), socat peut bloquer. Par exemple, en utilisant l'option -b 2048

Groupes d'option: FD
Voir aussi: pipe nommée

PROXY: <proxy>: <nomhôte>: <port>

Se connecte à un serveur proxy HTTP sur le port 8080 à l'aide de TCP / IP version 4 ou 6 en fonction de la spécification d'adresse, de la résolution de nom ou de l'option pf et envoie une demande

CONNECT pour hostname: port. Si le proxy accorde l'accès et réussit à se connecter à la cible, le transfert de données entre socat et la cible peut commencer. Le trafic n'a pas besoin d'être HTTP mais peut être un protocole arbitraire.

Groupes d'options: FD, SOCKET, IP4, IP6, TCP, HTTP, RETRY
Options utiles: proxyport, ignorecr, proxyauth, resolution, crnl, bind, connect-timeout, mss, sourceport, retry
Voir aussi: SOCKS, TCP

PTY

Génère un pseudo terminal (pty) et utilise son côté maître. Un autre processus peut ouvrir le côté esclave du pty en l'utilisant comme une ligne ou un terminal série. (Exemple). Si les deux mécanismes ptmx et openpty sont disponibles, ptmx est utilisé (POSIX).

Groupes d'options: FD, NAMED, PTY, TERMIOS
Options utiles: link, openpty, wait-slave, mode, user, group
Voir aussi: UNIX-LISTEN, PIPE, EXEC, SYSTEM

READLINE

Utilise readline GNU et l'historique sur stdio pour permettre l'édition et la réutilisation des lignes d'entrée (exemple).

En raison des restrictions de licence, la fonctionnalité readline est désactivée dans Debian. Voir BOGUES.

On peut utiliser STDIO à la place.

SOCKS4: <socks-server>: <hôte>: <port>

Se connecte via <socks-server> [adresse IP] à <hôte> [adresse IPv4] sur <port> [service TCP], en utilisant le protocole socks version 4 sur IP version 4 ou 6 selon la spécification d'adresse, la résolution de nom ou l'option pf (Exemple).

Groupes d'options: FD, SOCKET, IP4, IP6, TCP, SOCKS4, RETRY
Options utiles: socksuser, socksport, sourceport, pf, retry
Voir aussi: SOCKS4A, PROXY, TCP

SOCKS4A: <serveur-chaussettes>: <hôte>: <port>

comme SOCKS4, mais utilise le protocole Socks version 4a, laissant ainsi la résolution du nom d'hôte au serveur Socks.

Groupes d'options: FD, SOCKET, IP4, IP6, TCP, SOCKS4, RETRY

STDERR

Utilise le descripteur de fichier 2.

Groupes d'options: FD (TERMIOS, REG, SOCKET)
Voir aussi: FD

STDIN

Utilise le descripteur de fichier 0.

Groupes d'options: FD (TERMIOS, REG, SOCKET)
Options utiles: readbytes
Voir aussi: FD

STDIO

Utilise le descripteur de fichier 0 pour la lecture et 1 pour l'écriture.

Groupes d'options: FD (TERMIOS, REG, SOCKET)
Options utiles: readbytes
Voir aussi: FD

STDOUT

Utilise le descripteur de fichier 1.

Groupes d'options: FD (TERMIOS, REG, SOCKET)
Voir aussi: FD

SYSTEM: <shell-command>

fork un sous-processus qui établit la communication avec son processus parent et appelle le programme spécifié avec system (). <shell-command> [string] ne doit pas contenir “,” ou “!!” et que les méta-caractères du shell doivent être protégés. Après le démarrage du programme, socat écrit les données dans stdin du processus et lit à partir de sa sortie standard.

Groupes d'options: FD, SOCKET, EXEC, FORK, TERMIOS
Options utiles: chemin, fdin, fdout, chroot, su, su-d, nofork, pty, stderr, ctty, setsid, pipes, sigint, sigquit
Voir aussi: EXEC

TUN [: <if-addr> / <bits>]

Crée un périphérique TUN / TAP Linux et l'assigne éventuellement l'adresse et le masque de réseau fournis par les paramètres. L'interface réseau qui en résulte est presque prête à être utilisée par d'autres processus. socat sert son “côté fil”. Cette adresse nécessite un accès en lecture et en écriture au périphérique de clonage du tunnel, généralement / dev / net / tun, ainsi que la permission de définir des ioctl () s. Option iff-up est nécessaire pour activer immédiatement l'interface!

Groupes d'options: FD, NAMED, OPEN, TUN
Options utiles: iff-up, tun-device, nom-tun, type-tun, iff-no-pi
Voir aussi: ip-recv

Les groupes d'options

Groupe d'options FD

Ce groupe d'options contient des options appliquées à un descripteur de fichier de style UN * X, quelle que soit sa génération. Étant donné que tous les types d'adresse socat actuels sont basés sur un descripteur de fichier, ces options peuvent être appliquées à n'importe quelle adresse.

Certaines de ces options sont également membres d'un autre groupe d'options, qui

fournit un autre mécanisme non basé sur fd. Pour ces options, cela dépend du type d'adresse réel et de ses groupes d'options, quel mécanisme est utilisé. Le second mécanisme, non basé sur fd, est prioritaire.

cloexec = <bool>	Définit l'indicateur FD_CLOEXEC avec l'appel système fcntl () à la valeur <bool>. S'il est défini, le descripteur de fichier est fermé sur les appels de fonction de la famille exec (). Socat gère cet indicateur en interne pour le fds qu'il contrôle, donc dans la plupart des cas, il ne sera pas nécessaire d'appliquer cette option.
setlk	Essaie de définir un verrou en écriture discrétionnaire sur l'ensemble du fichier en utilisant l'appel système fcntl (fd, F_SETLK, ...). Si le fichier est déjà verrouillé, cet appel génère une erreur. Sous Linux, lorsque les autorisations de fichier pour le groupe sont “S” (gx, g + s) et que le système de fichiers est monté localement avec l'option “mand”, le verrouillage est obligatoire, empêchant ainsi les autres processus d'ouvrir le fichier.
setlkw	Essaie de définir un verrou en écriture discrétionnaire en attente sur l'ensemble du fichier à l'aide de l'appel système fcntl (fd, F_SETLKW, ...). Si le fichier est déjà verrouillé, cet appel est bloqué. Voir l'option setlk pour plus d'informations sur la nécessité de rendre ce verrou obligatoire.
setlk-rd	Essaie de définir un verrou de lecture discrétionnaire pour tout le fichier en utilisant l'appel système fcntl (fd, F_SETLK, ...). Si le fichier est déjà verrouillé en écriture, cet appel génère une erreur. Voir l'option setlk pour plus d'informations sur la nécessité de rendre ce verrou obligatoire.
setlkw-rd	Essaie de définir un verrou de lecture en attente discrétionnaire sur tout le fichier à l'aide de l'appel système fcntl (fd, F_SETLKW, ...). Si le fichier est déjà verrouillé en écriture, cet appel est bloqué. Voir l'option setlk pour plus d'informations sur la nécessité de rendre ce verrou obligatoire.
flock-ex	Essaie de définir un verrou de conseil exclusif bloquant sur le fichier à l'aide de l'appel système flock (fd, LOCK_EX). Socat se bloque dans cet appel si le fichier est verrouillé par un autre processus.
flock-ex-nb	Essaie de définir un verrou de conseil exclusif nonblock sur le fichier à l'aide de l'appel système flock (fd, LOCK_EX / LOCK_NB). Si le fichier est déjà verrouillé, cette option entraîne une erreur.
flock-sh	Essaie de définir un verrou consultatif partagé bloquant sur le fichier à l'aide de l'appel système flock (fd, LOCK_SH). Socat se bloque dans cet appel si le fichier est verrouillé par un autre processus.
flock-sh-nb	Essaie de définir un verrou de conseil partagé nonblock sur le fichier à l'aide de l'appel système flock (fd, LOCK_SH / LOCK_NB). Si le fichier est déjà verrouillé, cette option génère une erreur.
lock	Définit un verrou de blocage sur le fichier. Utilise le mécanisme setlk ou flock en fonction de la disponibilité de la plate-forme concernée. Si les deux sont disponibles, la variante POSIX (setlkw) est utilisée.
user = <utilisateur>	Définit le <utilisateur> (propriétaire) du flux. Si l'adresse est membre du groupe d'options NAMED , socat utilise l'appel système chown () après l'ouverture du fichier ou la liaison au socket du domaine UNIX (condition de concurrence !). Sans entrée dans le système de fichiers, socat définit l'utilisateur du flux à l'aide de l'appel système fchown (). Ces appels peuvent nécessiter des privilèges root.
user-late = <utilisateur>	Définit le propriétaire du fd sur <user> avec l'appel système fchown () après ouverture ou connexion du canal. Ceci est utile uniquement sur les entrées du système de fichiers.

group = <groupe>	Définit le <groupe> du flux. Si l'adresse est membre du groupe d'options NAMED , socat utilise l'appel système chown () après l'ouverture du fichier ou la liaison au socket du domaine UNIX (condition de concurrence !). Sans entrée dans le système de fichiers, socat définit le groupe du flux avec l'appel système fchown (). Ces appels peuvent nécessiter une appartenance à un groupe ou des privilèges root.
group-late = <groupe>	Définit le groupe du fd sur <group> avec l'appel système fchown () après ouverture ou connexion du canal. Ceci est utile uniquement sur les entrées du système de fichiers.
mode = <mode>	Définit le <mode> [mode_t] (permissions) du flux. Si l'adresse est membre du groupe d'options NAMED et utilise l'appel open () ou creat (), le mode est appliqué avec ceux-ci. Si l'adresse est membre du groupe d'options NAMED sans utiliser ces appels système, socat utilise l'appel système chmod () après avoir ouvert l'entrée du système de fichiers ou la liaison avec le socket du domaine UNIX (condition de concurrence !). Sinon, socat définit le mode du flux en utilisant fchmod (). Ces appels peuvent nécessiter la propriété ou les privilèges root.
perm-late = <mode>	Définit les autorisations de la valeur fd sur <mode> [mode_t] à l'aide de l'appel système fchmod () après ouverture ou connexion du canal. Ceci est utile uniquement sur les entrées du système de fichiers.
append = <bool>	Toujours écrire des données à la fin du fichier. Si l'adresse est membre du groupe d'options OPEN, socat utilise l'indicateur O_APPEND avec l'appel système open () (exemple). Sinon, socat applique l'appel fcntl (fd, F_SETFL, O_APPEND).
nonblock = <bool>	Essaie d'ouvrir ou d'utiliser un fichier en mode nonblock. Ses seuls effets sont que l'appel connect () des adresses TCP ne bloque pas et que l'ouverture d'un canal nommé pour la lecture ne bloque pas. Si l'adresse est membre du groupe d'options OPEN, socat utilise l'indicateur O_NONBLOCK avec l'appel système open (). Sinon, socat applique l'appel fcntl (fd, F_SETFL, O_NONBLOCK).
binary	Ouvre le fichier en mode binaire pour éviter les conversions de terminaison de ligne implicites (Cygwin).
text	Ouvre le fichier en mode texte pour forcer les conversions de terminaison de ligne implicites (Cygwin).
noinherit	Ne conserve pas ce fichier ouvert dans un processus généré (Cygwin).
cool-write	Simplifie l'écriture en cas d'échec avec EPIPE ou ECONNRESET et enregistre le message avec le niveau de notification au lieu de l' erreur . Cela évite que le fichier journal ne soit rempli de messages d'erreur inutiles lorsque socat est utilisé comme serveur ou proxy à haut volume où les clients abandonnent souvent la connexion. Cette option est expérimentale.
end-close	Modifie la méthode (dépendante de l'adresse) de la fin d'une connexion pour fermer les descripteurs de fichiers. Ceci est utile lorsque la connexion doit être réutilisée par ou partagée avec d'autres processus (exemple). Normalement, les connexions de socket se termineront par shutdown qui termine le socket même s'il est partagé par plusieurs processus. close “dissocie” le socket du processus mais le maintient actif tant qu'il existe encore des liens provenant d'autres processus. De même, lorsqu'une adresse de type EXEC ou SYSTEM est terminée, socat va généralement tuer explicitement le sous-processus. Avec cette option, il suffit de fermer les descripteurs de fichiers.
shut-none	Modifie la méthode (dépendante de l'adresse) d'arrêt de la partie écriture d'une connexion pour ne rien faire.

shut-down	
shut-close	Modifie la méthode (dépendante de l'adresse) d'arrêt de la partie écriture d'une connexion pour fermer \ (fd).
shut-null	Lorsqu'une adresse indique EOF, socat envoie un paquet de taille nulle au canal d'écriture de l'autre adresse pour transférer la condition EOF. Ceci est utile avec UDP et d'autres protocoles de datagramme. A été testé contre netcat et socat avec l'option null-eof.
null-eof	Normalement, socat ignorera les paquets vides (charge utile de taille nulle) arrivant sur les sockets de datagramme, donc il survit aux analyses de port. Avec cette option, socat interprète les paquets de datagrammes vides comme un indicateur EOF (voir shut-null).
ioctl-void = <demande>	Appelle ioctl() avec la valeur de la requête comme deuxième argument et NULL comme troisième argument. Cette option permet d'utiliser des ioctls qui ne sont pas explicitement implémentés dans socat.
ioctl-int = <demande>: <valeur>	Appelle ioctl() avec la valeur de la requête comme second argument et la valeur entière comme troisième argument.
ioctl-intp = <demande>: <valeur>	Appelle ioctl() avec la valeur de la requête comme deuxième argument et un pointeur sur la valeur entière comme troisième argument.
ioctl-bin = <demande>: <valeur>	Appelle ioctl() avec la valeur de la requête comme second argument et un pointeur sur la valeur de donnée donnée comme troisième argument. Ces données doivent être spécifiées sous forme <dalan>.
ioctl-string = <demande>: <valeur>	Appelle ioctl() avec la valeur de la requête comme second argument et un pointeur sur la chaîne donnée comme troisième argument. <dalan> forme.

Groupe d'options NAMED

Ces options fonctionnent sur les entrées du système de fichiers.

Voir aussi les options user, group et mode.

user-early = <utilisateur>	Modifie le <utilisateur> (propriétaire) de l'entrée du système de fichiers avant d'y accéder, en utilisant l'appel système chown (). Cet appel peut nécessiter des privilèges root.
group-early = <groupe>	Modifie le <groupe> de l'entrée du système de fichiers avant d'y accéder, en utilisant l'appel système chown (). Cet appel peut nécessiter une appartenance à un groupe ou des privilèges root.
perm-early = <mode>	Modifie le <mode> [mode_t] de l'entrée du système de fichiers avant d'y accéder, en utilisant l'appel système chmod (). Cet appel peut nécessiter la propriété ou les privilèges root.
umask = <mode>	Définit l'umask du processus sur <mode> [mode_t] avant d'accéder à l'entrée du système de fichiers (utile avec les sockets de domaine UNIX!). Cet appel peut affecter toutes les opérations ultérieures du processus socat !
unlink-early	Unlinks (supprime) le fichier avant de l'ouvrir et même avant d'appliquer user-early etc.
unlink	Unlinks (supprime) le fichier avant d'y accéder, mais après le début de l'utilisateur, etc.
unlink-late	Unlinks (supprime) le fichier après l'avoir ouvert pour le rendre inaccessible pour d'autres processus après une courte condition de course.

unlink-close	Supprime l'entrée du système de fichiers d'adresses lors de la fermeture de l'adresse. Pour les canaux nommés, l'écoute des sockets de domaine unix et les liens symboliques des adresses pty, la valeur par défaut est 1; pour les fichiers créés, les fichiers ouverts, les fichiers ouverts génériques et les sockets du domaine unix client, la valeur par défaut est 0.

Groupe d'options OPEN

Les options du groupe OPEN permettent de définir des indicateurs avec l'appel système open (). Par exemple l'option `creat` définit le drapeau O_CREAT.

Voir aussi les options append et nonblock.

creat = <bool>	Crée le fichier s'il n'existe pas (exemple).
dsync = <bool>	Bloque les appels à write() jusqu'à ce que metainfo soit physiquement écrit sur le média.
excl = <bool>	Avec l'option creat, si le fichier existe, c'est une erreur.
largefile = <bool>	Sur les systèmes 32 bits, autorise un fichier supérieur à 2 ^ 31 octets.
noatime	Définit les options O_NOATIME, donc les lectures ne modifient pas l'horodatage d'accès.
noctty = <bool>	Ne fait pas de ce fichier le terminal de contrôle.
nofollow = <bool>	Ne suit pas les liens symboliques.
nshare = <bool>	Ne permet pas de partager ce fichier avec d'autres processus.
rshare = <bool>	Ne permet pas aux autres processus d'ouvrir ce fichier pour l'écriture.
rsync = <bool>	Bloque write() jusqu'à ce que metainfo soit écrit physiquement sur le média.
sync = <bool>	Bloque write() jusqu'à ce que les données soient écrites physiquement sur le média.
rdonly = <bool>	Ouvre le fichier pour la lecture uniquement.
wronly = <bool>	Ouvre le fichier pour l'écriture seulement.
trunc	Tronque le fichier à la taille 0 lors de son ouverture.

Groupe d'options REG et BLK

Ces options sont généralement appliquées à un descripteur de fichier UN * X, mais leur sémantique n'a de sens que pour un fichier prenant en charge l'accès aléatoire.

seek = <offset>	Applique l'appel système lseek (fd, <offset>, SEEK_SET) (ou lseek64), positionnant ainsi le pointeur de fichier sur <offset> [off_t ou off64_t]. La valeur par défaut est 1, et non 0.
seek-cur = <offset>	Applique l'appel système lseek (fd, <offset>, SEEK_CUR) (ou lseek64), positionnant ainsi le pointeur de fichier <offset> [off_t ou off64_t] octets par rapport à sa position actuelle (qui est généralement 0). La valeur par défaut est 1, et non 0.
seek-end = <offset>	Applique l'appel système lseek (fd, <offset>, SEEK_END) (ou lseek64), positionnant ainsi le pointeur de fichier <offset> [off_t ou off64_t] octets par rapport à la fin actuelle des fichiers. La valeur par défaut est 1, et non 0.
ftruncate = <offset>	Applique l'appel système ftruncate (fd, <offset>) (ou ftruncate64 si disponible), tronquant ainsi le fichier à la position <offset> [off_t ou off64_t]. La valeur par défaut est 1, et non 0.
secret = <bool>	

unrm = <bool>	
compr = <bool>	
ext2-sync = <bool>	
immuable = <bool>	
ext2-append = <bool>	
nodump = <bool>	
ext2-noatime = <bool>	
journal-data = <bool>	
notail = <bool>	
dirsync = <bool>	

Ces options modifient les attributs de fichier non standard sur les systèmes d'exploitation et les systèmes de fichiers prenant en charge ces fonctionnalités, comme Linux avec ext2fs, ext3fs ou reiserfs. Voir man 1 chattr pour plus d'informations sur ces options. Il peut y avoir une situation de concurrence entre la création du fichier et l'application de ces options.

Groupe d'options PROCESS

Les options de ce groupe modifient les propriétés du processus au lieu d'affecter un seul canal de données. Pour les adresses EXEC et SYSTEM et pour les adresses de type LISTEN et CONNECT avec l'option FORK, ces options s'appliquent aux processus enfants au lieu du processus socat principal.

chroot = <répertoire>	Effectue une opération chroot() sur <directory> après avoir traité l'adresse (exemple). Cet appel peut nécessiter des privilèges root.
chroot-early = <répertoire>	Effectue une opération chroot() sur <directory> avant d'ouvrir l'adresse. Cet appel peut nécessiter des privilèges root.
setgid = <groupe>	Modifie le <groupe> principal du processus après avoir traité l'adresse. Cet appel peut nécessiter des privilèges root. Cette option ne supprime pas les privilèges liés aux autres groupes.
setgid-early = <groupe>	Comme setgid mais est effectué avant d'ouvrir l'adresse.
setuid = <utilisateur>	Modifie le <utilisateur> (propriétaire) du processus après avoir traité l'adresse. Cet appel peut nécessiter des privilèges root. Cette option ne supprime pas les privilèges liés au groupe. Vérifier si l'option su correspond mieux à vos besoins.
setuid-early = <utilisateur>	Comme setuid mais est effectué avant d'ouvrir l'adresse.
su = <utilisateur>	Modifie le <utilisateur> (propriétaire) et les groupes du processus après avoir traité l'adresse (exemple). Cet appel peut nécessiter des privilèges root.
su-d = <utilisateur>	Nom abrégé de sous-utilisateur retardé. Modifie le <utilisateur> (propriétaire) et les groupes du processus après avoir traité l'adresse (exemple). L'utilisateur et ses groupes sont récupérés avant un éventuel chroot (). Cet appel peut nécessiter des privilèges root.
setpgid = <pid_t>	Rend le processus membre du groupe de processus spécifié <pid_t>. Si aucune valeur n'est donnée ou si la valeur est 0 ou 1, le processus devient le leader d'un nouveau groupe de processus.

setsid	Fait du processus le leader d'une nouvelle session (exemple).

Groupe d'option READLINE

En raison de restrictions de licence, la fonctionnalité readline est désactivée dans Debian (voir BOGUES).

Ces options s'appliquent au type d'adresse readline.

history = <filename>	Lit et écrit l'historique de / vers <nomfichier> (exemple).
noprompt	Depuis la version 1.4.0, socat per default essaie de déterminer une invite - qui est ensuite transmise à l'appel readline - en mémorisant la dernière ligne incomplète de la sortie. Avec cette option, socat ne passe pas d'invite à readline, il commence donc à éditer les lignes dans la première colonne du terminal.
noecho = <pattern>	Spécifie un modèle régulier pour une invite qui empêche l'affichage de la ligne de saisie suivante à l'écran et son ajout à l'historique. L'invite est définie comme le texte qui a été généré à l'adresse readline après le dernier caractère de nouvelle ligne et avant la saisie d'un caractère de saisie. Le modèle est une expression régulière, par exemple `"^ [Pp] assword:. * $" Ou "([Uu] ser: | [Pp] assword :)"`. Voir regex \ (7) pour plus de détails. (Exemple)
prompt = <chaîne>	Transmet la chaîne en tant qu'invite à la fonction readline. readline imprime cette invite lors de la lecture de l'historique. Si cette chaîne correspond à une invite constante émise par un programme interactif sur l'autre adresse socat, un aspect cohérent peut être créé.

Groupe d'options APPLICATION

Ce groupe contient des options qui fonctionnent au niveau des données. Ces options ne s'appliquent qu'aux données “brutes” transférées par socat, mais pas aux données de protocole utilisées par des adresses comme PROXY.

cr	Convertit le caractère de terminaison de ligne par défaut NL ('\ n', 0x0a) en / de CR ('\ r', 0x0d) lors de l'écriture / lecture sur ce canal.
crnl	Convertit le caractère de terminaison de ligne par défaut NL ('\ n', 0x0a) en / de CRNL (“\ r \ n”, 0x0d0a) lors de l'écriture / lecture sur ce canal (exemple). socat supprime simplement tous les caractères CR.
ignoreeof	Lorsque EOF se produit sur ce canal, socat l' ignore et tente de lire plus de données (comme “tail -f”) (exemple).
readbytes = <octets>	socat ne lit que le nombre d'octets de cette adresse (l'adresse ne fournit que le nombre d'octets nécessaires au transfert et fait semblant d'être à la fin du processus). Doit être supérieur à 0.
lockfile = <nomfichier>	Si le fichier de verrouillage existe, quitte avec erreur. Si le fichier de verrouillage n'existe pas, le crée et continue, dissocie le fichier de verrouillage à la sortie.
waitlock = <nomfichier>	Si le fichier de verrouillage existe, attend qu'il disparaisse. Lorsque le fichier de verrouillage n'existe pas, le crée et continue, supprime le fichier de verrouillage à la sortie.
escape = <int>	Spécifie le code numérique d'un caractère qui déclenche EOF sur le flux d'entrée. Il est utile avec un terminal en mode brut (exemple).

Groupe d'options SOCKET

Ces options sont destinées à tous les types de sockets, par exemple IP ou UNIX. La plupart sont appliqués avec un appel à setsockopt ().

bind = <sockname>	Lie le socket à l'adresse de socket donnée en utilisant l'appel système bind (). La forme de <sockname> dépend du domaine socket: IP4 et IP6 autorisent la forme `[hostname|hostaddress][:(service|port]]` (exemple), les sockets de domaine UNIX nécessitent <filename>.
connect-timeout = <secondes>	Abandonne la tentative de connexion après <secondes> [timeval] avec le statut d'erreur.
so-bindtodevice = <interface>	Lie le socket à la <interface> donnée. Cette option peut nécessiter des privilèges root.
broadcast	
debug	Active le débogage du socket.
dontroute	Ne communique qu'avec des homologues directement connectés, n'utilise pas de routeurs.
keepalive	Permet d'envoyer des keepalives sur le socket.
linger = <secondes>	Bloque shutdown() ou close() jusqu'à la fin des transferts de données ou l'expiration du délai imparti [int].
oobinline	Place des données hors bande dans le flux de données en entrée.
priority = <priorité>	Définit le protocole défini par <priority> [<int>] pour les paquets sortants.
rcvbuf = <octets>	Définit la taille du tampon de réception après l'appel de socket () à <bytes> [int]. Avec les sockets TCP, cette valeur correspond à la taille de fenêtre maximale de la socket.
rcvbuf-late = <octets>	Définit la taille du tampon de réception lorsque le socket est déjà connecté à <octets> [int]. Avec les sockets TCP, cette valeur correspond à la taille de fenêtre maximale de la socket.
rcvlowat = <octets>	Spécifie le nombre minimal d'octets reçus [int] jusqu'à ce que la couche socket transmette les données mises en mémoire tampon à socat.
rcvtimeo = <secondes>	Définit le délai de réception [timeval].
reuseaddr	Permet à d'autres sockets de se lier à une adresse même si certaines parties (par exemple le port local) sont déjà utilisées par socat (exemple).
sndbuf = <octets>	Définit la taille du tampon d'envoi après l'appel de socket () à <bytes> [int].
sndbuf-late = <octets>	Définit la taille du tampon d'envoi lorsque le socket est connecté à <octets> [int].
sndlowat = <octets>	Spécifie le nombre minimum d'octets dans le tampon d'envoi jusqu'à ce que la couche socket envoie les données à <octets> [int].
sndtimeo = <secondes>	Définit le délai d'attente d'envoi en secondes [timeval].
pf = <chaîne>	

Force l'utilisation de la version ou du protocole IP spécifié. <string> peut être quelque chose comme “ip4” ou “ip6”. La valeur résultante est utilisée comme premier argument pour les appels à socket() ou à socketpair(). Cette option affecte la résolution des adresses et la syntaxe requise pour les options de liaison et de plage. |

type = <type>	Définit le type de socket, spécifié comme second argument pour les appels à socket () ou socketpair (), à <type> [int]. La résolution d'adresse n'est pas affectée par cette option. Sous Linux, 1 signifie socket orienté flux, 2 signifie socket datagramme et 3 signifie socket brut.
prototype	Définit le protocole de la socket, spécifié comme troisième argument pour les appels socket () ou socketpair (), à <prototype> [int]. La résolution d'adresse n'est pas affectée par cette option. 6 signifie TCP, 17 signifie UDP.
so-timestamp	Définit l'option de socket SO_TIMESTAMP. Cela permet de recevoir et d'enregistrer des messages auxiliaires d'horodatage.
setsockopt-int = <niveau>: <optname>: <optval>	Invoque setsockopt() pour le socket avec les paramètres donnés. level [int] est utilisé comme second argument de setsockopt () et spécifie la couche, par exemple SOL_TCP pour TCP (6 sous Linux) ou SOL_SOCKET pour la couche socket (1 sous Linux). optname [int] est le troisième argument de setsockopt () et indique quelle option de socket doit être définie. Pour les nombres réels, il faudra peut-être rechercher les fichiers d'inclusion appropriés de votre système. Le 4ème paramètre setsockopt(), valeur [int], est passé à la fonction par pointeur et pour la longueur, le paramètre sizeof \ (int) est pris implicitement.
setsockopt-bin = <niveau>: <optname>: <optval>	Comme setsockopt-int, mais <optval> doit être fourni au format dalan et spécifie une séquence arbitraire d'octets; le paramètre de longueur est automatiquement dérivé des données.
setsockopt-string = <niveau>: <optname>: <optval>	Comme setsockopt-int, mais <optval> doit être une chaîne. Cette chaîne est transmise à la fonction avec un caractère nul final et le paramètre de longueur est automatiquement dérivé des données.

Groupe d'options UNIX

Ces options s'appliquent aux adresses basées sur un domaine UNIX.

unix-tightsocklen = [0/1]	Sur les opérations de socket, transmet une longueur d'adresse de socket qui n'inclut pas l'intégralité de l'enregistrement struct sockaddr_un mais (en plus des autres composants) uniquement la partie pertinente du nom de fichier ou de la chaîne abstraite. La valeur par défaut est 1.

Groupes d'options IP4 et IP6

Ces options peuvent être utilisées avec les sockets IPv4 et IPv6.

tos = <tos>	Définit le champ TOS (type de service) des paquets sortants sur <tos> [octet] (voir RFC 791).
ttl = <ttl>	Définit le champ TTL (time to live) des paquets sortants sur <ttl> [octet].
ip-options = <données>	Définit les options IP comme le routage source. Doit être donné sous forme binaire, le format recommandé est un “x” précédé d'un nombre pair de chiffres hexadécimaux. Cette option peut être utilisée plusieurs fois, les données sont ajoutées. Par exemple, pour vous connecter à l'hôte 10.0.0.1 via une passerelle en utilisant une route source libre, utiliser la passerelle comme paramètre d'adresse et définir une route source libre en utilisant l'option ip-options = x8307040a000001. Les options IP sont définies dans la RFC 791.
mtudiscover = <0/1/2>	Prend 0, 1, 2 pour ne jamais vouloir, ou utiliser toujours le chemin MTU découvrir sur ce socket.

ip-pktinfo	Définit l'option de socket IP_PKTINFO. Cela permet de recevoir et de consigner des messages auxiliaires contenant l'adresse de destination et l'interface (Linux) (exemple).
ip-recverr	Définit l'option de socket IP_RECVERR. Cela permet de recevoir et d'enregistrer des messages auxiliaires contenant des informations d'erreur détaillées.
ip-recvopts	Définit l'option de socket IP_RECVOPTS. Cela permet de recevoir et de consigner les messages auxiliaires des options IP (Linux, * BSD).
ip-recvtos	Définit l'option de socket IP_RECVTOS. Cela permet de recevoir et de consigner les messages auxiliaires TOS (type de service) (Linux).
ip-recvttl	Définit l'option de socket IP_RECVTTL. Cela permet de recevoir et de consigner les messages auxiliaires TTL (Time to Live) (Linux, * BSD).
ip-recvdstaddr	Définit l'option de socket IP_RECVDSTADDR. Cela permet de recevoir et d'enregistrer des messages auxiliaires contenant l'adresse de destination (* BSD) (exemple).
ip-recvif	Définit l'option de socket IP_RECVIF. Cela permet de recevoir et d'enregistrer des messages auxiliaires d'interface (* BSD) (exemple).
ip-add-membership	Crée le membre socket du groupe multicast spécifié. Ceci est actuellement uniquement implémenté pour IPv4. L'option prend l'adresse IP du groupe de multidiffusion et des informations sur l'interface réseau souhaitée. Les syntaxes admises sont les suivantes: **ip-add-membership = <adresse multicast: adresse-interface>** **ip-add-membership = <adresse multicast: nom-interface>** **ip-add-membership = <adresse multicast: interface-index>** **ip-add-membership = <adresse multicast: adresse-interface: nom-interface>** **ip-add-membership = <adresse multicast: adresse-interface: interface-index>** La syntaxe la plus courante est la première, tandis que les autres ne sont disponibles que sur les systèmes qui fournissent struct mreqn (Linux). Les indices des interfaces réseau actives peuvent être affichés à l'aide de l'utilitaire procan .
ip-multicast-if = <nom d'hôte>	Spécifie le nom d'hôte ou l'adresse de l'interface réseau à utiliser pour le trafic multidiffusion.
ip-multicast-loop = <bool>	Spécifie si le trafic multidiffusion sortant doit retourner à l'interface.
ip-multicast-ttl = <octet>	Définit le TTL utilisé pour le trafic multicast sortant. La valeur par défaut est 1.
res-debug	Ces options définissent les options de résolution correspondantes (résolution de noms). Ajouter “= 0” pour effacer une option par défaut. Voir homme resolver \ (5) pour plus d'informations sur ces options. ces options ne sont valables que pour l'adresse à laquelle elles sont appliquées.
res-aaonly	
res-usevc	
res-primaire	
res-igntc	
res-recurse	
res-defnames	
res-stayopen	
res-dnsrch	

Groupe d'option IP6

Ces options ne peuvent être utilisées que sur des sockets IPv6. Voir les options IP pour les options pouvant être appliquées aux sockets IPv4 et IPv6.

ipv6only = <bool>	Définit l'option de socket IPV6_V6ONLY. Si 0, la pile TCP acceptera également les connexions utilisant le protocole IPv4 sur le même port. La valeur par défaut dépend du système.
ipv6-recvdstopts	Définit l'option de socket IPV6_RECVDSTOPTS. Cela permet de recevoir et de consigner les messages auxiliaires contenant les options de destination.
ipv6-recvhoplimit	Définit l'option de socket IPV6_RECVHOPLIMIT. Cela permet de recevoir et de consigner les messages auxiliaires contenant le hoplimit.
ipv6-recvhopopts	Définit l'option de socket IPV6_RECVHOPOPTS. Cela permet de recevoir et de consigner les messages auxiliaires contenant les options de saut.
ipv6-recvpktinfo	Définit l'option de socket IPV6_RECVPKTINFO. Cela permet de recevoir et de consigner des messages auxiliaires contenant l'adresse de destination et l'interface.
ipv6-unicast-hops = link (TYPE_INT) (<int>)	Définit l'option de socket IPV6_UNICAST_HOPS. Cela définit la limite du nombre de sauts (TTL) pour les paquets de monodiffusion sortants.
ipv6-recvrthdr	Définit l'option de socket IPV6_RECVRTHDR. Cela permet de recevoir et d'enregistrer des messages auxiliaires contenant des informations de routage.
ipv6-tclass	Définit l'option de socket IPV6_TCLASS. Cela définit la classe de transfert des paquets sortants.
ipv6-recvtclass	Définit l'option de socket IPV6_RECVTCLASS. Cela permet de recevoir et de consigner les messages auxiliaires contenant la classe de transfert.

Groupe d'options TCP

Ces options peuvent être appliquées aux sockets TCP. Ils fonctionnent en appelant setsockopt () avec les paramètres appropriés.

cork	N'envoie pas de paquets plus petits que MSS (taille maximale du segment).
defer-accept	En écoutant, accepte les connexions uniquement lorsque les données du pair sont arrivées.
keepcnt = <nombre>	Définit le nombre de keepalives avant de fermer le socket sur <count> [int].
keepidle = <secondes>	Définit le temps d'inactivité avant d'envoyer le premier keepalive à <secondes> [int].
keepintvl = <secondes>	Définit l'intervalle entre deux keepalives à <secondes> [int].
linger2 = <secondes>	Définit l'heure de maintien du socket dans l'état FIN-WAIT-2 sur <secondes> [int].
mss = <octets>	Définit le MSS (taille maximale du segment) après l'appel de socket () à <octets> [int]. Cette valeur est ensuite proposée au pair avec le paquet SYN ou SYN / ACK (exemple).
mss-late = <octets>	Définit le MSS du socket après que la connexion a été établie à <octets> [int].
nodelay	Désactive l'algorithme Nagle pour mesurer le RTT (temps d'aller-retour).

rfc1323	Active les options TCP RFC1323: échelle de la fenêtre TCP, mesure du temps d'aller-retour (RTTM) et protection contre les numéros de séquence enveloppés (PAWS) (AIX).
stdurg	Active la gestion du pointeur urgent conforme à la RFC1122 (AIX).
syncnt = <nombre>	Définit le nombre maximal de retransmissions SYN pendant la connexion à <count> [int].
md5sig	Permet de générer des résumés MD5 sur les paquets (FreeBSD).
notake	Désactive l'utilisation des options TCP (FreeBSD, MacOSX).
nopush	définit l'option de socket TCP_NOPUSH (FreeBSD, MacOSX).
sack-disable	Désactive l'utilisation de la fonctionnalité d'acquittement sélectif (OpenBSD).
signature-enable	Permet la génération de fichiers MD5 sur les paquets (OpenBSD).
abort-threshold = <millisecondes>	Définit le délai d'attente d'une réponse de l'homologue sur une connexion établie (HP-UX).
conn-abort-threshold = <millisecondes>	Définit le délai d'attente d'une réponse du serveur lors de la connexion initiale (HP-UX).
keepinit	Définit le délai d'attente d'une réponse du serveur lors de la connexion \ () avant d'abandonner. Valeur en demi-secondes, la valeur par défaut est 150 (75s) (Tru64).
paws	Active la fonctionnalité “protéger contre les numéros de séquence encapsulés” (Tru64).
sackena	Active l'acquittement sélectif (Tru64).
tsoptena	Active l'option d'horodatage qui permet le recalcul RTT sur les connexions existantes (Tru64).

Groupe d'options SCTP

Ces options peuvent être appliquées aux sockets de flux SCTP.

sctp-nodelay	Définit l'option de socket SCTP_NODELAY qui désactive l'algorithme Nagle.
sctp-maxseg = <octets>	Définit l'option de socket SCTP_MAXSEG sur <octets> [int]. Cette valeur est ensuite proposée au pair avec le paquet SYN ou SYN / ACK.

Groupes d'options UDP, TCP et SCTP

Nous trouvons ici des options liées au mécanisme du port réseau et qui peuvent donc être utilisées avec les adresses client et serveur UDP, TCP et SCTP.

sourceport = <port>	Pour les connexions sortantes (client) TCP et UDP, il définit la source <port> à l'aide d'un appel bind () supplémentaire. Avec les adresses d'écoute TCP ou UDP, socat ferme immédiatement la connexion si le client n'utilise pas ce port source (exemple).
lowport	Les connexions TCP et UDP (client) sortantes avec cette option utilisent un port source aléatoire inutilisé entre 640 et 1023 incl. Sur les systèmes d'exploitation de classe UNIX, cela nécessite des privilèges root et indique donc que le processus client est autorisé par la racine locale. Les adresses d'écoute TCP et UDP avec cette option arrêtent immédiatement la connexion si le client n'utilise pas de port source ⇐ 1023. Ce mécanisme peut fournir une autorisation limitée dans certaines circonstances.

Après avoir accepté une connexion, teste si le pair est à portée. Pour les adresses IPv4, la plage d'adresses prend la forme adresse/bits, par exemple 10.0.0.0/8 ou adresse:masque, par exemple 10.0.0.0:255.0.0.0 (exemple); pour IPv6, il s'agit de [ip6-address/bits], par exemple [::1/128]. Si l'adresse du client ne correspond pas, socat émet un avertissement et continue d'écouter / recevoir.

Utilise la bibliothèque libwrap (tcpd) de Wietse Venema pour déterminer si le client est autorisé à se connecter. Les fichiers de configuration sont /etc/hosts.allow et /etc/hosts.deny par défaut, voir "man 5 hosts_access" pour plus d'informations. L'option <nom> (type string) est transmise aux fonctions wrapper en tant que nom de processus démon (exemple). S'il est omis, le nom de base de l'invocation de socats (argv [0]) est transmis. Si les deux options tcpwrap et range sont appliquées à une adresse, les deux conditions doivent être remplies pour autoriser la connexion.

allow-table = <nomfichier>	Prend le fichier spécifié au lieu de /etc/hosts.allow.
deny-table = <nomfichier>	Prend le fichier spécifié au lieu de /etc/hosts.deny.
tcpwrap-etc = <nom du répertoire>	Recherche hosts.allow et hosts.deny dans le répertoire spécifié. Est remplacé par les options hosts-allow et hosts-deny.

Groupe d'options SOCKS

Lorsqu'on utilise des adresses de type SOCKS, certaines options spécifiques aux chaussettes peuvent être définies.

socksport = <service tcp>	Remplace le service par défaut "socks" ou le port 1080 pour le port du serveur socks avec le <service TCP>.
socksuser = <utilisateur>	Envoie le <user> [string] dans le champ du nom d'utilisateur au serveur socks. Le nom d'utilisateur par défaut est $ LOGNAME ou $ USER (exemple).

Groupe d'option d'écoute

Options spécifiques aux prises d'écoute.

backlog = <nombre>	Définit la valeur de backlog transmise avec l'appel système listen () à <count> [int]. La valeur par défaut est 5.
max-children = <nombre>	

Groupe options CHILD

Options pour les adresses avec plusieurs connexions via des processus enfants.

fork	Après avoir établi une connexion,ouvre un processus enfant et fait suivre le processus parent en essayant de créer plus de connexions, soit en écoutant, soit en se connectant dans une boucle (exemple).

OPENSSL-CONNECT et OPENSSL-LISTEN diffèrent quand ils débouchent réellement sur l'enfant:
- OPENSSL-LISTEN fonce avant la prise de contact SSL
- OPENSSL-CONNECT accélère après.

Les options RETRY et FOREVER ne sont pas héritées par le processus enfant.

Sur certains systèmes d'exploitation (par exemple FreeBSD), cette option ne fonctionne pas pour les adresses UDP-LISTEN.

Groupe d'options RANGE

Ces options vérifient si un client connecté doit avoir accès. Ils peuvent être appliqués à l'écoute et à la réception des sockets réseau. Les options tcp-wrappers appartiennent à ce groupe.

Groupe d'options HTTP

Options pouvant être fournies avec des adresses de type HTTP. La seule adresse HTTP actuellement implémentée est la connexion proxy.

proxyport = <service TCP>	Remplace le port proxy HTTP par défaut 8080 par <service TCP>.
ignorecr	Le protocole HTTP nécessite l'utilisation de CR + NL comme terminateur de ligne. Lorsqu'un serveur proxy viole cette norme, socat peut ne pas comprendre sa réponse. Cette option demande à socat d'interpréter NL en tant que terminaison de ligne et d'ignorer le CR dans la réponse. Néanmoins, socat envoie CR + NL au proxy.
proxyauth = <nom d'utilisateur>:<mot de passe>	Fournir une authentification "de base" au serveur proxy. L'argument de l'option est utilisé avec un en-tête "Proxy-Authorization: Base" au format base64. le nom d'utilisateur et le mot de passe sont visibles pour chaque utilisateur de la machine locale dans la liste des processus. Le nom d'utilisateur et le mot de passe sont transférés sur le serveur proxy non cryptés (encodés en base64) et peuvent être détectés.
resolve	Par défaut, socat envoie au proxy une demande CONNECT contenant le nom d'hôte cible. Avec cette option, socat résout le nom d'hôte localement et envoie l'adresse IP. Veuillez noter que, conformément à la RFC 2396, seule la résolution de noms pour les adresses IPv4 est implémentée.

Groupe choix EXEC

Options pour les adresses qui appellent un programme.

path = <chaîne>	Remplacer la variable d'environnement PATH pour effectuer une recherche dans le programme avec <string>. This value $ PATH est également efficace dans le processus enfant.
login	Préfixe argv [0] pour un appel execvp () avec '-', en faisant une sorte de shell se comporte comme un shell de connexion.

Groupe choix FORK

Les adresses EXEC ou SYSTEM invoquent un programme utilisant un processus enfant et transfert de données entre programme et programme. Le mécanisme de communication interprocessus peut être influencé par les options suivantes. Par défaut, un socketpair () is created et à stdin et assigné stdout du enfant Processus, TANDIS Que stderr is du Hérité Processus socat et le Qué enfant utiliser les Processus Descripteurs de 0 et 1 FICHIERS verser Communiquer avec le socat principale Processus.

nofork	Ne pas utiliser un sous-processus pour exécuter le programme, mais appeler execvp \ () ou system \ () directement à partir de l'instance réelle de socat. Cela évite la surcharge d'un autre processus entre le programme et son homologue, mais introduit de nombreuses restrictions: - il ne peut être appliquée Deuxième adresse socat . - il ne peut pas être appliqué à une partie d'une adresse double. - la première adresse socat ne peut pas être OPENSSL ou READLINE - les options socat -b, -t, -D, -l, -v, -x deviennent inutiles - pour les deux adresses, les options ignoreeof, cr et crnl deviennent inutiles - pour la deuxième adresse (celle avec l'option nofork), les options append, cloexec, flock, user, group, mode, non-block, perm-late, setlk et setpgid ne peuvent pas être appliquées. Certains pourraient être utilisés sur la première adresse.
pipes	Crée une paire de canaux sans nom pour la communication interprocessus au lieu d'une paire de sockets.
openpty	Établit la communication avec le sous-processus en utilisant un pseudo-terminal créé avec openpty () au lieu de celui par défaut (socketpair ou ptmx).
ptmx	Établit la communication avec le sous-processus en utilisant un pseudo-terminal créé en ouvrant / dev / ptmx ou / dev / ptc au lieu du défaut (socketpair).
pty	Établit la communication avec le sous-processus en utilisant un pseudo-terminal au lieu d'une paire de sockets. Crée le pty avec un mécanisme disponible. Si openpty et ptmx sont tous deux disponibles, il utilise ptmx car il est compatible POSIX (exemple).
ctty	Rend le pty le contrôle du sous-processus (exemple).
stderr	Dirige stderr du sous processus vers son canal de sortie en faisant de stderr un dup () de stdout (exemple).
fdin = <fdnum>	Assigne le canal d'entrée des sous-processus à son descripteur de fichier <fdnum> au lieu de stdin (0). Le programme lancé à partir du sous-processus doit utiliser cette fd pour lire les données de socat (exemple).
fdout = <fdnum>	Assigne le canal de sortie des sous-processus à son descripteur de fichier <fdnum> au lieu de stdout (1). Le programme lancé à partir du sous-processus doit utiliser cette fd pour écrire des données sur socat (exemple).
soupir , sigint , sigquit	A socat signaux de passage de ce type au sous - processus. Si aucune adresse n'a cette option, socat se termine sur ces signaux.

Groupe d'options TERMIOS

Pour les adresses qui fonctionnent sur un tty (par exemple, stdio, fichier: / dev / tty, exec: ..., pty), les paramètres de terminal définis dans le mécanisme UN * X option d'adresse.

les modifications des paramètres du terminal interactif sont toujours effectives après la fin du processus , que l'on peut nettoyer en utilisant “reset” ou “stty sane” dans le shell. Pour les adresses EXEC et SYSTEM avec option PTY, ces options apposent aux processus pty par enfant.

b0	Déconnecte le terminal.
b19200	Règle la vitesse de la ligne série 19 200 bauds. Certains autres taux sont possibles; Utiliser quelque chose comme `socat -hh | grep 'b[1-9]'` pour trouver toutes les vitesses supportées par votre implémentation. sur certains systèmes d'exploitation, ces options peuvent ne pas être disponibles. Utiliser ispeed ou ospeed à la place.
echo = <bool>	Active ou désactive l'écho local.
icanon = <bool>	Définit ou efface le mode canonique, permettant la mise en mémoire tampon de lignes et certains caractères spéciaux.
raw	Définit le mode brut, transmettant ainsi les entrées et les sorties presque sans traitement. Cette option est obsolète, utiliser plutôt l'option rawer ou cfmakeraw.
rawer	Rend le terminal plus brut que l'option brute. Cette option désactive implicitement l'écho. (Exemple).
cfmakeraw	Définit le mode brut en appelant cfmakeraw () ou en simulant cet appel. Cette option désactive implicitement l'écho.
ignbrk = <bool>	Ignore ou interprète le caractère BREAK (par exemple, ^C)
brkint = <bool>	
bs<0/1>	
bsdly = <0/1>	
clocal = <bool>	
cr<0/1/2/3>	Définit le délai de retour chariot à 0, 1, 2 ou 3, respectivement. 0 signifie pas de délai, les autres valeurs dépendent du terminal.
crdly = <0/1/2/3>	
cread = <bool>	
crtscts = <bool>	
cs<5/6/7/8>	Définit la taille des caractères à 5, 6, 7 ou 8 bits respectivement.
csize = <0/1/2/3>	
cstopb = <bool>	Définit deux bits d'arrêt plutôt qu'un.
dsusp = <octet>	Définit la valeur du caractère VDSUSP qui suspend le processus de premier plan en cours et réactive le shell (tous sauf Linux).
echoctl = <bool>	Les échos contrôlent les caractères en notation de chapeau (par ex.
echoe = <bool>	
echok = <bool>	
echoke = <bool>	
echonl = <bool>	
echoprt = <bool>	
eof = <octet>	
eol = <octet>	
eol2 = <octet>	
effacer = <octet>	
jeter = <octet>	
ff0	
ff1	
ffdly = <bool>	
flusho = <bool>	

hupcl = <bool>	
icrnl = <bool>	
iexten = <bool>	
igncr = <bool>	
ignpar = <bool>	
imaxbel = <bool>	
inlcr = <bool>	
inpck = <bool>	
intr = <octet>	
isig = <bool>	
ispeed = <unsigned-int>	Définit le débit en bauds pour les données entrantes sur cette ligne. Voir aussi: ospeed, b19200
istrip = <bool>	
iuclc = <bool>	
ixany = <bool>	
ixoff = <bool>	
ixon = <bool>	
kill = <octet>	
lnext = <octet>	
min = <octet>	
nl0	Définit le délai de la nouvelle ligne sur 0.
nl1	
nldly = <bool>	
noflsh = <bool>	
ocrnl = <bool>	
ofdel = <bool>	
ofill = <bool>	
olcuc = <bool>	
onlcr = <bool>	
onlret = <bool>	
onocr = <bool>	
opost = <bool>	Active ou désactive le traitement de sortie; par exemple, convertit NL en CR-NL.
ospeed = <unsigned-int>	Définit le débit en bauds pour les données sortantes sur cette ligne. Voir aussi: ispeed, b19200
parenb = <bool>	Activer la génération de parité à la sortie et au contrôle de parité pour les entrées.
parmrk = <bool>	
parodd = <bool>	
pendin = <bool>	
quit = <octet>	
reprint = <octet>	
sane	Amène le terminal a choisi comme un état par défaut utile.
start = <octet>	
stop = <octet>	

susp = <octet>	
swtc = <octet>	
tab<0/1/2/3>	
tabdly = <unsigned-int>	
time = <octet>	
tostop = <bool>	
vt<0/1>	
vtdly = <bool>	
werase = <octet>	
xcase = <bool>	
xtabs	
i-pop-all	Avec UNIX System V STREAMS, prend en charge tous les pilotes de la pile.
i-push = <chaîne>	Avec UNIX System V STREAMS, pousse le pilote (module) avec le nom donné (chaîne) sur la pile. Par exemple, pour assurer un périphérique de caractères sur Solaris supporte termios, etc., utiliser les options suivantes: i-pop-all, i-push = ptem, i-push = ldterm, i-push = ttcompat

Groupe de choix PTY

Ces options sont les suivantes:

link = <nomfichier>	Génère un lien symbolique qui pointe vers le pseudo-terminal (pty). Cela pourrait aider à résoudre le problème que les ptys sont générés avec des noms plus ou moins imprévisibles, rendant difficile l'accès direct au pty généré automatiquement. Avec cette option, l'utilisateur peut spécifier un point “fix” dans la hiérarchie des fichiers, ce qui l'aide à accéder au pty actuel (exemple). En commençant par socat la version 1.4.3, le lien symbolique est supprimé lorsque l'adresse est fermée (mais voir option unlink-fermeture).
wait-slave	Bloque la phase ouverte jusqu'à ce qu'un processus ouvre le côté esclave du pty. Habituellement, socat continue après avoir eu le pty en ouvrant la suite ou en entrant dans la boucle de transfert. Avec la fonction wait-slave, socat assister à ce que le processus ouvre le côté esclave du pty avant de continuer. Cette option ne permet pas le système d'exploitation fourni d'appel système poll (). Et Cela dépend d'un comportement non documenté, de sorte qu'il ne fonctionne pas sur tous les systèmes d'exploitation. Il a été testé avec Linux, FreeBSD, NetBSD et Tru64 avec openpty.
pty-interval = <secondes>	Lorsque l'option wait-slave est définie, socat vérifie périodiquement la condition HUP en utilisant poll () pour trouver si le côté esclave du pty a été ouvert. L'intervalle d'interrogation par défaut est 1s. Utiliser l'option pty-interval [timeval] pour modifier cette valeur.

Groupe d'options OPENSSL

Ces options s'appliquent aux types aux adresses openssl et openssl-listen.

cipher = <liste de chiffrement>	Sélectionne la liste des articles pouvant être utilisés pour la connexion. -Se Reporter la de la page par rapport aux chiffrements manuel, section FORMAT DE LA LISTE DES CIPHEROS , verser obtain des informations sur la syntax détaillées, les facts et la Valeur par défaut de <Liste de chiffrement>. Plusieurs chaînes de chiffrement peuvent être données, séparées par «:». Quelques chaînes de chiffrement simples: - **3DES** : suite de chiffrement avec triple DES. - **MD5**: suite de chiffrement avec MD5. - **aNULL**: suite de chiffrement sans authentification. - **NULL**: N'utilise pas le chiffrement. - **HIGH**: Utiliser une suite de chiffrement avec un chiffrement “élevé”. l'homologue doit prendre en charge la propriété sélectionnée ou que la négociation échoue.
method = <méthode ssl>	Définir la version du protocole à utiliser. Les chaînes valides (non sensibles à la casse) sont les suivantes: - **SSL2**: protocole SSL version 2. - **SSL3**: protocole SSL version 3. - **SSL23**: meilleur protocole SSL ou TLS disponible. C'est la valeur par défaut lorsque cette option n'est pas fournie. - **TLS1**: protocole TLS version 1 - **TLS1.1**: protocole TLS version 1.1 - **TLS1.2**: protocole TLS version 1.2 - **DTLS1**: protocole DTLS version 1.
verify = <bool>	Contrôle la vérification du certificat de l'homologue. La valeur par défaut est 1 (true). Désactiver la vérification peut ouvrir votre socket pour tout le monde, rendant le cryptage inutile!
cert = <nomfichier>	
key = <nomfichier>	Spécifie le fichier avec la clé privée. La clé privée peut se trouver dans ce fichier ou dans le fichier fourni avec l'option cert. La partie qui doit prouver qu'elle est le propriétaire d'un certificat a besoin de la clé privée.
dhparams = <nomfichier>	Spécifie le fichier avec les paramètres Diffie Hellman. Ces paramètres peuvent également se trouver dans le fichier fourni avec l'option cert, auquel cas l'option dhparams n'est pas nécessaire.
cafile = <nomfichier>	Spécifie le fichier avec les certificats d'autorité approuvés (root). Le fichier doit être au format PEM et doit contenir un ou plusieurs certificats. La partie qui vérifie l'authentification de son homologue approuve uniquement les certificats contenus dans ce fichier.
capath = <dirname>	Spécifie le répertoire avec les certificats (racine) approuvés. Le répertoire doit contenir des certificats au format PEM et leurs hashs (voir la documentation OpenSSL)
egd = <nomfichier>	Sur certains systèmes, openssl nécessite une source explicite de données aléatoires. Spécifier le nom du socket où un démon de collecte d'entropie comme egd fournit des données aléatoires, par exemple / dev / egd-pool.
pseudo	Sur les systèmes où openssl ne peut pas trouver de source d'entropie et où aucun démon de collecte d'entropie ne peut être utilisé, cette option active un mécanisme de fourniture de pseudo-entropie. Ceci est réalisé en prenant le temps actuel en microsecondes pour alimenter le générateur de nombres pseudo-aléatoires libc avec une valeur initiale. openssl est alors alimenté avec la sortie des appels aléatoires \ (). Ce mécanisme n'est pas suffisant pour générer des clés sécurisées!

compress	Activer ou désactiver l'utilisation de la compression pour une connexion. Si on choisi “none”, la compression est désactivée, et le paramètre “auto” permet à OpenSSL de choisir le meilleur algorithme disponible pris en charge par les deux parties. La valeur par défaut est de ne pas toucher aux paramètres liés à la compression. nécessite OpenSSL 0.9.8 ou supérieur et la désactivation de la compression avec OpenSSL 0.9.8 affecte toutes les nouvelles connexions du processus.
commonname = <chaîne>	Indique le nom de famille que le certificat homologue doit correspondre. Avec l'adresse **OPENSSL-CONNECT**, cela remplace le nom d'hôte ou l'adresse IP donné. Avec **OPENSSL-LISTEN**, cela active la vérification du nom des certificats homologues. Cette option n'a de sens que lorsque l'option verify n'est pas désactivée et que le chiffrement choisi fournit un certificat homologue.
fips	Mode actif FIPS s'il est compilé. Pour plus d'informations sur la norme d'implantation du cryptage FIPS, voir http://oss-institute.org/fips-faq.html. Ce mode peut nécessiter que les certificats soient générés avec une version FIPS de openssl. Définir ou effacer cette option sur une adresse pour toutes les adresses OpenSSL de ce processus.

Groupe d'option RETRY

Options permettant de réessayer certains systèmes, notamment les tentatives de connexion.

retry = <num>	Nombre de tentatives avant que la connexion ou la tentative d'écoute soit interrompue. La valeur par défaut est 0, ce qui signifie juste une tentative.
interval = <timepec>	Temps entre deux tentatives consécutives (secondes, [timespec]). La valeur par défaut est 1 seconde.
forever	Effectue un nombre illimité de nouvelles tentatives.

Groupe d'options TUN

Options qui contrôlent les adresses des périphériques d'interface TUN / TAP Linux.

tun-device = <fichier-périphérique>	Indique à socat pour prendre un autre chemin pour le clone TUN. La valeur par défaut est /dev /net /tun.
tun-name = <if-name>	Donne à un réseau résultant d'une dénomination spécifique au lieu du système (tun0, tun1, etc.)
tun-type = [tun/tap]	Définit le type du périphérique TUN; Utiliser cette option pour générer un périphérique TAP. Voir le document Linux pour la différence entre ces types. Lorsqu'on essaye de créer un tunnel entre deux systèmes, leurs types doivent être identiques.
iff-no-pi	Définit l'indicateur IFF_NO_PI qui contrôle le périphérique avec des informations de paquets supplémentaires dans le tunnel. Lorsqu'on essaye de mettre en place un tunnel entre deux périphériques TUN, ces indicateurs doivent avoir les mêmes valeurs.
iff-up	Définir le statut du réseau d'interface TUN UP. Fortement recommandé.
iff-broadcast	Définit l'indicateur BROADCAST de l'interface réseau TUN.
iff-debug	Définir l'indicateur DEBUG de l'interface réseau TUN.
iff-loopback	Définit l'indicateur LOOPBACK de l'interface réseau TUN.
iff-pointopoint	Définit l'indicateur POINTOPOINT du périphérique TUN.
iff-notrailers	Définit l'indicateur NOTRAILERS du périphérique TUN.

iff-running	Définit l'indicateur RUNNING du périphérique TUN.
iff-noarp	Définit l'indicateur NOARP du périphérique TUN.
iff-promisc	Définit l'indicateur PROMISC du périphérique TUN.
iff-allmulti	Définit l'indicateur ALLMULTI du périphérique TUN.
iff-master	Définit l'indicateur MASTER du périphérique TUN.
iff-esclave	Définit l'indicateur SLAVE du périphérique TUN.
siff-multicast	Définit le drapeau MULTICAST du périphérique TUN.
iff-portsel	Définit l'indicateur PORTSEL du périphérique TUN.
iff-automedia	Définit l'indicateur AUTOMEDIA du périphérique TUN.
iff-dynamique	Définit l'indicateur DYNAMIC du périphérique TUN.

VALEURS DONNÉES

Cette section explique les différents types de données que peuvent prendre les paramètres d'adresse et les options d'adresse.

plage d'adresses	Est actuellement uniquement implémenté pour IPv4 et IPv6. Voir adresse-option range
bool	“0” ou “1”; si la valeur est omise, “1” est pris.
octet	Un nombre int non signé, lu avec strtoul (), inférieur ou égal à UCHAR_MAX.
ligne de commande	Une chaîne spécifiant un nom de programme et ses arguments, séparés par des espaces simples.
Les données	Une spécification de données brutes suivant la syntaxe Dalan . Actuellement, la seule forme valide est une chaîne commençant par «x» suivie d'un nombre pair de chiffres hexadécimaux, spécifiant une séquence d'octets.
annuaire	Une chaîne avec la sémantique habituelle du nom de répertoire UN * X.
établissement	Nom d'une fonction syslog en minuscules.
fdnum	Un type int non signé, lu avec strtoul (), en spécifiant un descripteur de fichier UN * X.
nom de fichier	Une chaîne avec la sémantique habituelle du nom de fichier UN * X.
groupe	Si le premier caractère est un chiffre décimal, la valeur est lue avec strtoul () comme entier non signé spécifiant un identifiant de groupe. Sinon, il doit s'agir d'un nom de groupe existant.
int	Un nombre suivant les règles de la fonction strtol () avec la base “0”, c'est-à-dire un nombre décimal, un nombre octal avec un “0” et un nombre hexadécimal avec “0x”. La valeur doit correspondre à un C int.
interface	Une chaîne spécifiant le nom de périphérique d'une interface réseau, comme indiqué par ifconfig ou procan, par exemple “eth0”.
adresse IP	Une adresse IPv4 en notation numérique et en points, une adresse IPv6 en notation hexadécimale entre crochets ou un nom d'hôte résolvant une adresse IPv4 ou IPv6.

Exemples: 127.0.0.1, [:: 1], www.dest-unreach.org, dns1 |

Adresse IPv4	Une adresse IPv4 en notation numérique et à points ou un nom d'hôte résolvant une adresse IPv4. Exemples: 127.0.0.1, www.dest-unreach.org, dns2

Adresse IPv6	Une adresse iPv6 en notation hexnumbers-and-colons entre crochets ou un nom d'hôte résolvant une adresse IPv6. Exemples: [:: 1], [1234: 5678: 9abc: def0: 1234: 5678: 9abc: def0], ip6name.domain.org
long	Un nombre lu avec strtol(). La valeur doit correspondre à un C long.
long long	Un nombre lu avec strtoll(). La valeur doit correspondre à un long C long.
off_t	Un nombre de signatures dépendant de l'implémentation, généralement 32 bits, lu avec strtol ou strtoll.
off64_t	Un nombre de signatures dépendant de l'implémentation, généralement 64 bits, lu avec strtol ou strtoll.
mode_t	Un entier non signé, lu avec strtoul(), spécifiant les bits de mode (permission).
pid_t	Un nombre, lu avec strtol (), spécifiant un identifiant de processus.
Port	Un uint16_t (numéro non signé 16 bits) spécifiant un port TCP ou UDP, lu avec strtoul().
protocole	Un numéro non signé de 8 bits, lu avec strtoul().
size_t	Un nombre non signé avec des limitations de taille_t, lu avec strtoul.
sockname	Une adresse de socket Voir adresse-option `bind`
chaîne	Une séquence de caractères ne contenant pas “\ 0” et, selon la position dans la ligne de commande, “:”, “,” ou “!!”. Il faudra peut-être échapper les méta-caractères du shell dans la ligne de commande.
Service TCP	Un nom de service, ne commençant pas par un chiffre, résolu par getservbyname(), ou un numéro de 16 bits non signé int lu avec strtoul().
timeval	Un double flotteur spécifiant les secondes; le nombre est mappé dans une structure timeval, constituée de secondes et de microsecondes.
timespec	Un double flotteur spécifiant les secondes; le nombre est mappé dans un struct timespec, constitué de secondes et de nanosecondes.
Service UDP	Un nom de service, ne commençant pas par un chiffre, résolu par getservbyname(), ou un numéro de 16 bits non signé int lu avec strtoul().
unsigned int	Un numéro lu avec strtoul(). La valeur doit tenir dans un C unsigned int.
user	Si le premier caractère est un chiffre décimal, la valeur est lue avec strtoul () comme un entier non signé spécifiant un identifiant d'utilisateur. Sinon, il doit s'agir d'un nom d'utilisateur existant.

Simuler des requêtes HTTP

On peut saisir du texte avec la bibliothèque readline et il sera envoyé en TCP à www.domain.org sur le port 80 (www).

```
socat -d -d READLINE,history=$HOME/.http\_history
TCP4:www.domain.org:www,crnl
```

Simple transfert de données entre 2 flux TCP.

Tout ce qui arrive sur le port 80 (www) de la machine locale est envoyé vers www.domain.org et inversement.

```
socat TCP4-LISTEN:www TCP4:www.domain.org:www
```

Ecrire tout le flux reçu sur le port 3334 dans un fichier.

```
socat -u TCP4-LISTEN:3334,reuseaddr,fork OPEN:/tmp/in.log,creat,append
```

Simuler une connexion sur un équipement

```
(sleep 5; echo yes; sleep 5; echo $PASSWORD; sleep 2; echo ; echo "screen-
length disable"; sleep 2; echo "dis cur") | socat - EXEC:"ssh
$USERNAME@xx.xxx.xxx.xxx2.33",setsid,pty,ctty
```

Transfert des données entre STDIO (-) et une connexion TCP4

```
socat - TCP4:www.domain.org:80
```

transfère des données entre STDIO (-) et une connexion TCP4 au port 80 de l'hôte www.domain.org. Cet exemple se traduit par une connexion interactive similaire à telnet ou netcat. Les paramètres du terminal stdin ne sont pas modifiés, on peut donc fermer le relais avec ^ D ou abandonner avec ^ C.

```
socat -d -d READLINE,history=$HOME/.http\_history
TCP4:www.domain.org:www,crnl
```

Ceci est similaire à l'exemple précédent, mais on peut éditer la ligne en cours de manière bash (READLINE) et utiliser le fichier d'historique .http_history; socat imprime des messages sur la progression (-d -d). Le port est spécifié par nom de service (www) et les caractères de terminaison de ligne réseau (crnl) corrects au lieu de NL sont utilisés.

Simple redirecteur de port TCP

```
socat TCP4-LISTEN:www TCP4:www.domain.org:www
```

Installe un simple redirecteur de port TCP. Avec TCP4-LISTEN, il écoute le port local “www” jusqu'à ce qu'une connexion arrive, l'accepte, puis se connecte à l'hôte distant (TCP4) et commence le transfert de données. Il n'acceptera pas une deuxième connexion.

TCP port forwarder

```
socat -d -d -lmlocal2 TCP4-
LISTEN:80,bind=myaddr1,reuseaddr,fork,su=nobody,range=10.0.0.0/8
TCP4:www.domain.org:80,bind=myaddr2
```

TCP port forwarder, chaque côté lié à une autre adresse IP locale (bind). Cet exemple gère un nombre presque arbitraire de connexions parallèles ou consécutives en générant un nouveau processus après chaque accept (). Il offre un peu de sécurité en faisant appel à l'utilisateur après avoir fui; il ne permet que les connexions du réseau privé 10 (gamme); à cause de reuseaddr, cela permet un redémarrage

immédiat après la fin du processus maître, même si certains sockets enfants ne sont pas complètement arrêtés. Avec -lmlocal2, socat se connecte à stderr jusqu'à ce que la boucle d'acceptation soit atteinte. La journalisation supplémentaire est dirigée vers syslog avec facilité local2.

Serveur simple qui accepte les connexions

```
socat TCP4-LISTEN:5555,fork,tcpwrap=script
EXEC:/bin/myscript,chroot=/home/sandbox,su-d=sandbox,pty,stderr
```

un serveur simple qui accepte les connexions (TCP4-LISTEN) et fork's un nouveau processus enfant pour chaque connexion; chaque enfant agit comme un seul relais. Le client doit correspondre aux règles pour le nom du processus démon “script” dans /etc/hosts.allow et /etc/hosts.deny, sinon l'accès lui est refusé (voir “man 5 hosts_access”). Pour exécuter le programme, le processus enfant chroote vers / home / sandbox , su vers user sandbox, puis démarre le programme / home / sandbox / bin / myscript . Socat et myscript communiquent via un pseudo tty (pty); Le stderr de myscript est redirigé vers stdout, donc ses messages d'erreur sont transférés via socat au client connecté.

relais SMTP

```
socat EXEC:"mail.sh target@domain.com",fdin=3,fdout=4
TCP4:mail.relay.org:25,crnl,bind=alias1.server.org,mss=512
```

mail.sh est un script shell, distribué avec socat , qui implémente un simple client SMTP. Il est programmé pour “parler” SMTP sur ses FD 3 (in) et 4 (out). Les options fdin et fdout indiquent à socat d'utiliser ces FD pour communiquer avec le programme. Comme mail.sh hérite de stdin et de stdout alors que socat ne les utilise pas, le script peut lire un corps de courrier à partir de stdin. Socat fait de alias1 votre adresse source locale (bind), prend en charge la terminaison de ligne réseau correcte (crnl) et envoie au plus 512 octets de données par paquet (mss).

connexion port série

```
socat -,escape=0x0f /dev/ttyS0,rawer,crnl
```

ouvre une connexion interactive via la ligne série, par exemple pour dialoguer avec un modem. rawer définit les paramètres de terminal de la console et de ttyS0 sur des valeurs praticables, crnl les convertit pour corriger les caractères de nouvelle ligne. escape permet de terminer le processus socat avec le contrôle de caractère-O.

Interception des connexions XWindow

```
socat UNIX-LISTEN:/tmp/.X11-unix/X1,fork
SOCKS4:host.victim.org:127.0.0.1:6000,socksuser=nobody,sourceport=20
```

avec UNIX- LISTEN , socat ouvre une socket de domaine UNIX d'écoute /tmp/.X11-unix/X1 . Ce chemin

correspond à l'affichage local de XWindow: 1 sur votre machine, les connexions client XWindow à DISPLAY =: 1 sont donc acceptées. Socat parle ensuite avec le serveur SOCKS4 host.victim.org qui pourrait autoriser des connexions basées sur le port source en raison d'une faiblesse liée à FTP dans ses filtres IP statiques. Socat prétend être appelé par socksuser nobody, et demande à être connecté au port de bouclage 6000 (seules les configurations sockd faibles le permettent). Donc, nous obtenons une connexion au serveur XWindow des victimes et, si cela ne nécessite pas de cookies MIT ou d'authentification Kerberos, nous pouvons commencer à travailler. Il ne peut y avoir qu'une seule connexion à la fois, car TCP ne peut établir qu'une seule session avec un ensemble donné d'adresses et de ports.

transfert de données unidirectionnel

```
socat -u /tmp/readdata,seek-end=0,ignoreeof -
```

c'est un exemple de transfert de données unidirectionnel (-u). Socat transfère les données du fichier / tmp / readdata (adresse implicite de GOPEN), en commençant à la fin actuelle (seek-end = 0 permet à socat de commencer la lecture à la fin du fichier en cours;) en mode “tail -f” (ignoreeof). Le “fichier” peut également être une socket de domaine UNIX en écoute (n'utiliser pas d’option de recherche).

Connexion ssh automatique

```
 (sleep 5; echo PASSWORD; sleep 5; echo ls; sleep 1) | socat - EXEC:'ssh -l
user server',pty,setsid,ctty
```

Exécute une session ssh sur un serveur. Utilise un pty pour la communication entre socat et ssh, rend ssh contrôlant tty (ctty), et en fait le propriétaire d'un nouveau groupe de processus (setsid), donc ssh accepte le mot de passe de socat.

Collecteur de messages simple

```
socat -u TCP4-LISTEN:3334,reuseaddr,fork OPEN:/tmp/in.log,creat,append
```

implémente un collecteur de messages simple basé sur le réseau. Pour chaque client se connectant au port 3334, un nouveau processus enfant est généré (option fork). Toutes les données envoyées par les clients sont ajoutées au fichier /tmp/in.log. Si le fichier n'existe pas, c'est socat creat. L'option reuseaddr permet le redémarrage immédiat du processus du serveur.

pseudo modem série/ssh

```
socat PTY,link=$HOME/dev/vmodem0,rawer,wait-slave EXEC:"ssh
modemserver.us.org socat - /dev/ttyS0,nonblock,rawer"
```

génère un pseudo terminal (PTY) sur le client accessible sous le lien symbolique $HOME/dev/vmodem0 . Une application qui attend une ligne série ou un modem peut être configurée

pour utiliser $HOME/dev/vmodem0 ; son trafic sera dirigé vers un modemserver via ssh où une autre instance de socat le lie à /dev/ttyS0 .

proxy relay

```
socat TCP4-LISTEN:2022,reuseaddr,fork
PROXY:proxy:www.domain.org:22,proxyport=3128,proxyauth=user:pass
```

démarre un redirecteur qui accepte les connexions sur le port 2022 et les dirige via le démon proxy écoutant sur le port 3128 (proxyport) sur le proxy hôte, en utilisant la méthode CONNECT, où ils sont authentifiés en tant que “utilisateur” avec “pass” (proxyauth). Le proxy doit établir des connexions pour héberger www.domain.org sur le port 22 à ce moment-là.

Connexion sécurisée SSL

```
socat - OPENSSL:server:4443,cafile=server.crt,cert=client.pem
```

est un client OpenSSL qui tente d'établir une connexion sécurisée à un serveur SSL. Option cafile spécifie un fichier qui contient des certificats de confiance: nous faisons confiance au serveur uniquement lorsqu'il présente l'un de ces certificats et preuves qu'il possède la clé privée associée. Sinon, la connexion est terminée. Avec cert, un fichier contenant le certificat client et la clé privée associée est spécifié. Cela est nécessaire si le serveur souhaite une authentification client; beaucoup de serveurs Internet ne le font pas.

La première adresse ('-') peut être remplacée par presque toute autre adresse socat.

Serveur OPENSSL

```
socat OPENSSL-
LISTEN:4443,reuseaddr,pf=ip4,fork,cert=server.pem,cafile=client.crt PIPE
```

est un serveur OpenSSL qui accepte les connexions TCP, présente le certificat à partir du fichier server.pem et force le client à présenter un certificat vérifié par rapport à cafile.crt.

La deuxième adresse («PIPE») peut être remplacée par presque toutes les autres adresses socat.

Pour obtenir des instructions sur la génération et la distribution des clés et des certificats OpenSSL, consulter le fichier socat docu socat-openssl.txt.

création d'un fichier sparse de 100 Go

```
echo |socat -u - file:/tmp/bigfile,create,largefile,seek=100000000000
```

crée un fichier sparse de 100 Go; Cela nécessite un type de système de fichiers qui le supporte (ext2, ext3, reiserfs, jfs, pas minix, vfat). L'opération d'écriture de 1 octet peut être longue (reiserfs: quelques minutes; ext2: “non”), et le fichier résultant peut consommer de l'espace disque

uniquement avec ses inodes (reiserfs: 2 Mo; ext2: 16 Ko).

Impression des connexions entrantes

```
socat tcp-l:7777,reuseaddr,fork system:’filan -i 0 -s >&2’,nofork
```

écoute les connexions TCP entrantes sur le port 7777. Pour chaque connexion acceptée, appelle un shell. Ce shell a ses stdin et stdout directement connectés au socket TCP (nofork). Le shell démarre filan et lui permet d'imprimer les adresses de socket à stderr (votre fenêtre de terminal).

éditeur binaire primitif

```
echo -en "\0\14\0\0\c" |socat -u - file:/usr/bin/squid.exe,seek=0x00074420
```

fonctionne comme un éditeur binaire primitif: il écrit les 4 octets 000 014 000 000 dans l'exécutable / usr / bin / squid à l'offset 0x00074420 (c'est un patch du monde réel pour exécuter l'exécutable squid de Cygwin sous Windows, réel par mai 2004) .

Filtre internet

```
socat - tcp:www.blackhat.org:31337,readbytes=1000
```

Se connecte à un service inconnu et empêche d'être inondé.

Fusion de flux TCP

```
socat -U TCP:target:9999,end-close TCP-L:8888,reuseaddr,fork
```

fusionne les données provenant de différents flux TCP sur le port 8888 avec un seul flux à cibler: 9999. L'option end-close empêche les processus fils séparés par la deuxième adresse de terminer la connexion partagée à 9999 (close \ (2) dissocie simplement l'inode qui reste actif tant que le processus parent est actif; shutdown \ (2) active terminer la connexion).

Interrogation serveurs de temps

```
socat - UDP4-DATAGRAM:192.168.1.0:123,sp=123,broadcast,range=192.168.1.0/24
```

envoie une diffusion au réseau 192.168.1.0/24 et reçoit les réponses des serveurs de temps. Ignore les paquets NTP des hôtes situés en dehors de ce réseau.

```
socat - SOCKET-
DATAGRAM:2:2:17:x007bxc0a80100x0000000000000000,bind=x007bx00000000x00000000
00000000,setsockopt-
int=1:6:1,range=x0000xc0a80100x0000000000000000:x0000xffffff00x0000000000000
```

```
000
```

est sémantiquement équivalent à l'exemple précédent, mais tous les paramètres sont spécifiés sous forme générique. la valeur 6 de setsockopt-int correspond à la valeur Linux pour SO_BROADCAST.

Broadcast

```
socat - IP4-DATAGRAM:255.255.255.255:44,broadcast,range=10.0.0.0/8
```

envoie une diffusion au(x) réseau(x) local(aux) à l'aide du protocole 44. Accepte les réponses de la plage d'adresses privées uniquement.

```
socat - UDP4-DATAGRAM:224.255.0.1:6666,bind=:6666,ip-add-
membership=224.255.0.1:eth0
```

transfère des données de stdin vers l'adresse de multidiffusion spécifiée en utilisant UDP. Les ports locaux et distants sont tous deux 6666. Indique à l'interface eth0 d'accepter également les paquets de multidiffusion du groupe donné. Plusieurs hôtes sur le réseau local peuvent exécuter cette commande, de sorte que toutes les données envoyées par l'un des hôtes seront reçues par toutes les autres. Il existe de nombreuses raisons possibles d'échec, notamment les filtres IP, les problèmes de routage, la mauvaise sélection de l'interface par le système d'exploitation, les ponts ou un commutateur mal configuré.

Connexion à un interface virtuel

```
socat TCP:host2:4443 TUN:192.168.255.1/24,up
```

établit un côté d'un réseau virtuel (mais pas privé!) avec host2 où un processus similaire peut s'exécuter, avec UDP-L et tun address 192.168.255.2. Ils peuvent se joindre à l'aide des adresses 192.168.255.1 et 192.168.255.2. Le streaming par exemple. via TCP ou SSL ne garantit pas de conserver les limites des paquets et peut donc provoquer une perte de paquets.

liaison Point à point (PPPD)

```
socat PTY,link=/var/run/ppp,rawer INTERFACE:hdlc0
```

contourner le problème que pppd nécessite un périphérique série et peut donc ne pas être en mesure de travailler sur une ligne synchrone représentée par un périphérique réseau. socat crée un PTY pour rendre pppd heureux, se lie à l'interface réseau hdlc0 et peut transférer des données entre les deux appareils. Utiliser ensuite pppd sur le périphérique /var/run/ ppp.

serveur HTTP echo simple

```
socat -T 1 -d -d TCP-L:10081,reuseaddr,fork,crlf SYSTEM:"echo -e
\"\\\"HTTP/1.0 200 OK\\\nDocumentType: text/plain\\\n\\\ndate:
```

```
\$\(date\)\\\nserver:\$SOCAT\_SOCKADDR:\$SOCAT\_SOCKPORT\\\nclient:
\$SOCAT\_PEERADDR:\$SOCAT\_PEERPORT\\\n\\\"\"; cat; echo -e
\"\\\"\\\n\\\"\""
```

crée un serveur HTTP echo simple: chaque client HTTP qui se connecte obtient une réponse HTTP valide contenant des informations sur l'adresse et le port du client tels qu'ils sont vus par l'hôte du serveur, l'adresse de l'hôte (pouvant varier sur les serveurs multi-hôtes) demande du client.

impression des variables d'environnement

```
  socat -d -d UDP4-RECVFROM:9999,so-broadcast,so-timestamp,ip-pktinfo,ip-
recverr,ip-recvopts,ip-recvtos,ip-recvttl!!- SYSTEM:'export; sleep 1' |grep
SOCAT
```

attend un paquet UDP entrant sur le port 9999 et imprime les variables d'environnement fournies par socat. Sur les systèmes basés sur BSD, il faudra remplacer ip-pktinfo par ip-recvdstaddr, ip-recvif. SOCAT_IP_DSTADDR est particulièrement intéressant: il contient l'adresse cible du paquet, qui peut être une adresse unicast, multicast ou broadcast.

DIAGNOSTIC

Socat utilise un mécanisme de journalisation qui permet de filtrer les messages par gravité. Les sévérités fournies sont plus ou moins compatibles avec la priorité syslog appropriée. Avec une ou quatre occurrences de l'option de ligne de commande -d, la priorité la plus basse des messages émis peut être sélectionnée. Chaque message contient un caractère majuscule unique spécifiant la gravité des messages (l'un des caractères F, E, W, N, I ou D).

Gravité	Description
FATAL	Conditions nécessitant la fin du programme sans condition et immédiate.
ERREUR	Conditions qui empêchent un traitement correct du programme. Habituellement, le programme est terminé (voir option -s).
WARNING	Quelque chose n'a pas fonctionné correctement ou est dans un état où un traitement ultérieur correct ne peut pas être garanti, mais pourrait être possible.
NOTE	Actions intéressantes du programme, par exemple pour superviser socat dans une sorte de mode serveur.
INFO	Description de ce que fait le programme, et peut-être pourquoi cela se produit. Permet de surveiller les cycles de vie des descripteurs de fichiers.
DEBUG	Description du fonctionnement du programme, de tous les appels du système ou de la bibliothèque et de leurs résultats.

Les messages de journal peuvent être écrits dans stderr, dans un fichier ou dans syslog.

En sortie, socat donne le statut 0 s'il est terminé en raison d'un dépassement EOF ou d'un délai d'inactivité, avec une valeur positive en cas d'erreur et une valeur négative en cas d'erreur fatale.

VARIABLES D'ENVIRONNEMENT

Les variables d'entrée transportent des informations de l'environnement vers socat, les variables de sortie sont définies par socat pour être utilisées dans des scripts et des programmes exécutés.

Dans les variables de sortie commençant par “SOCAT”, ce préfixe est en fait remplacé par le nom en majuscule de l'exécutable ou la valeur de l'option -lp.

Variable	**Description**
SOCAT_DEFAULT_LISTEN_IP (entrée)	(Valeurs 4 ou 6) Définit la version IP à utiliser pour les adresses listen, recv et recvfrom si aucune option pf (protocole-famille) n'est indiquée. Est remplacé par les options socat -4 ou -6.
SOCAT_PREFERRED_RESOLVE_IP (entrée)	(Valeurs 0, 4 ou 6) Définit la version IP à utiliser lors de la résolution des noms d'hôte cible lorsque la version n'est pas spécifiée par type d'adresse, option pf (famille de protocole) ou format d'adresse. Si la résolution de nom ne renvoie pas d'entrée correspondante, le premier résultat (avec une version IP différente) est pris. Avec la valeur 0, socat sélectionne toujours le premier enregistrement et sa version IP.
SOCAT_FORK_WAIT (entrée)	Spécifie la durée (en secondes) pendant laquelle les processus parent et enfant sont suspendus après un fork réussi (). Utile pour le débogage.
SOCAT_VERSION (sortie)	Socat définit cette variable sur sa chaîne de version, par exemple “1.7.0.0” pour les versions publiées ou par exemple “1.6.0.1 + envvar” pour les versions temporaires; peut être utilisé dans les scripts invoqués par socat.
SOCAT_PID (sortie)	Socat définit cette variable sur son identifiant de processus. Dans le cas d'une option d'adresse de fork, SOCAT_PID obtient l'identifiant du processus enfant. fork pour exec et system ne modifie pas SOCAT_PID.
SOCAT_PPID (sortie)	Socat définit cette variable sur son identifiant de processus. En cas de fork, SOCAT_PPID conserve le pid du processus maître.
SOCAT_PEERADDR (sortie)	Avec les adresses de socket passives (toutes les adresses LISTEN et RECVFROM), cette variable est définie sur une chaîne décrivant l'adresse de socket homologue. Les informations de port ne sont pas incluses.
SOCAT_PEERPORT (sortie)	Avec les adresses de socket passives appropriées (TCP, UDP et SCTP-LISTEN et RECVFROM), cette variable est définie sur une chaîne contenant le numéro du port homologue.
SOCAT_SOCKADDR (sortie)	Avec toutes les adresses LISTEN, cette variable est définie sur une chaîne décrivant l'adresse de socket locale. Les informations de port ne sont pas incluses exemple
SOCAT_SOCKPORT (sortie)	Avec les adresses TCP-LISTEN, UDP-LISTEN et SCTP-LISTEN, cette variable est définie sur le port local.
SOCAT_TIMESTAMP (sortie)	Avec toutes les adresses RECVFROM où l'option d'adresse so-timestamp est appliquée, socat définit cette variable sur l'horodatage résultant.

Variable	Description
SOCAT_IP_OPTIONS (sortie)	Avec toutes les adresses RECVFROM basées sur IPv4 où l'option d'adresse ip-recvopts est appliquée, socat remplit cette variable avec les options IP du paquet reçu.
SOCAT_IP_DSTADDR (sortie)	Avec toutes les adresses RECVFROM basées sur IPv4 où l'option d'adresse ip-recvdstaddr (BSD) ou ip-pktinfo (autres plates-formes) est appliquée, socat définit cette variable sur l'adresse de destination du paquet reçu. Ceci est particulièrement utile pour identifier les paquets adressés par diffusion et multidiffusion.
SOCAT_IP_IF (sortie)	Avec toutes les adresses RECVFROM basées sur IPv4 où l'option d'adresse ip-recvif (BSD) ou ip-pktinfo (autres plates-formes) est appliquée, socat définit cette variable sur le nom de l'interface où le paquet a été reçu.
SOCAT_IP_LOCADDR (sortie)	Avec toutes les adresses RECVFROM basées sur IPv4 où l'option d'adresse ip-pktinfo est appliquée, socat définit cette variable sur l'adresse de l'interface où le paquet a été reçu.
SOCAT_IP_TOS (sortie)	Avec toutes les adresses RECVFROM basées sur IPv4 où l'option d'adresse ip-recvtos est appliquée, socat définit cette variable sur le type de service (TOS) du paquet reçu.
SOCAT_IP_TTL (sortie)	Avec toutes les adresses RECVFROM basées sur IPv4 où l'option d'adresse ip-recvttl est appliquée, socat définit cette variable sur la durée de vie (TTL) du paquet reçu.
SOCAT_IPV6_HOPLIMIT (sortie)	Avec toutes les adresses RECVFROM basées sur IPv6 où l'option d'adresse ipv6-recvhoplimit est appliquée, socat définit cette variable sur la valeur hoplimit du paquet reçu.
SOCAT_IPV6_DSTADDR (sortie)	Avec toutes les adresses RECVFROM basées sur IPv6 où l'option d'adresse ipv6-recvpktinfo est appliquée, socat définit cette variable sur l'adresse de destination du paquet reçu.
SOCAT_IPV6_TCLASS (sortie)	Avec toutes les adresses RECVFROM basées sur IPv6 où l'option d'adresse ipv6-recvtclass est appliquée, socat définit cette variable sur la classe de transfert du paquet reçu.
SOCAT_OPENSSL_X509_ISSUER (sortie)	Champ émetteur du certificat homologue
SOCAT_OPENSSL_X509_SUBJECT (sortie)	Champ objet du certificat homologue
SOCAT_OPENSSL_X509_COMMONNAME (sortie)	commonName entrées du sujet des certificats homologues. Les valeurs multiples sont séparées par //.
SOCAT_OPENSSL_X509_* (sortie)	toutes les autres entrées du sujet des certificats homologues
SOCAT_OPENSSL_X509V3_DNS (sortie)	Entrées DNS des extensions de certificats homologues - champ subjectAltName. Les valeurs multiples sont séparées par //.
HOSTNAME (entrée)	Est utilisé pour déterminer le nom d'hôte pour la journalisation (voir -lh).

Variable	Description
LOGNAME (entrée)	Est utilisé comme nom pour le nom d'utilisateur du client socks si aucun utilisateur socks n'est donné Avec les options su et su-d, **LOGNAME** est défini sur le nom d'utilisateur donné.
USER (entrée)	Est utilisé comme nom pour le nom d'utilisateur du client socks si aucun utilisateur socks n'est donné et LOGNAME est vide. Avec les options su et su-d, **USER** est défini sur le nom d'utilisateur donné.
SHELL (sortie)	Avec les options su et su-d, **SHELL** est défini sur le shell de connexion de l'utilisateur donné.
PATH (sortie)	Peut être défini avec le chemin d'option pour les adresses exec et système.
HOME (sortie)	Avec les options su et su-d, **HOME** est défini sur le répertoire de base de l'utilisateur donné.