Seyon: Quick Start Guide

Peter D. Hiscocks
Syscomp Electronic Design Limited
phiscock@ee.ryerson.ca
www.syscompdesign.com November 9, 2005

1 Overview

The `seyon` program is a useful program for communicating from a host PC to some device connected to a serial port (COM port in Windows-speak). This is a useful arrangement for interfacing a modern PC to some electronic hardware.

However, as this is written, serial ports are becoming obsolete, replaced by USB ports. The programming of a USB interface is extremely complex, and so it is not easily accessible to someone with a minor interest in USB. Fortunately, there are devices and drivers that make a USB connection appear as a connection to a serial port (see `http://www.ftdichip.com/`). Then a hardware device may be accessed as if it were connected to a serial port.

If the hardware device receives and sends commands as ASCII strings, then the terminal emulator may be used to operate the hardware.

This note is provided as a *quick start* guide to using the seyon terminal emulator. The `man` pages for seyon provide further detail.

2 The Terminal Emulator

A *terminal emulation* program is a program that sucks the intelligence out of a personal computer and turns it into a relic of the past: the *dumb terminal*. To appreciate the absence of functions of the classic dumb terminal, we have to journey back to the Jurassic Age of computer technology, the Time of the Mainframe, between perhaps 1960 and 1975. Mainframe computers were large, expensive, and had impressive control panels that only a selected few priests were allowed to operate. The dinosaurs required air conditioning and raised floors (to accomodate cables) and so were kept in separate rooms. Lesser mortals accessed programs via the *glass teletype*, a terminal equipped with a video display and a keyboard, the dumb terminal. IBM had its own standard for these things, and a few of them can still be found in the corridors of high technology enterprises, line cord removed, evidence that yesterdays status symbol is tomorrows computer trash.

The dumb terminal existed at a time when computers and electronic devices were expensive. From an economic point of view, it made sense to have one hideously expensive mainframe and a bunch of low cost terminals to communicate with the mainframe. Each terminal had the absolute minimum of intelligence to detect a key press on the keyboard and spit that out a connector on the back, or to receive a character stream and display that on the video screen. The electronics was 'bubblegum' LSI logic. Later models might have had a UART or primitive microprocessor.

Even though mainframes and dumb terminals are no longer with us, the standard lives on as the lowest common denominator for communication with a computer. Thus, in the situation that concerns us, the USB hardware will communicate with a device much like a dumb terminal. We load a program into our P/C's that *emulates* a dumb terminal. As a bonus, we can still use all the nice P/C features, such as the hard disk for storage and other utilities.

The Windows operating systems provide the `Hyperterminal` terminal emulator. Under Linux, there are two common terminal emulator programs: `minicom` and `seyon`.

3 Installing Seyon

In my experience with RedHat and Suse distributions, neither seyon or minicom are installed automatically. However, the rpm's are on the distribution disks, so they can be installed relatively easily.

Most recently, I installed seyon on an Acer laptop running Suse 9.2. The installation advised me to ensure that any user operating seyon should be part of the `uucp` group.

1. Check that seyon is installed:

   ```
   phiscock@panther:~> which seyon
   /usr/X11R6/bin/seyon
   ```

2. Seyon requires that there be a directory `.seyon` in the home directory of the user.

   ```
   phiscock@panther:~> mkdir .seyon
   ```

   This is a hidden directory, so you must use `ls -a` to see it (and the other hidden directories).

3. The `.seyon` directory must contain a `phonelist`, `protocols` and `startup` files.

4. Here is my version of the phone list. Most of it is comments. The phone numbers (which will appear in a `seyon` dialogue) are at the end.

   ```
   # This is the file that is used as a phone list by Seyon
   # Peter Hiscocks, December 2001
   # The required fields (in order) are:
   # phone number
   # host name
   # These other fields are used to override the defaults.
   # If the string contains spaces it must be quoted, eg:
   #       PREFIX="AT %c1"
   # The optional fields, each with its own keyword, are:
   # Keyword:      Function:                                Example:
   # SCRIPT        Script file to run when connected        SCRIPT=athena
   # BPS           Baud rate                                BAUD=9600
   # PARITY        Parity (0=n, 1=odd, 2=even)              PARITY=0
   # STOPB         Stop Bits (1 or 2)                       STOPB=1
   # PREFIX        Modem dialing prefix                     PREFIX=ATDP
   ```

```
# SUFFIX        This must contain <return>              SUFFIX=^M

# Foonet
527-972-0466 foonet-9600 BPS=9600
527-083-0467 footnet-fast BPS=38400
```

5. Here is my `protocols` file. It defines what commands `seyon` should use for uploads and downloads to a remote system. In this case, the `zmodem` protocol requires the programs `sz` and `rz`. This can be an empty file if no protocols are required.

```
# This file contains the instructions for seyon to actuate uploading.
# Peter Hiscocks, December 2001
# The required fields (in order) are:
# The title: one word or a quoted string
# The command: if it starts with '$', stdin and stdout are redirected to the
#  modem port.
# A single 'n' or 'y' to indicate whether a parameter is required.

# The line shown below supports Zmodem uploads and asks for a file name,
#  which can contain wildcard characters.
"SEND - Zmodem" "$sz -vv" y

# The line shown below may not be necessary for downloads, but this is
#  quoted from the docs:
"RECEIVE - Zmodem" "$rz -vv" n
```

6. Here is my `startup` file. It establishes the initial `seyon` configuration. The configuration can be changed at the seyon control panel.

```
# This is the file that is used to initialize Seyon
# Peter Hiscocks, December 2001
set baud 9600
set bits 8
# Parity is 'none'
set parity 0
set stopBits 1
# Newline translation, if required
# Can be nl, cr, or cr/lf
# Edit and uncomment the following line
# set newlineTranslation cr

# This enables hardware handshaking using rts and cts control signals.
set rtscts on
```

7. (Optional) Create the `.Xdefaults` file in your home directory. This is the location to specify many of the operating characteristics of an XWindows program. In this file, we specify that the timeout should be 60 seconds and the program should display in colour.

```
! Seyon
Seyon*dialTimeOut: 60
Seyon*customization: -color
```

The timeout specification was required to enable logging on to a system that took a very long time to respond with a dial tone. The color specification is not necessary on some systems. So this file may well not be necessary on your system. Check the seyon `man` page for other specifications.

8. Start the seyon program.

```
phiscock@panther:~> seyon -modem /dev/ttyS0&
[2] 8143
```

You must somehow specify which port you would like to connect to, which requires the `-modem /dev/ttyS0` specification for serial port `/dev/ttyS0`. The terminal `&` puts the process into the background. The system indicates that it has a process ID of 8143.

Notice that USB simulated ports are known as `/dev/ttyUSB0`, `/dev/ttyUSB1` and so on.

You cannot change serial ports from within `seyon`, you have to restart the program.

At this point, you should have both a control window `Seyon Command Center` and terminal window `Seyon Terminal Emulator`. The terminal emulator is an `xterm`, so it can be manipulated with `xterm` commands.

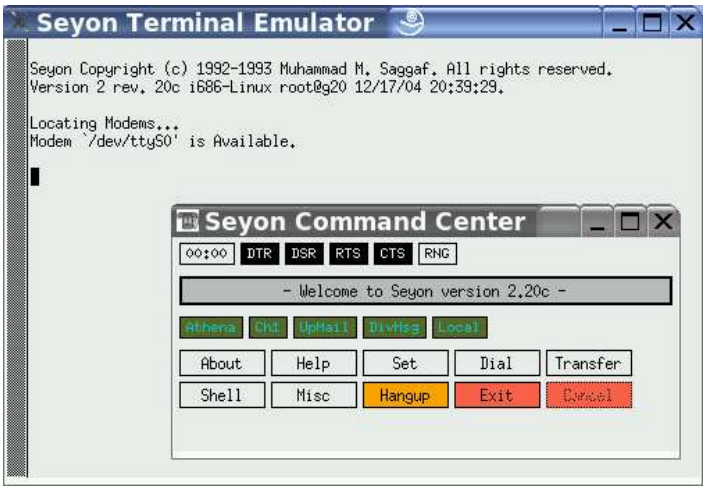

Figure 1: Seyon Windows

You are now connected to the desired serial port. Typing in commands should send them to your serial port hardware, and messages from that hardware should show up on the terminal screen. Note particularly the `Misc->Divert` command which allows one to send a text file to the serial port. This is useful in uploading a program (of S-Records, say) into a microprocessor system attached to the serial port.

4 References

Using Minicom and Seyon
Chapter 11 of *Learning Debian GNU/Linux*
Bill McCarty
O'Reilly Books, 1999
`http://www.oreilly.com/catalog/debian/chapter/book/ch11_07.html`

Debugging the Serial Port
Peter Hiscocks, 2003
`http://www.ee.ryerson.ca/~phiscock/`