Short guide to Serial (RS-232) communications in Linux

Matti Pastell
matti.pastell@helsinki.fi

November 23, 2007

1 Finding serial ports

Serial ports appear in Linux as device files, which means you can access them conviniently with same command as e.g. text files.

1. Normal serial ports appear as **/dev/ttyS#**, so first is **/dev/ttyS0**, second **/dev/ttyS1** and so on.
2. USB to serial adapters should work out of the box (from kernel 2.4 on I think) without any extra drivers and they appear as **/dev/ttyUSB#** ie. **/dev/ttyUSB0**

2 Serial port settings

Serial port settings can be changed using **stty** command. (See "man stty")
Example: Set serial port /dev/ttyS0 baud rate to 57600 and odd parity

```
stty -F /dev/ttyS0 57600 parodd
```

3 Reading from and writing to serial port

You can send commands to serial port using command **echo**:

```
echo "command" > /dev/ttyUSB0
```

And read data from serial port using **cat** :

```
cat /dev/ttyUSB0
```

redirect output to a file:

```
cat /dev/ttyUSB0 > file.txt
```

4 Parsing data

A good option for parsing data is to use awk or gawk. Awk (gawk is GNU Awk) is a programming language, that is very good in modifying text files and usually doesn't need any loops because it handles all rows in a file by default. You can pipe data from serial port to gawk for parsing. The gawk manual can be found at http://www.gnu.org/software/gawk/manual/gawk.pdf

Gawk example: Print (to terminal) current time in H:M:S format and first three characters from each row of data read from serial port.

```
cat /dev/ttyUSB0 |  gawk '{print strftime("%T"), substr($1 ,1,3)}'
```

5 Running process in the background

You can start backround data logging process (a single command or a shell script) using the command **nohup**. The process runs even if you close the terminal or logout from the computer. Example:

```
nohup cat /dev/ttyS0 |  gawk '{substr($1 ,1,3)}' > result.txt
```

Kill the datalogging process (Note that you will kill all cat processes so you might want not use the command if you are not the only user or if you are logging from several ports and only want to stop one) :

```
killall cat
```

Less brutal alternetive is to find the process ID using "ps ax" and killing the specific process:

```
kill -9 PID
```

6 Plotting data

What if you wan't to plot your data in real time? I use program called **kst** http://kst.kde.org/ that can plot updating data from any text file. Works nicely and allows me to start plotting the data any time and turn off obviously useless plotting while I'm away from the computer. Kst is a kde app, but it works fine with also other WM's and DE's (I use openbox).

7 Terminal software

Use minicom or cutecom as terminal software. (like hyperterminal in Windows)

8 Logging serial data using R

You can also log data directly into R statistical software, at least with slow measurements. This is of course convenient for analysing purposes. The logging is done as follows using the **scan** command:

R example: reading in 6 comma separated numbers from /dev/stty0 into variable data (the command is broken into two lines):

```
data<-scan(file="/dev/ttyS0",n=6,
what=list(c1=(1),c2=(1),c3=(2),c4=(3),c5=(4),c6=(5)), sep=',')
```

9 Why use Linux for datalogging?

You can accomplish simple datalogging apps with just a few lines of text and parsing the data is very simple with gawk. Logging data from terminal takes very little system resources. For Example: I'm currently logging data from serial port at 220hz, parsing the data with gawk in realtime from hex to decimal format, converting the output of three channels of A/D converter to voltage data, adding a time stamp and saving datat to disk. And the whole process takes less than 1% CPU power (Celeron 2,66 Ghz) and 1600K memory! This means that you can also use old hardware, especially if you don't need a GUI at all. You don't have to worry about rebooting the computer logging the data anymore either and remote access is secure and convenient with ssh (even from your mobile if you wish).