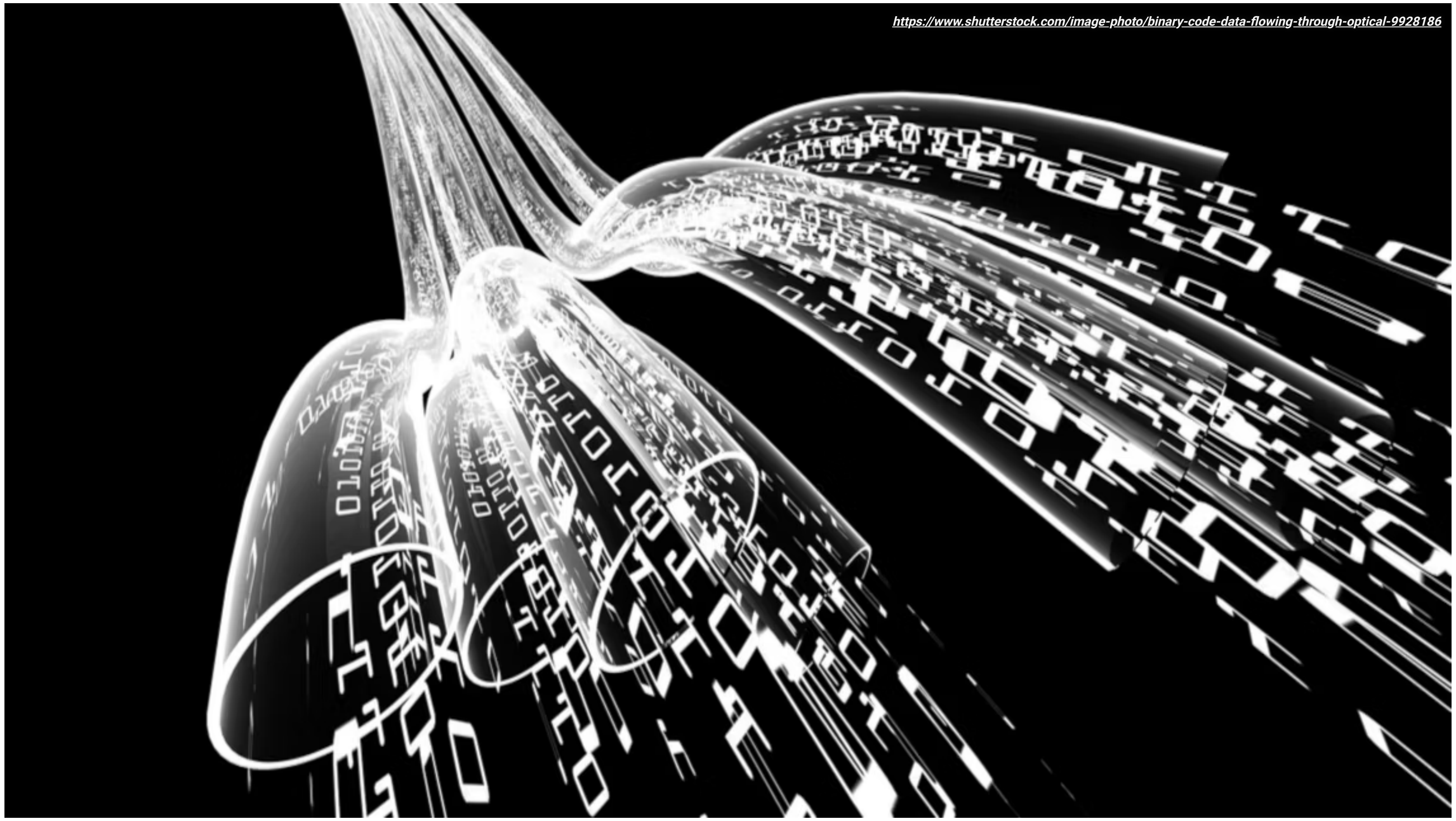
How to Read Data From a Socket in Bash on Linux

Collect data on the Linux command line from network socket connections.

BY DAVE MCKAY

PUBLISHED APR 20, 2022


https://www.shutterstock.com/image-photo/binary-code-data-flowing-through-optical-9928186

Quick Links

Socket Clients and Servers
Relationships Are Never Easy
Listening on a Socket
Sending Data to a File
Sending Data to a File and the Terminal Window
Accepting Multiple Connections
Connecting to a Server
Quick and Easy

The Linux command line lets you retrieve data by either listening on a socket or connecting to a socket. The data can be captured in a text file. We show you how.

Socket Clients and Servers

Sockets allow networked software to communicate. They were first implemented in the 4.2BSD Unix operating system, which was created at the University of California, Berkeley, in 1983. They were quickly adopted by System V Unix and Microsoft Windows.

RELATED:
What Does "Everything Is A File" Mean In Linux?

A socket is an endpoint of a software network connection, abstracted so that it can be treated as a file handle. That means it fits with the general Unix and Linux design principle of "everything is a file." We don't mean the physical socket on the wall that you plug your network cable into.

If a program connects to a socket on another piece of software, it is considered to be the client of the other software. Software that allows other software to request connections is called the server. These terms are used independently from other uses of client and server in the IT world. To avoid confusion they are sometimes called socket client and socket server to remove ambiguity. We're going to call them clients and servers.

Link copied to clipboard

ckets are implemented as an application programming interface (API), allowing software developers to call on the socket functionality from within their code. That's fine if you're a programmer, but what if you're not? Or perhaps you are, but your use case doesn't warrant writing an application? Linux provides command-line tools that let you use---basic---socket servers and socket clients, according to your needs, to retrieve or receive data from other socket-enabled processes.

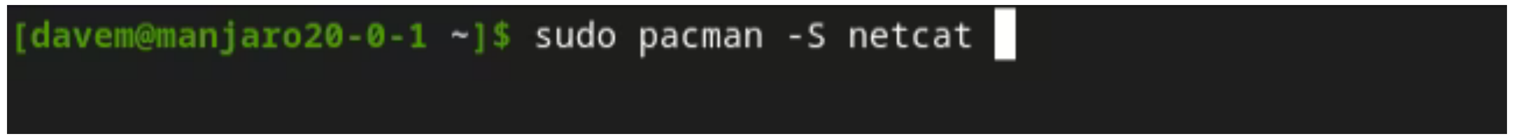
RELATED:
What Is An API, And How Do Developers Use Them?

Relationships Are Never Easy

The programs we're going to use are `nc` and `ncat`. These two utilities have a strange relationship. The `nc` program is a rewrite of `ncat`, which is much older than `nc` . But `ncat` has been rewritten too, and it now lets us do some things `nc` can't. And there are many implementations of `ncat`, which itself is a derivative of a tool called `netcat`. On top of that, on most distributions, `nc` is a symbolic link to `ncat` and not a separate program.

We checked recent Arch, Manjaro, Fedora, and Ubuntu distributions. The only one that required the tools to be installed was Manjaro. On Manjaro, you need to install the `netcat` package to get `nc`, but you don't get `ncat`, you get `netcat`. And on Manjaro, `nc` is a symbolic link to `netcat`.

```
sudo pacman -S netcat
```



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

The bottom line is, on Manjaro use `netcat` when you see `ncat` in the examples in this article.

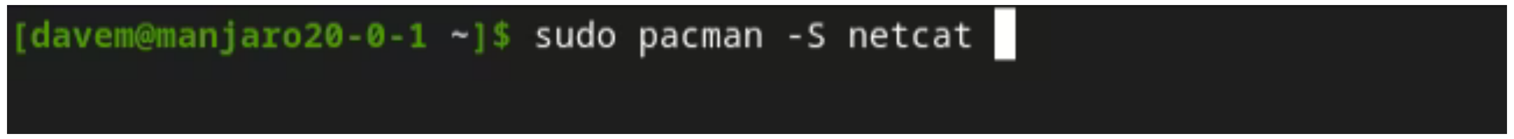
Listening on a Socket

If software listens for incoming socket connections, it's acting as a server. Any data that comes over the socket connection is said to be received by the server. We can replicate this behavior very easily using `nc`. Any received data is displayed in the terminal window.

We need to tell `nc` to listen for connections, using the `-l` (listen) option, and we need to specify the port we are going to listen for connections on. Any client programs or processes that try to connect to this instance of `nc` must use the same port. We tell `nc` which port to listen on by using the `-p` (port) option.

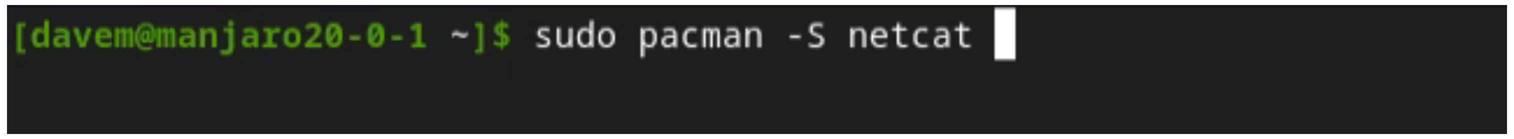
This command starts `nc` as a socket server, listening for a connection on port 6566:

```
nc -l -p 6566
```



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

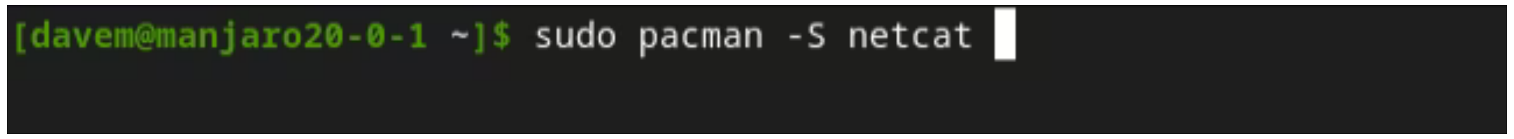
While it waits for an incoming connection, `nc` produces no output. Once a connection is made, any retrieved information is displayed in the terminal window. Here, a connection has been made by a client program that is identifying itself as "client 1."



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

Everything displayed by `nc` is received from the client. This client happens to send its name, and a numbered message containing the time and date.

When the client breaks its connection, `nc` terminates and you're returned to the terminal prompt.

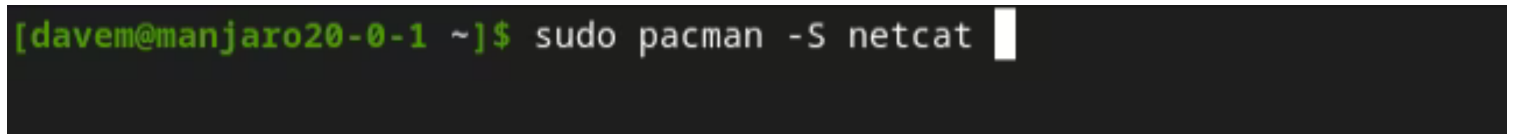


```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

Sending Data to a File

To capture the data from the client in a file, we can send the output from `nc` to a file using redirection. This command saves the received data in a file called "logfile.txt."

```
nc -l -p 6566 > logfile.txt
```

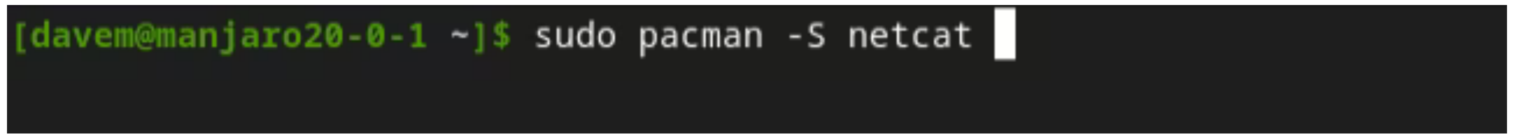


```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

You won't see any output---it is going into the file---and, paradoxically, you won't know if a connection has occurred until `nc` terminates. Being returned to the command prompt indicates a connection has occurred and has been terminated by the client.

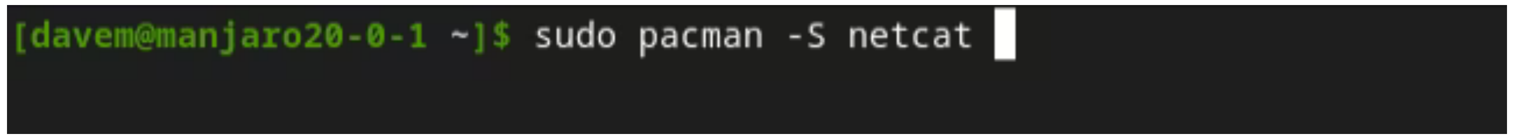
We can use `less` to review the contents of the "logfile.txt" file.

```
less logile.txt
```



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

You can then scroll through the data, and search using less's built-in functions.



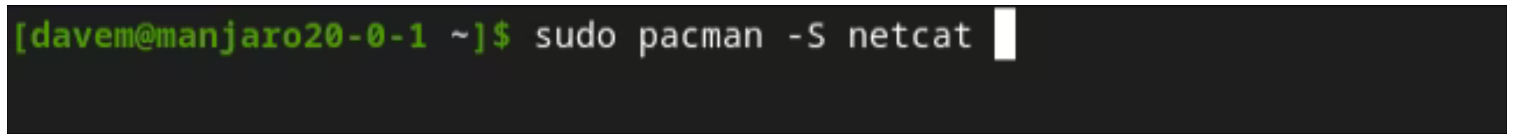
```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

RELATED:
How To Use The Less Command On Linux

Sending Data to a File and the Terminal Window

If you want to see the data scrolling by in the terminal window and have it sent to a file at the same time, pipe the output from `nc` into `tee` .

```
nc -l -p 6566 | tee logfile.txt
```



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

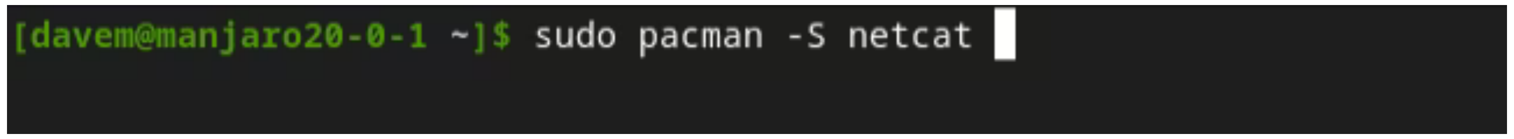
Accepting Multiple Connections

All that's fine, but it does have limitations. We can only accept one connection. We're limited to receiving data from a single client. Also, when that client drops the connection our socket server `nc` terminates.

If you need to accept multiple connections we need to use `ncat`. we'll need to tell `ncat` to listen, and to use a particular port, just like we did with `nc`. But we'll also use the `-k` (keep alive) option. This tells `ncat` to keep running and accepting connections from clients even when the last active connection drops.

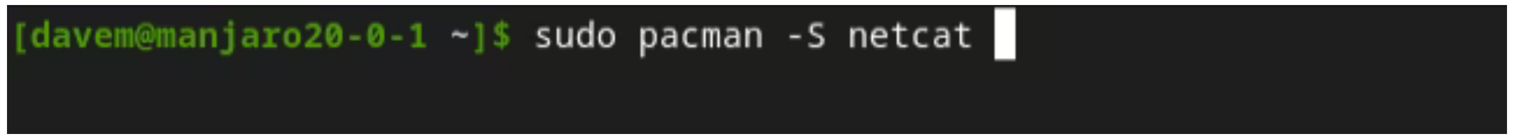
This means `ncat` will run until we choose to terminate it with "Ctrl-C." New connections will be accepted whether `ncat` is currently connected to any clients or not.

```
ncat -k -l -p 6566
```



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

We can see the data from the different clients appearing in the output from `ncat` as they connect.



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

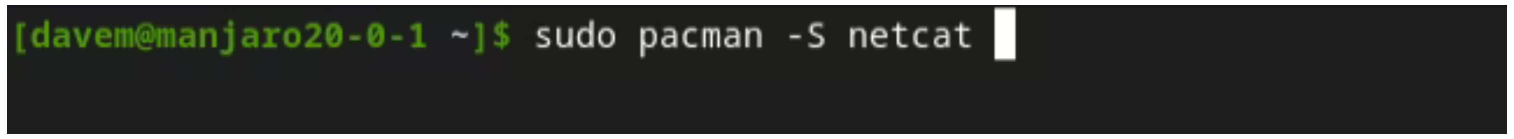
Connecting to a Server

We can also use `nc` as a socket client and connect to another program that is accepting connections, and is acting as a server. In this scenario, `nc` is the socket client. To do this we need to tell `nc` where the server software is located on the network.

One way to do this is to provide an IP address and a port number. If the server is on the same pc that we're running `nc` on, we can use the loopback IP address of 127.0.0.1. Not that there are no flags used to indicate the server address and port number. We just provide the appropriate values.

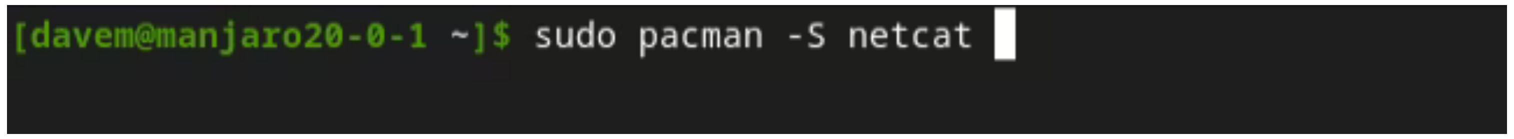
To connect to a server on the same PC, and using port 6566, we could use the loopback IP address. The command to use is:

```
nc 127.0.0.1 6566
```



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

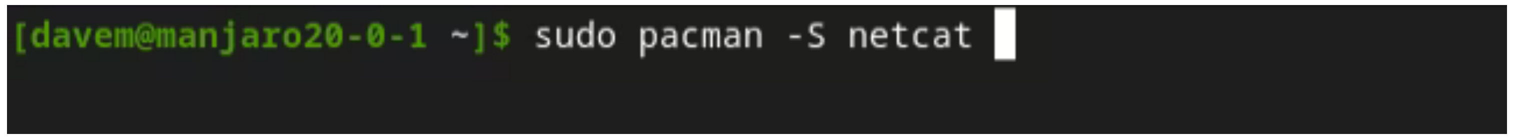
Data that `nc` retrieves from the server scrolls by in the terminal window.



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

If you know the network name of the computer running the server software, you can use that instead of the IP address.

```
nc sulaco 6566
```



```
[davem@manjaro20-0-1 ~]$ sudo pacman -S netcat
```

Use "Ctrl+C" to break a connection.

Quick and Easy

`nc` and `ncat` fit the bill when you don't want to write a custom socket handler, but you need to gather data from some socket-enabled source. Redirecting the output into a file lets you review the output using `less`, and parse the file using utilities like `grep`.

RELATED:
How To Use The Grep Command On Linux

Related Topics

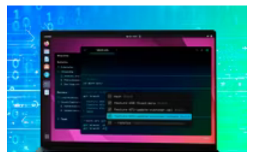
LINUX FEATURES

About The Author

Dave McKay • Author

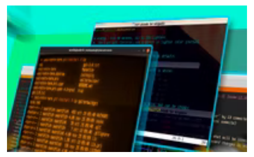
Dave McKay first used computers when punched paper tape was in vogue, and he has been programming ever since. After over 30 years in the IT industry, he became a full-time technology journalist. His first published article,...

RECOMMENDED ARTICLES



Linux & macOS Terminal

I Tried Out an AI-Powered Linux Terminal, Here's How



Linux & macOS Terminal

Your Default Linux Terminal Emulator Is Dull, So Take