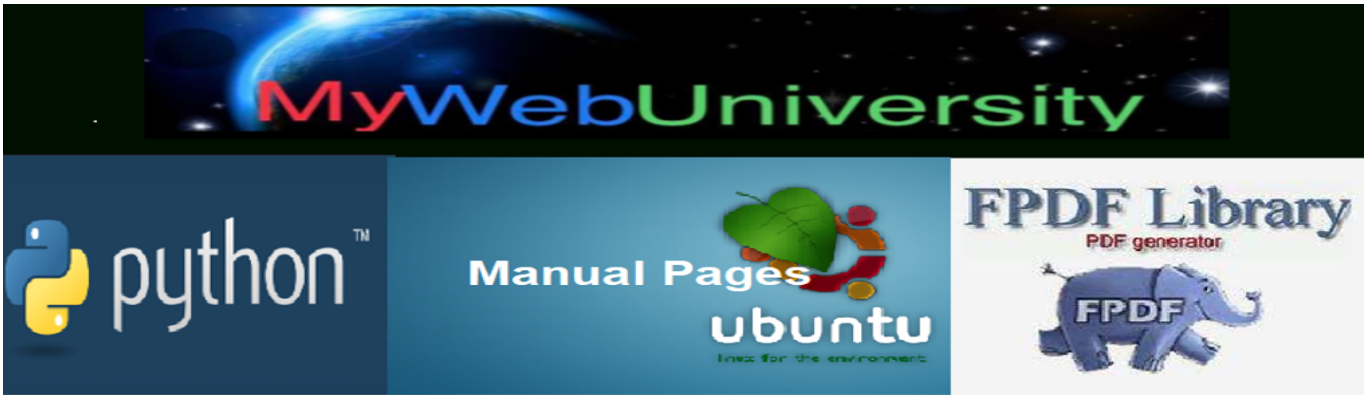

Rocky Enterprise Linux 9.2 Manual Pages on command 'nc_openbsd.1'

$ man nc_openbsd.1

NC(1) BSD General Commands Manual NC(1)

NAME

nc ? arbitrary TCP and UDP connections and listens

SYNOPSIS

nc [-46bCDdFhklNnrStUuvZz] [-I length] [-i interval] [-M ttl] [-m minttl] [-O length]
[-P proxy_username] [-p source_port] [-q seconds] [-s sourceaddr] [-T keyword]
[-V rtable] [-W recvlimit] [-w timeout] [-X proxy_protocol] [-x proxy_address[:port]]
[destination] [port]

DESCRIPTION

The nc (or netcat) utility is used for just about anything under the sun involving TCP, UDP, or UNIX-domain sockets. It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, do port scanning, and deal with both IPv4 and IPv6. Unlike telnet(1), nc scripts nicely, and separates error messages onto standard error instead of sending them to standard output, as telnet(1) does with some.

Common uses include:

- ? simple TCP proxies
- ? shell-script based HTTP clients and servers
- ? network daemon testing
- ? a SOCKS or HTTP ProxyCommand for ssh(1)
- ? and much, much more

The options are as follows:

-4 Use IPv4 addresses only.

-6 Use IPv6 addresses only.

-b Allow broadcast.

-C Send CRLF as line-ending. Each line feed (LF) character from the input data is translated into CR+LF before being written to the socket. Line feed characters that are already preceded with a carriage return (CR) are not translated. Received data is not affected.

-D Enable debugging on the socket.

-d Do not attempt to read from stdin.

-F Pass the first connected socket using sendmsg(2) to stdout and exit. This is useful in conjunction with -X to have nc perform connection setup with a proxy but then leave the rest of the connection to another program (e.g. ssh(1) using the ssh_config(5) ProxyUseFdpass option). Cannot be used with -U.

-h Print out the nc help text and exit.

-I length

Specify the size of the TCP receive buffer.

-i interval

Sleep for interval seconds between lines of text sent and received. Also causes a delay time between connections to multiple ports.

-k When a connection is completed, listen for another one. Requires -l. When used to? gether with the -u option, the server socket is not connected and it can receive UDP datagrams from multiple hosts.

-l Listen for an incoming connection rather than initiating a connection to a remote host. The destination and port to listen on can be specified either as non-optional arguments, or with options -s and -p respectively. Cannot be used together with -x or -z. Additionally, any timeouts specified with the -w option are ignored.

-M ttl Set the TTL / hop limit of outgoing packets.

-m minttl

Ask the kernel to drop incoming packets whose TTL / hop limit is under minttl.

-N shutdown(2) the network socket after EOF on the input. Some servers require this to finish their work.

-n Do not perform domain name resolution. If a name cannot be resolved without DNS, an error will be reported.

-O length

Specify the size of the TCP send buffer.

-P proxy_username

Specifies a username to present to a proxy server that requires authentication. If no username is specified then authentication will not be attempted. Proxy authenti? cation is only supported for HTTP CONNECT proxies at present.

-p source_port

Specify the source port nc should use, subject to privilege restrictions and avail? ability.

-q seconds

after EOF on stdin, wait the specified number of seconds and then quit. If seconds is negative, wait forever (default). Specifying a non-negative seconds implies -N.

-r Choose source and/or destination ports randomly instead of sequentially within a range or in the order that the system assigns them.

-S Enable the RFC 2385 TCP MD5 signature option.

-s sourceaddr

Set the source address to send packets from, which is useful on machines with multi? ple interfaces. For UNIX-domain datagram sockets, specifies the local temporary socket file to create and use so that datagrams can be received. Cannot be used to? gether with -x.

-T keyword

Change the IPv4 TOS/IPv6 traffic class value. keyword may be one of critical, inetcontrol, lowcost, lowdelay, netcontrol, throughput, reliability, or one of the DiffServ Code Points: ef, af11 ... af43, cs0 ... cs7; or a number in either hex or decimal.

-t Send RFC 854 DON'T and WON'T responses to RFC 854 DO and WILL requests. This makes it possible to use nc to script telnet sessions.

-U Use UNIX-domain sockets. Cannot be used together with -F or -x.

-u Use UDP instead of TCP. Cannot be used together with -x. For UNIX-domain sockets, use a datagram socket instead of a stream socket. If a UNIX-domain socket is used, a temporary receiving socket is created in /tmp unless the -s flag is given.

-V rtable

Set the routing table to be used.

-v Produce more verbose output.

-W recvlimit

Terminate after receiving recvlimit packets from the network.

-w timeout

Connections which cannot be established or are idle timeout after timeout seconds. The -w flag has no effect on the -l option, i.e. nc will listen forever for a con? nection, with or without the -w flag. The default is no timeout.

-X proxy_protocol

Use proxy_protocol when talking to the proxy server. Supported protocols are 4 (SOCKS v.4), 5 (SOCKS v.5) and connect (HTTPS proxy). If the protocol is not speci? fied, SOCKS version 5 is used.

-x proxy_address[:port]

Connect to destination using a proxy at proxy_address and port. If port is not specified, the well-known port for the proxy protocol is used (1080 for SOCKS, 3128 for HTTPS). An IPv6 address can be specified unambiguously by enclosing proxy_address in square brackets. A proxy cannot be used with any of the options -lsuU.

-Z DCCP mode.

-z Only scan for listening daemons, without sending any data to them. Cannot be used together with -l.

destination can be a numerical IP address or a symbolic hostname (unless the -n option is given). In general, a destination must be specified, unless the -l option is given (in which case the local host is used). For UNIX-domain sockets, a destination is required and is the socket path to connect to (or listen on if the -l option is given).

port can be specified as a numeric port number or as a service name. Port ranges may be specified as numeric port numbers of the form nn-mm. In general, a destination port must be specified, unless the -U option is given.

CLIENT/SERVER MODEL

It is quite simple to build a very basic client/server model using nc. On one console, start nc listening on a specific port for a connection. For example:

$ nc -l 1234

nc is now listening on port 1234 for a connection. On a second console (or a second machine), connect to the machine and port being listened on:

$ nc -N 127.0.0.1 1234

There should now be a connection between the ports. Anything typed at the second console

will be concatenated to the first, and vice-versa. After the connection has been set up, nc does not really care which side is being used as a ?server? and which side is being used as a ?client?. The connection may be terminated using an EOF (?^D?), as the -N flag was given.

There is no -c or -e option in this netcat, but you still can execute a command after con? nection being established by redirecting file descriptors. Be cautious here because opening a port and let anyone connected execute arbitrary command on your site is DANGEROUS. If you really need to do this, here is an example:

On ?server? side:

```
$ rm -f /tmp/f; mkfifo /tmp/f
$ cat /tmp/f | /bin/sh -i 2>&1 | nc -l 127.0.0.1 1234 > /tmp/f
```

On ?client? side:

```
$ nc host.example.com 1234
$ (shell prompt from host.example.com)
```

By doing this, you create a fifo at /tmp/f and make nc listen at port 1234 of address 127.0.0.1 on ?server? side, when a ?client? establishes a connection successfully to that port, /bin/sh gets executed on ?server? side and the shell prompt is given to ?client? side. When connection is terminated, nc quits as well. Use -k if you want it keep listening, but if the command quits this option won't restart it or keep nc running. Also don't forget to remove the file descriptor once you don't need it anymore:

```
$ rm -f /tmp/f
```

DATA TRANSFER

The example in the previous section can be expanded to build a basic data transfer model. Any information input into one end of the connection will be output to the other end, and input and output can be easily captured in order to emulate file transfer.

Start by using nc to listen on a specific port, with output captured into a file:

```
$ nc -l 1234 > filename.out
```

Using a second machine, connect to the listening nc process, feeding it the file which is to be transferred:

```
$ nc -N host.example.com 1234 < filename.in
```

After the file has been transferred, the connection will close automatically.

TALKING TO SERVERS

It is sometimes useful to talk to servers ?by hand? rather than through a user interface. It can aid in troubleshooting, when it might be necessary to verify what data a server is

sending in response to commands issued by the client. For example, to retrieve the home page of a web site:

```
$ printf "GET / HTTP/1.0\r\n\r\n" | nc host.example.com 80
```

Note that this also displays the headers sent by the web server. They can be filtered, us? ing a tool such as sed(1), if necessary.

More complicated examples can be built up when the user knows the format of requests re? quired by the server. As another example, an email may be submitted to an SMTP server us? ing:

```
$ nc [-C] localhost 25 << EOF
HELO host.example.com
MAIL FROM:<user@host.example.com>
RCPT TO:<user2@host.example.com>
DATA
Body of email.
.
QUIT
EOF
```

PORT SCANNING

It may be useful to know which ports are open and running services on a target machine. The -z flag can be used to tell nc to report open ports, rather than initiate a connection. Usu? ally it's useful to turn on verbose output to stderr by use this option in conjunction with -v option.

For example:

```
$ nc -zv host.example.com 20-30
Connection to host.example.com 22 port [tcp/ssh] succeeded!
Connection to host.example.com 25 port [tcp/smtp] succeeded!
```

The port range was specified to limit the search to ports 20 - 30, and is scanned by in? creasing order (unless the -r flag is set).

You can also specify a list of ports to scan, for example:

```
$ nc -zv host.example.com http 20 22-23
nc: connect to host.example.com 80 (tcp) failed: Connection refused
nc: connect to host.example.com 20 (tcp) failed: Connection refused
Connection to host.example.com port [tcp/ssh] succeeded!
```

nc: connect to host.example.com 23 (tcp) failed: Connection refused

The ports are scanned by the order you given (unless the -r flag is set).

Alternatively, it might be useful to know which server software is running, and which ver?
sions. This information is often contained within the greeting banners. In order to re?
trieve these, it is necessary to first make a connection, and then break the connection when
the banner has been retrieved. This can be accomplished by specifying a small timeout with
the -w flag, or perhaps by issuing a "QUIT" command to the server:

```
$ echo "QUIT" | nc host.example.com 20-30
SSH-1.99-OpenSSH_3.6.1p2
Protocol mismatch.
220 host.example.com IMS SMTP Receiver Version 0.84 Ready
```

EXAMPLES

Open a TCP connection to port 42 of host.example.com, using port 31337 as the source port,
with a timeout of 5 seconds:

```
$ nc -p 31337 -w 5 host.example.com 42
```

Open a UDP connection to port 53 of host.example.com:

```
$ nc -u host.example.com 53
```

Open a TCP connection to port 42 of host.example.com using 10.1.2.3 as the IP for the local
end of the connection:

```
$ nc -s 10.1.2.3 host.example.com 42
```

Create and listen on a UNIX-domain stream socket:

```
$ nc -lU /var/tmp/dsocket
```

Connect to port 42 of host.example.com via an HTTP proxy at 10.2.3.4, port 8080. This exam?
ple could also be used by ssh(1); see the ProxyCommand directive in ssh_config(5) for more
information.

```
$ nc -x10.2.3.4:8080 -Xconnect host.example.com 42
```

The same example again, this time enabling proxy authentication with username ?ruser? if the
proxy requires it:

```
$ nc -x10.2.3.4:8080 -Xconnect -Pruser host.example.com 42
```

SEE ALSO

cat(1), ssh(1)

AUTHORS

Original implementation by *Hobbit* <hobbit@avian.org>.

Rewritten with IPv6 support by

Eric Jackson <ericj@monkey.org>.

Modified for Debian port by Aron Xu ?aron@debian.org?.

CAVEATS

UDP port scans using the -uz combination of flags will always report success irrespective of the target machine's state. However, in conjunction with a traffic sniffer either on the target machine or an intermediary device, the -uz combination could be useful for communica? tions diagnostics. Note that the amount of UDP traffic generated may be limited either due to hardware resources and/or configuration settings.