How to Create Virtual Serial Port in Linux

Last updated: July 6, 2023

Written by: Sidrah Abdullah (https://www.baeldung.com/linux/author/sidrahabdullah)

Administration (https://www.baeldung.com/linux/category/administration)

Networking (https://www.baeldung.com/linux/category/networking)

socat (https://www.baeldung.com/linux/tag/socat)

(https://www.baeldung.com/linux/devops-lead-with-experience-in-lamp-lemp-and-ci-cd-stack)

1. Introduction

There are several ways to create a virtual serial port in Linux to test and debug serial communication protocols. In this tutorial, we'll first discuss the *socat* (/linux/socat-command#:~:text=Socat%20is%20a%20flexible%2C%20multi,%2C%20TCP%2C%20or%20standard%20input.) utility to create a virtual serial port. After that, we'll look at the *tty0tty* (https://sourceforge.net/projects/tty0tty/) kernel module update for creating a virtual port.

2. Virtual Serial Ports

Virtual serial ports are a logical representation of physical serial ports. These ports don't connect with the serial port. A virtual serial port allows us to emulate a physical serial port, thereby removing the requirements of cable and physical communication interface.

Additionally, a virtual port allows software packages to communicate with internal memory using the null modem emulator. The null modem emulator is a virtual driver for Linux that enables virtual serial port communication.

3. Create a Virtual Serial Port Using *socat* Utility

The first method to create a virtual serial port is using the *socat* utility. **This utility pipes the port traffic to executables using standard input and output.** Here's the syntax for *socat*:

```
socat [OPTIONS] <ADDRESS1> <ADDRESS2>
```

The two addresses can be:

- *stdio*: standard input/output
- *udp:host:port*: UDP connection to host and port
- *tcp:host:port*: TCP connection to host and port
- *file:<filename>*: file or FIFO (named pipe)
- *pty*: pseudo-terminal
- *pty,link=<filename>*: pseudo-terminal with a symbolic link
- *pty,rawer,link=<filename>*: pseudo-terminal with a raw data transfer and a symbolic link

3.1. Install *socat* Utility

Although the *socat* utility is usually available by default, we can install it on our machine. To install it, we'll use the *apt-get* (https://linux.die.net/man/8/apt-get) package or the *yum* (/linux/yum-and-apt) package:

```
# Debian-based Linux
$ sudo apt-get install -y socat
# RHEL-based Linux
$ sudo yum install -y socat
```

This installs *socat* on the Linux machine.

3.2. Create Simple Virtual Serial Port

For instance, to create two pseudo terminals with the debug mode on, we'd type *socat* with the *-d* option and *pty,rawer* in place of the address:

```
$ socat -d -d pty,rawer,echo=0 pty,rawer,echo=0
2023/05/04 09:32:35 socat[606] N PTY is /dev/pts/2
2023/05/04 09:32:35 socat[606] N PTY is /dev/pts/3
2023/05/04 09:32:35 socat[606] N starting data transfer loop
with FDs [5,5] and [7,7]
```

The *-d -d* option after the *socat* command prints fatal, error, warning, and notice messages. In addition, the *pty,rawer,echo=0* arguments create two pseudo terminals under */dev/pts*. These terminals act as an end-point for serial connection. To connect these two, we can use any serial communication tool, such as minicom (https://help.ubuntu.com/community/Minicom). For example, to connect */dev/pts/2* at a baud rate of *9600*, we'd run:

```
$ minicom -D /dev/pts/2 -b 9600
```

This connects */dev/pts/2* with a baud rate of *9600*.

3.3. Create Serial Port With Baud Rate

Alternatively, we can create a virtual port with a custom baud rate by adding the baud argument to the *socat* command:

```
$ socat -d -d pty,rawer,echo=0,link=/tmp/ttyV0, baud=115200
pty,rawer,echo=0,link=/tmp/ttyV1,baud=115200
```

This creates a pair of virtual rates with custom names of */tmp/ttyV0* and */tmp/ttyV1*.

4. Create a Virtual Serial Port Using *ttyotty* Kernel Module

The Linux modem emulator, ***ttyotty*, creates a virtual serial port driver for Linux.** Additionally, we can use these ports to establish communication between serial applications. In other words, **it provides a**

pair of virtual TTY devices that we can use as a null-modem cable replacement.

4.1. Install *tty0tty* on Ubuntu

To install *tty0tty* on Ubuntu, we'll use the *apt-get* command:

```
$ sudo apt-get install tty0tty
```

We'll follow the instructions on the screen to complete the installation.

On the other hand, we can install it from the website. First, we'll download the package from SourceForge (http://sourceforge.net/projects/tty0tty/files/). Next, we'll clone this GitHub repository (https://github.com/freemed/tty0tty). After that, we'll extract the package using the *tar* (/linux/tar-command) command:

```
$ tar xf tty0tty-1.2.tgz
```

Next, we'll build the kernel modules:

```
$ cd tty0tty-1.2/module
$ make
```

After that, we'll copy the new kernel module in the current kernel module directory:

```
$ sudo cp tty0tty.ko /lib/modules/$(uname -
r)/kernel/drivers/misc/
```

Lastly, we'll load the module using the depmod (https://man7.org/linux/man-pages/man8/depmod.8.html) command:

```
$ sudo depmod
```

The command analyzes the dependent kernel modules and the associated map file consisting of information related to these dependency modules. This ensures that all the required modules are loaded before loading the *tty0tty* module.

4.2. Create a Virtual Serial Port

Now, to create and use the serial port, we use the *modprobe* (https://man7.org/linux/man-pages/man8/modprobe.8.html) command:

```
$ sudo modprobe tty0tty
```

After that, we'll give permission to the new serial port using *chmod* (/linux/chown-chmod-permissions):

```
$ sudo chmod 666 /dev/tnt*
```

We can now access the serial ports as */dev/tnt0* and */dev/tnt1*. To connect these two devices, we'll use a serial communication tool such as minicom (https://help.ubuntu.com/community/Minicom):

```
$ minicom /dev/tnt0 -b 9600
```

In addition, we can change the baud rate of the ports created using *tty0tty*:

```
$ sudo modprobe tty0tty baud_base=115200
```

This should create the port with the baud rate of *115200*.

4.3. Make Module Update Persistent

To make this kernel module change persistent, we'll edit the */etc/modules* or */etc/modules.conf*. For this step, we'll use the *nano* (/linux/files-vi-nano-emacs) text editor:

```
$ nano /etc/modules
```

After that, we'll add the module name at the end of the file:

```
ttyOtty
```

Lastly, we'll press *Ctrl + S* to save and *Ctrl + X* to exit the *nano* editor.

However, it's crucial to remember that this change will be overwritten by the kernel updates. To elaborate, if we update the kernel, we'll have to build the *ttyOtty* module once again by following all the steps.

5. Conclusion

To conclude, by using virtual serial ports, we can test serial communication protocols without accessing physical hardware. In this article, we've discussed two ways to create virtual ports, *socat* utility, and *ttyOtty* kernel module. The *socat* module offers more flexibility and advantage than *ttyOtty*, making it a powerful tool for advanced users.

Comments are closed on this article!

CATEGORIES

ADMINISTRATION (/LINUX/CATEGORY/ADMINISTRATION)
FILES (/LINUX/CATEGORY/FILES)
FILESYSTEMS (/LINUX/CATEGORY/FILESYSTEMS)
INSTALLATION (/LINUX/CATEGORY/INSTALLATION)

NETWORKING (/LINUX/CATEGORY/NETWORKING)
PROCESSES (/LINUX/CATEGORY/PROCESSES)
SCRIPTING (/LINUX/CATEGORY/SCRIPTING)
SEARCH (/LINUX/CATEGORY/SEARCH)
SECURITY (/LINUX/CATEGORY/SECURITY)
WEB (/LINUX/CATEGORY/WEB)

SERIES

LINUX ADMINISTRATION (/LINUX/LINUX-ADMINISTRATION-SERIES)
LINUX FILES (/LINUX/LINUX-FILES-SERIES)
LINUX PROCESSES (/LINUX/LINUX-PROCESSES-GUIDE)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE FULL ARCHIVE (HTTPS://WWW.BAELDUNG.COM/LINUX/FULL_ARCHIVE)
EDITORS (HTTPS://WWW.BAELDUNG.COM/EDITORS)

TERMS OF SERVICE (HTTPS://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
PRIVACY POLICY (HTTPS://WWW.BAELDUNG.COM/PRIVACY-POLICY)
COMPANY INFO (HTTPS://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
CONTACT (/CONTACT)