Linux Serial Console

To use a serial port as console you need to compile the support into your kernel - by default it is not compiled in. For PC style serial ports it’s the config option next to ″Standard/generic (dumb) serial support″. You must compile serial support into the kernel and not as a module.

It is possible to specify multiple devices for console output. You can define a new kernel command line option to select which device(s) to use for console output.

The format of this option is:

```
        console=device,options

        device:         tty0 for the foreground virtual console
                        ttyX for any other virtual console
                        ttySx for a serial port
                        lp0 for the first parallel port
                        ttyUSB0 for the first USB serial device

        options:        depend on the driver. For the serial port this
                        defines the baudrate/parity/bits/flow control of
                        the port, in the format BBBBPNF, where BBBB is the
                        speed, P is parity (n/o/e), N is number of bits,
                        and F is flow control (’r’ for RTS). Default is
                        9600n8. The maximum baudrate is 115200.
```

You can specify multiple console= options on the kernel command line. Output will appear on all of them. The last device will be used when you open /dev/console. So, for example:

```
        console=ttyS1,9600 console=tty0
```

defines that opening /dev/console will get you the current foreground virtual console, and kernel messages will appear on both the VGA console and the 2nd serial port (ttyS1 or COM2) at 9600 baud.

Note that you can only define one console per device type (serial, video).

If no console device is specified, the first device found capable of acting as a system console will be used. At this time, the system first looks for a VGA card and then for a serial port. So if you don’t have a VGA card in your system the first serial port will automatically become the console.

You will need to create a new device to use /dev/console. The official /dev/console is now character device 5,1.

(You can also use a network device as a console. See Documentation/networking/netconsole.txt for information on that.)

Here’s an example that will use /dev/ttyS1 (COM2) as the console. Replace the sample values as needed.

1. Create /dev/console (real console) and /dev/tty0 (master virtual

console):

```
   cd /dev
   rm -f console tty0
   mknod -m 622 console c 5 1
   mknod -m 622 tty0 c 4 0
```

2. LILO can also take input from a serial device. This is a very useful option. To tell LILO to use the serial port: In lilo.conf (global section):

```
   serial  = 1,9600n8 (ttyS1, 9600 bd, no parity, 8 bits)
```

3. Adjust to kernel flags for the new kernel, again in lilo.conf (kernel section)

```
   append = "console=ttyS1,9600"
```

4. Make sure a getty runs on the serial port so that you can login to it once the system is done booting. This is done by adding a line like this to /etc/inittab (exact syntax depends on your getty):

```
   S1:23:respawn:/sbin/getty -L ttyS1 9600 vt100
```

5. Init and /etc/ioctl.save

   Sysvinit remembers its stty settings in a file in /etc, called `/etc/ioctl.save'. REMOVE THIS FILE before using the serial console for the first time, because otherwise init will probably set the baudrate to 38400 (baudrate of the virtual console).

6. /dev/console and X
   Programs that want to do something with the virtual console usually open /dev/console. If you have created the new /dev/console device, and your console is NOT the virtual console some programs will fail. Those are programs that want to access the VT interface, and use /dev/console instead of /dev/tty0. Some of those programs are:

   Xfree86, svgalib, gpm, SVGATextMode

   It should be fixed in modern versions of these programs though.

   Note that if you boot without a console= option (or with console=/dev/tty0), /dev/console is the same as /dev/tty0. In that case everything will still work.

7. Thanks

   Thanks to Geert Uytterhoeven <geert@linux-m68k.org> for porting the patches from 2.1.4x to 2.1.6x for taking care of the integration of these patches into m68k, ppc and alpha.

Miquel van Smoorenburg <miquels@cistron.nl>, 11-Jun-2000