netcat (nc) TCP/UDP Swiss army knife

CONCEPTS

nc copies stdin to the socket and the socket to stdout. Pipes and redirects do the rest.

stdin typed or piped
↓ sent over
nc client or -l server
↓ TCP / UDP
peer host:port
↓ replies to
stdout screen, file, pipe

Two variants exist: OpenBSD nc (-l <port>, -N, -k, -U, proxies) and traditional nc (-l -p <port>, -q, -e). Check with: nc -h.

Debian: nc may point to netcat-traditional; install netcat-openbsd for the OpenBSD flags. Ncat (nmap) is a third variant.

CONNECT

nc *<host> <port>*
Connect and talk interactively

nc -v *<host> <port>*
Verbose: report connection status

nc -w *<secs> <host> <port>*
Timeout for connect and idle

nc -n *<ip> <port>*
No DNS lookups

nc -4 / nc -6
Force IPv4 / IPv6 (OpenBSD)

nc -C *<host> <port>*
Send CRLF line endings (traditional)

nc -N *<host> <port>*
Close socket on stdin EOF (OpenBSD)

nc -q *<secs> <host> <port>*
Quit after stdin EOF (traditional)

nc -s *<addr> <host> <port>*
Choose source address

nc -p *<port> <host> <port>*
Choose source port

nc -i *<secs> <host> <port>*
Pause between lines sent

LISTEN

nc -l *<port>*
Listen once on a TCP port (OpenBSD)

nc -l -p *<port>*
Listen once (traditional syntax)

nc -lv *<port>*
Listen and show who connects

nc -lk *<port>*
Keep listening after a client leaves

nc -l *<addr> <port>*
Bind to one address only

nc -lu *<port>*
Listen for UDP datagrams

nc -lU *<path>*
Listen on a Unix socket (OpenBSD)

nc -W *<n>* **-l** *<port>*
Exit after n packets (OpenBSD)

SCAN PORTS

nc -zv *<host> <port>*
Test whether one port is open

nc -zv *<host> <from>-<to>*
Scan a range of ports

nc -zv *<host> <port1> <port2>*
Scan several ports

nc -zvw 2 *<host> <from>-<to>*
Scan with a short timeout

nc -zuv *<host> <port>*
UDP probe

nc -zvr *<host> <from>-<to>*
Scan in random order

TRANSFER DATA

nc -l *<port>* **>** *<file>*
Receive into a file

nc *<host> <port>* **<** *<file>*
Send a file

nc -u *<host> <port>*
Send lines as UDP datagrams

echo '*<text>***' | nc -u -w 1** *<host> <port>*
Fire a single UDP message

TRANSFER DATA (cont.)

tar cf - *<dir>* **| nc** *<host> <port>*
Stream a directory

nc -l *<port>* **| tar xf -**
Unpack a streamed directory

nc -U *<path>*
Connect to a Unix socket (OpenBSD)

nc -o *<file> <host> <port>*
Hex dump of traffic (traditional)

PROXY AND SHELL

nc -x *<proxy>*:*<port> <host> <port>*
Connect through a SOCKS5 proxy (OpenBSD)

nc -x *<proxy>*:*<port>* **-X connect** *<host> <port>*
Use an HTTP CONNECT proxy (OpenBSD)

nc -X 4 -x *<proxy> <host> <port>*
Pick SOCKS version 4, 4A or 5

nc -e *<prog>* **-l -p** *<port>*
Run a program per connection (traditional)

nc -g *<gw> <host> <port>*
Source-routing hop (traditional)

RECIPES

Check a service is reachable

```
nc -zvw 3 db.example.com 5432
# 'succeeded' / 'open' = port reachable
# timeout = filtered, refused = closed
```

Speak HTTP by hand

```
printf 'GET / HTTP/1.1\r\nHost: example.com\r\nConnection: close\r\n\r\n' \
  | nc -w 5 example.com 80
```

Copy a file between hosts

```
# receiver
nc -l 9000 > backup.tar.gz
# sender
nc -N receiver.example.com 9000 < backup.tar.gz
```

Throwaway chat or echo test

```
# host A
nc -lv 9000
# host B: type, lines appear on A
nc hostA 9000
```