

How to find all serial devices (ttyS, ttyUSB, ..) on Linux without opening them?



Launch yourself.



What is the proper way to get a list of all available serial ports/devices on a Linux system?

In other words, when I iterate over all devices in `/dev/`, how do I tell which ones are serial ports in the classic way, that is, those usually supporting baud rates and [RTS/CTS](#) flow control?

The solution would be coded in C.

I ask because I am using a third-party library that does this clearly wrong: It appears to only iterate over `/dev/ttys*`. The problem is that there are, for instance, serial ports over USB (provided by USB-RS232 adapters), and those are listed under `/dev/ttyUSB*`. And reading the [Serial-HOWTO at Linux.org](#), I get the idea that there'll be other name spaces as well, as time comes.

So I need to find the official way to detect serial devices. The problem is that none appears to be documented, or I can't find it.

I imagine one way would be to open all files from `/dev/tty*` and call a specific `ioctl()` on them that is only available on serial devices. Would that be a good solution, though?

Update

[hrickards](#) suggested to look at the source for "setserial". Its code does exactly what I had in mind:

First, it opens a device with:

```
fd = open (path, O_RDWR | O_NONBLOCK)
```

Then it invokes:

```
ioctl (fd, TIOCGSERIAL, &serinfo)
```

If that call returns no error, then it's a serial device, apparently.

I found similar code in [Serial Programming/termios](#), which suggested to also add the `O_NOCTTY` option.

There is one problem with this approach, though:

When I tested this code on BSD Unix (that is, Mac OS X), it worked as well. **However***, serial devices that are provided through Bluetooth cause the system (driver) to try to connect to the Bluetooth device, which takes a while before it'll return with a timeout error. This is caused by just opening the device. And I can imagine that similar things can happen on Linux as well - ideally, I should not need to open the device to figure out its type. I wonder if there's also a way to invoke `ioctl` functions without an `open`, or open a device in a way that it does not cause connections to be made?

What should I do?

linux serial-port

edited Oct 10 '12 at 19:56



[dsolimano](#)
5,671 3 23 39

asked Mar 27 '10 at 16:56



[Thomas Tempelmann](#)
3,729 2 37 61

8 Answers

The `/sys` filesystem should contain plenty information for your quest. My system (2.6.32-40-generic #87-Ubuntu) suggests:

```
/sys/class/tty
```

Which gives you descriptions of all TTY devices known to the system. A trimmed down example:

```
# ll /sys/class/tty/ttyUSB*
lrwxrwxrwx 1 root root 0 2012-03-28 20:43 /sys/class/tty/ttyUSB0 ->
../../../../devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.4/2-1.4:1.0/ttyUSB0/tty/ttyUSB0/
lrwxrwxrwx 1 root root 0 2012-03-28 20:44 /sys/class/tty/ttyUSB1 ->
../../../../devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.3/2-1.3:1.0/ttyUSB1/tty/ttyUSB1/
```

Following one of these links:

```
# ll /sys/class/tty/ttyUSB0/
insgesamt 0
drwxr-xr-x 3 root root 0 2012-03-28 20:43 ./
```

```
drwxr-xr-x 3 root root    0 2012-03-28 20:43 ../
-r--r--r-- 1 root root 4096 2012-03-28 20:49 dev
lrwxrwxrwx 1 root root    0 2012-03-28 20:43 device -> ../../../../ttyUSB0/
drwxr-xr-x 2 root root    0 2012-03-28 20:49 power/
lrwxrwxrwx 1 root root    0 2012-03-28 20:43 subsystem ->
../../../../../../../../../../../../class/tty/
-rw-r--r-- 1 root root 4096 2012-03-28 20:43 uevent
```

Here the `dev` file contains this information:

```
# cat /sys/class/tty/ttyUSB0/dev
188:0
```

This is the major/minor node. These can be searched in the `/dev` directory to get user-friendly names:

```
# ll -R /dev |grep "188, *0"
crw-rw---- 1 root dialout 188,    0 2012-03-28 20:44 ttyUSB0
```

The `/sys/class/tty` dir contains all TTY devices but you might want to exclude those pesky virtual terminals and pseudo terminals. I suggest you examine only those which have a `device/driver` entry:

```
# ll /sys/class/tty/*/device/driver
lrwxrwxrwx 1 root root 0 2012-03-28 19:07 /sys/class/tty/ttyS0/device/driver ->
../../../../bus/pnp/drivers/serial/
lrwxrwxrwx 1 root root 0 2012-03-28 19:07 /sys/class/tty/ttyS1/device/driver ->
../../../../bus/pnp/drivers/serial/
lrwxrwxrwx 1 root root 0 2012-03-28 19:07 /sys/class/tty/ttyS2/device/driver ->
../../../../bus/platform/drivers/serial8250/
lrwxrwxrwx 1 root root 0 2012-03-28 19:07 /sys/class/tty/ttyS3/device/driver ->
../../../../bus/platform/drivers/serial8250/
lrwxrwxrwx 1 root root 0 2012-03-28 20:43 /sys/class/tty/ttyUSB0/device/driver ->
../../../../../../../../bus/usb-serial/drivers/ftdi_sio/
lrwxrwxrwx 1 root root 0 2012-03-28 21:15 /sys/class/tty/ttyUSB1/device/driver ->
../../../../../../../../bus/usb-serial/drivers/ftdi_sio/
```

answered Mar 28 '12 at 19:26



A.H.

29.4k 6 51 70

Work on work you love. From home.



stackoverflowcareers

In recent kernels (not sure since when) you can list the contents of `/dev/serial` to get a list of the serial ports on your system. They are actually symlinks pointing to the correct `/dev/` node:

```
flu0@laptop:~$ ls /dev/serial/
total 0
drwxr-xr-x 2 root root 60 2011-07-20 17:12 by-id/
drwxr-xr-x 2 root root 60 2011-07-20 17:12 by-path/
flu0@laptop:~$ ls /dev/serial/by-id/
total 0
lrwxrwxrwx 1 root root 13 2011-07-20 17:12 usb-Prolific_Technology_Inc._USB-
Serial_Controller-if00-port0 -> ../../ttyUSB0
flu0@laptop:~$ ls /dev/serial/by-path/
total 0
lrwxrwxrwx 1 root root 13 2011-07-20 17:12 pci-0000:00:0b.0-usb-0:3:1.0-port0 ->
../../ttyUSB0
```

This is a USB-Serial adapter, as you can see. Note that when there are no serial ports on the system, the `/dev/serial/` directory does not exist. Hope this helps :).

answered Jul 20 '11 at 21:17



flu0

131 1 3

2 This is a function of udev (specifically its configuration in `/lib/udev/rules.d/??-persistent-serial.rules`), which was introduced in 2.5. – [ergosys](#) Aug 14 '13 at 4:16

I'm doing something like the following code. It works for USB-devices and also the stupid serial8250-devuices that we all have 30 of - but only a couple of them really works.

Basically I use concept from previous answers. First enumerate all tty-devices in `/sys/class/tty/`. Devices that does not contain a `/device` subdir is filtered away. `/sys/class/tty/console` is such a device. Then the devices actually containing a devices in then accepted as valid serial-port depending on the target of the driver-symlink fx.

```
$ ls -al /sys/class/tty/ttyUSB0//device/driver
lrwxrwxrwx 1 root root 0 sep  6 21:28 /sys/class/tty/ttyUSB0//device/driver ->
```

```
../../../../bus/platform/drivers/usbserial
```

and for ttyS0

```
$ ls -al /sys/class/tty/ttyS0//device/driver
lrwxrwxrwx 1 root root 0 sep  6 21:28 /sys/class/tty/ttyS0//device/driver ->
../../../../bus/platform/drivers/serial8250
```

All drivers driven by serial8250 must be probes using the previously mentioned ioctl.

```
if (ioctl(fd, TIOCGSERIAL, &serinfo)==0) {
    // If device type is no PORT_UNKNOWN we accept the port
    if (serinfo.type != PORT_UNKNOWN)
        the_port_is_valid
```

Only port reporting a valid device-type is valid.

The complete source for enumerating the serialports looks like this. Additions are welcome.

```
#include <stdlib.h>
#include <dirent.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <linux/serial.h>

#include <iostream>
#include <list>

using namespace std;

static string get_driver(const string& tty) {
    struct stat st;
    string devicedir = tty;

    // Append '/device' to the tty-path
    devicedir += "/device";

    // Stat the devicedir and handle it if it is a symlink
    if (lstat(devicedir.c_str(), &st)==0 && S_ISLNK(st.st_mode)) {
        char buffer[1024];
        memset(buffer, 0, sizeof(buffer));

        // Append '/driver' and return basename of the target
        devicedir += "/driver";

        if (readlink(devicedir.c_str(), buffer, sizeof(buffer)) > 0)
            return basename(buffer);
    }
    return "";
}

static void register_comport( list<string>& comList, list<string>& comList8250,
const string& dir) {
    // Get the driver the device is using
    string driver = get_driver(dir);

    // Skip devices without a driver
    if (driver.size() > 0) {
        string devfile = string("/dev/") + basename(dir.c_str());

        // Put serial8250-devices in a seperate list
        if (driver == "serial8250") {
            comList8250.push_back(devfile);
        } else
            comList.push_back(devfile);
    }
}

static void probe_serial8250_comports(list<string>& comList, list<string>
comList8250) {
    struct serial_struct serinfo;
    list<string>::iterator it = comList8250.begin();

    // Iterate over all serial8250-devices
    while (it != comList8250.end()) {

        // Try to open the device
        int fd = open((*it).c_str(), O_RDWR | O_NONBLOCK | O_NOCTTY);

        if (fd >= 0) {
            // Get serial_info
            if (ioctl(fd, TIOCGSERIAL, &serinfo)==0) {
                // If device type is no PORT_UNKNOWN we accept the port
                if (serinfo.type != PORT_UNKNOWN)
                    comList.push_back(*it);
            }
            close(fd);
        }
        it ++;
    }
}
```

```
list<string> getComList() {
    int n;
    struct dirent **namelist;
    list<string> comList;
    list<string> comList8250;
    const char* sysdir = "/sys/class/tty/";

    // Scan through /sys/class/tty - it contains all tty-devices in the system
    n = scandir(sysdir, &namelist, NULL, NULL);
    if (n < 0)
        perror("scandir");
    else {
        while (n--) {
            if (strcmp(namelist[n]->d_name, "..") && strcmp(namelist[n]-
>d_name, ".")) {

                // Construct full absolute file path
                string devicedir = sysdir;
                devicedir += namelist[n]->d_name;

                // Register the device
                register_comport(comList, comList8250, devicedir);
            }
            free(namelist[n]);
        }
        free(namelist);
    }

    // Only non-serial8250 has been added to comList without any further testing
    // serial8250-devices must be probe to check for validity
    probe_serial8250_comports(comList, comList8250);

    // Return the list of detected comports
    return comList;
}

int main() {
    list<string> l = getComList();

    list<string>::iterator it = l.begin();
    while (it != l.end()) {
        cout << *it << endl;
        it++;
    }

    return 0;
}
```

edited Sep 6 '12 at 20:11

answered Sep 6 '12 at 13:46



Søren Holm

61 1 3

Lone link is [considered a poor answer](#) since it is meaningless by itself and target resource is not guaranteed to be alive in the future. Please try to include at least summary of information you are linking to. – [jok](#) Sep 6 '12 at 14:01

This is awesome - thanks Søren! – [Michael Burr](#) Jul 17 '13 at 23:03

I think I found the answer in my kernel source documentation: `/usr/src/linux-2.6.37-rc3/Documentation/filesystems/proc.txt`

1.7 TTY info in `/proc/tty`

Information about the available and actually used tty's can be found in the directory `/proc/tty`. You'll find entries for drivers and line disciplines in this directory, as shown in Table 1-11.

Table 1-11: Files in `/proc/tty`

File	Content
<code>drivers</code>	list of drivers and their usage
<code>ldiscs</code>	registered line disciplines
<code>driver/serial</code>	usage statistic and status of single tty lines

To see which tty's are currently in use, you can simply look into the file `/proc/tty/drivers`:

```
> cat /proc/tty/drivers
pty_slave    /dev/pts      136  0-255 pty:slave
pty_master   /dev/ptm      128  0-255 pty:master
pty_slave    /dev/ttyp     3    0-255 pty:slave
pty_master   /dev/pty      2    0-255 pty:master
serial       /dev/cua      5    64-67 serial:callout
serial       /dev/ttyS     4    64-67 serial
/dev/tty0    /dev/tty0     4    0 system:vtmaster
/dev/ptmx    /dev/ptmx     5    2 system
/dev/console /dev/console  5    1 system:console
/dev/tty     /dev/tty      5    0 system:/dev/tty
unknown     /dev/tty      4    1-63 console
```

Here is a link to this file: http://git.kernel.org/?p=linux/kernel/git/next/linux-next.git;a=blob_plain;f=Documentation/filesystems/proc.txt;hb=e8883f8057c0f7c9950fa9f20568f37bfa62f34a

answered Jan 15 '11 at 19:17

 **mk2**
51 1 1

Yes, that seems to be working. However, this solution requires me to read a text file and parse it. I wonder if there's a better way, i.e. a API that lets me get these contents in a structured binary format. – [Thomas Tempelmann](#) Jun 28 '11 at 14:26

setserial with the -g option appears to do what you want and the C source is available at <http://www.koders.com/c/fid39344DABD14604E70DF1B8FEA7D920A94AF78BF8.aspx>.

answered Mar 27 '10 at 17:05

 **hrickards**
516 1 6 20

I looked at the code and it has the flaw I explain in my question at the end as it has to open the device, which may already lead to a connection attempt - which in turn is not good. But then, maybe Linux drivers are smarter than current OSX driver when it comes to bluetooth support, as they won't open a connection right away? Who knows? Maybe I should start a new question to clarify that specifically. If it turns out that that's fine, then I can accept your answer here as well. Hmmm... – [Thomas Tempelmann](#) May 15 '10 at 8:40

I do not have a USB serial device, but there must be a way to find the real ports using the HAL libraries directly:

```
=====
#!/usr/bin/env bash
#
# Uses HAL to find existing serial hardware
#


for sport in $(hal-find-by-capability --capability serial) ; do
    hal-get-property --udi "${sport}" --key serial.device
done
=====
```

The posted python-dbus code nor this sh script lists the bluetooth /dev/rfcomm* devices, so it is not the best solution.

Note that on other unix platforms, the serial ports are not named ttyS? and even in linux, some serial cards allow you to name the devices. Assuming a pattern in the serial devices names is wrong.

edited Dec 31 '10 at 2:01

answered Dec 31 '10 at 1:49

 **kelk1**
21 2

I have no serial device here to test it, but if you have python and dbus you can try it yourself.


```
import dbus
bus = dbus.SystemBus()
hwmanager = bus.get_object('org.freedesktop.Hal', '/org/freedesktop/Hal/Manager')
hwmanager_i = dbus.Interface(hwmanager, 'org.freedesktop.Hal.Manager')
print hwmanager_i.FindDeviceByCapability("serial")
```

If it fails you can search inside `hwmanager_i.GetAllDevicesWithProperties()` to see if the capability name "serial" that I just guessed has a different name.

HTH

edited Mar 31 '10 at 18:41

answered Mar 27 '10 at 19:42

 **baol**
2,627 14 33

Using `/proc/tty/drivers` only indicates which tty drivers are loaded. If you're looking for a list of the serial ports check out `/dev/serial`, it will have two subdirectories: `by-id` and `by-path`.

EX:

```
# find . -type l
```

9/22/2015

How to find all serial devices (ttyS, ttyUSB, ..) on Linux without opening them? - Stack Overflow

```
./by-path/usb-0:1.1:1.0-port0  
./by-id/usb-Prolific_Technology_Inc._USB-Serial_Controller-if00-port0
```

Thanks to this post: <http://superuser.com/questions/131044/how-do-i-know-which-dev-ttys-is-my-serial-port>

answered Apr 1 at 4:15



blarf

51 2