PANDORA PRODUCTS

Pandora Products Manual
nulmodem TCP/IP Link

Jim Schimpf

Document Number: PAM-200608001
Revision Number: 0.3
16 March 2016

Pandora Products.
215 Uschak Road
Derry, PA 15627

Pandora Products.

215 Uschak Road

Derry, PA 15627

Phone: 412.829.8145

Fax: 724.539.1276

Web: pandora.dyn-o-saur.com:8080/~jim/

Email: vze35xda@verizon.net

Document Revision History

Version	Author	Description	Date
0.1	js	Initial Version	21-Aug-2006
0.2	js	Editing update	24-Oct-2006
0.3	js	Minor editing	16-Mar-2016

Contents

1 Introduction

nulmodem is designed to act like the hardware NUL modem used in RS-232 work to link two connectors of the same sex together. When you have two RS-232 ports that are both say male you have to have a female-female adapter but in additon you usually have to swap pins 2 and 3 so the transmit of one unit is connected to the receive of the other. **nulmodem** is designed to do a similar function for TCP/IP connections.

For TCP/IP connections you have some programs like **telnet** where they link out to an address. If you want to have two telnet programs talk to each other you cannot do it directly. There must be something between them that would accept connections from each program than cross connect the transmit data from one side to the receive of the other.

nulmodem has a number of modes where it can do the following:

- Accept connection on both ports.
- Make an outbound connection on one port and accept connections on the other.
- Make outbound connections from both ports.
- Accept a connection on a single port and echo received data back to the sender.

In additon **nulmodem** can be scripted using a Lua5[**?**, **?**] script. The Lua script will have functions with defined names (see 4) and these functions will be called by events in **nulmodem**.

2 Syntax

```
nulmodem [-p0 <port #>][-ip0 <IP address>]
         [-p1 <port #>][-ip1 <IP address>]
         [-c <Lua Script>]
         [-loop][-log]
         [-v]
```

Flag	Field	Use
-p0	TCP Port	Port used by channel 0
-ip0	TCP Address	Addess used by channel 0
-p1	TCP Port	Port used by channel 1
-ip1	TCP Address	Address used by channel 1
-loop	NONE	Enables loop back
-log	NONE	Enable logging
-v	NONE	Show syntax - don't run

The program has two channels (0 and 1) and the command line options specify how each of these channels will connect. If only the port is specified for a channel then it is a server and will wait for a connection to it. If both port and IP address are given for a channel it will attempt to connect to this address.

2.1 Option Details

2.1.1 -p0 <TCP Port>

This sets the port used by channel 0 and can be any number between 1 and 65535. If given alone without an -ip0 flag then **nulmodem** will wait for a connection on this port. In this case use of ports below 1024 will require admin priviledges.

DEFAULT = NONE

2.1.2 -ip0 <IP Address>

This is the IP address used for channel 0. If given along with a port value (-p0) then **nulmodem** will attempt to connect to this address (the program will shutdown if it cannot).

DEFAULT = NONE

2.1.3 -p1 <TCP Port>

This sets the port used by channel 1 and can be any number between 1 and 65535. If given alone without an -ip0 flag then **nulmodem** will wait for a connection on this port. In this case use of ports below 1024 will require admin priviledges.

DEFAULT = NONE

2.1.4 -ip1 <IP Address>

This is the IP address used for channel 1. If given along with a port value (-p1) then **nulmodem** will attempt to connect to this address (the program will shutdown if it cannot).

DEFAULT = NONE

2.1.5 -loop

When present this will tell **nulmodem** to loop the transmit data from channel 0 back to the receive data. Thus you only need to set up channel 0 and any data input will be echoed back to the output. Useful for testing applications.

DEFAULT = NO LOOP

2.1.6 -log

When present will tell **nulmodem** to put all data from either channel to stdout. Useful to monitor data flow

DEFAULT = NO LOG

2.1.7 -c <Lua script>

The name following this is a Lua script that can modify the behavior of **nulmodem**. See 4

2.1.8 -v

When present **nulmodem** will print the following and NOT run

```
nulmodem TCP/IP <=> TCP/IP Connect [Aug 21 2006 14:16:28]
nulmodem - TCP/IP 0 <-> TCP/IP 1 Version Aug 21 2006 14:16:28
SYNTAX: nulmodem[-p0 <TCP/IP Port 0>]
               [-tcp0 <TCP/IP Port 0>]
               [-p1 <TCP/IP Port 1>]
               [-tcp1 <TCP/IP Port 1>]
               [-c <Lua connect Script>]
               [-loop]
               [-log]
               [-v]
    -p0    <TCP/IP Port 0> DEFAULT = NONE
    -tcp0  <TCP/IP Port 0> DEFAULT = NONE (connect)
    -p1    <TCP/IP Port 1> DEFAULT = NONE
    -tcp1  <TCP/IP Port 1> DEFAULT = NONE (connect)
    -c     <Lua script>    DEFAULT = NONE
    -loop  Loop back on 0  DEFAULT = OFF
    -log   Log to stdout   DEFAULT = OFF
    -c <Lua Script>        DEFAULT = NONE
    -v     Show this message and return
```

3 nulmodem Use

3.1 Introduction

nulmodem is used to connect to TCP/IP connections together. For example if you wish to connect two telnet sessions together so that what is typed one appears on the other. This is not possible directly since telnet is a client and connects out. But if you set up **nulmodem** to be a server on both sides then you just connect to both sides with the two telnets. Thus

```
nulmodem -po 7000 -p1 7010
```

Now the two telnet sessions would:

```
telnet localhost 7000   -- To connect in on channel 0
```

the second session would be:

```
    telnet localhost 7010   -- To connect in on channel 1
```

3.2 Use cases

3.2.1 Connecting two clients

This would be similar to connecting the two telnet sessions shown in the introduction

3.2.2 Connecting two servers

In this case you would set both channel 0 and channel 1 to connect to each server then either they would talk to each other or you would use a Lua script to control communication.

```
    nulmodem -ip <server 0> -p0 7000 -ip1 <server 1> -p1 7010
```

3.2.3 Connecting a server and client

Technically this isn't necessary since they could connect themselves but if you want to use logging or a lua script it might be useful. So if the client is on channel 0 and the server is on channel 1 you get:

```
    nulmodem -p0 7000 -ip1 <server 1> -p1 7010
```

4 Lua Script Use

4.1 Scripting

Information on the Lua language can be found here: Lua language site www.Lua.org and this book [**?**].

4.2 Lua Event Functions

The state leading to the function call will be detailed below, see 5 for further information on what functions are available.

4.2.1 ON_START()

This function is called as **nulmodem** is starting but before it has begun monitoring for TCP/IP connection.

4.2.2 ON_CONN()

This function is called when a new TCP/IP connection is accepted but before **nulmodem** enters into the terminal server loop, or the connected state.

4.2.3 ON_DROP()

This function is called after the TCP/IP connection is dropped and before **nulmodem** enters the idle loop.

4.2.4 ON_PERIODIC()

This function is called every second **nulmodem** is running (in both idle and connected states)

5 Lua Extensions

There are libraries added to this embedded Lua to supply the specialized functions for **nulmodem** operation. There are three libraries added, timer, serial and net.

5.1 timer - Library

5.1.1 Open a timer timer.topen()

This function is used to open a millisecond timer for delay loops or other timing operations. There can be any number of these timers created during a run of **nulmodem**.

`t = timer.topen()`

INPUT	NAME	USE
	-	
OUTPUT	t	Handle to created timer struct or nil if failure

5.1.2 Close a timer timer.tclose(t)

This function is used to free the memory used by a timer after use.

`timer.tclose(t)`

INPUT	NAME	USE
	t	This is the handle of a timer, the memory will be freed
OUTPUT	NONE	

5.1.3 Restart a timer timer.tstart(t)

This function is used to zero a timer. For example after a delay has elapsed then start can be called and the delay re-run.

```
timer.tstart(t)
```

INPUT	**NAME**	**USE**
	t	This timer will be re-started after this call
OUTPUT	NONE	

5.1.4 Check a timer for done timer.tdone(t,delay)

This function will return 1 when the timer passes the delay time. Up to that time it will return nil.

```
done = timer.tdone(t,delay)
```

INPUT	**NAME**	**USE**
	t	This is the handle of a timer
	delay	Delay time in ms
OUTPUT	done	nil till the timer passes the delay time and 1 after

5.2 net - Library

This libary is used to access the open TCP/IP port of **nulmodem**. It will be used to send or recieve data. In addition it can monitor the connection status.

5.2.1 Write data to TCP/IP port net.swrite[0/1](data)

This function will send data to the open TCP/IP port.

```
count = net.swrite0(data) -- for channel 0
```

INPUT	**NAME**	**USE**
	data	This string of data will be sent to the TCP/IP port
OUTPUT	count	# of bytes sent

5.2.2 Read data from TCP/IP port net.sread[0/1](len)

This function will read data from the TCP/IP port. It will read up to len bytes and returns.

```
data = net.sread0(len)  -- For channel 0
```

INPUT	**NAME**	**USE**
	len	# Bytes of data to read
OUTPUT	data	Data as read from the port

5.2.3 Control TCP/IP connection net.drop()

When called this will drop the TCP/IP connection

`net.drop()`

INPUT	**NAME**	**USE**
	-	
OUTPUT	NONE	

5.2.4 Read TCP/IP connection status net.cstat()

This function will return the status of the TCP/IP connection, 1 if connected and nil if idle

`status = net.cstat()`

INPUT	**NAME**	**USE**
	-	
OUTPUT	status	1 if connection up, nil if idle

6 Example Script

This is a simple example script that uses all the **nulmodem** Lua functions. It will show a message on start and put out periodic messages while a client is connected to the port.

```
--------------------------------------------------------------------
--
-- function ON_START()    - Run at nulmodem startup
--
-- INPUT:   NONE
--
-- OUTPUT:  NONE
--
---------------------------------------------------------------------
function ON_START()
   local out
   print()
   print("------------------------------------")
   out = string.format("**** nulmodem SERVER UP *****")
   print(out)
   out = string.format("[%s]\n[%s]",
                        "Lua 5.0.2",
            "R. Ierusalimschy, L. H. de Figueiredo & W. Celes")
   print(out)
   print("------------------------------------")
   print()
end
--------------------------------------------------------------------
--
-- function ON_CONN()    - Run at network connection startup
--
-- INPUT:   NONE
```

```
--
-- OUTPUT:  NONE
--          This will put out message to Channel 0 & 1
--
---------------------------------------------------------------------
function ON_CONN()
   local out
   local t
   local count
   out = string.format("Connection UP\n")
   print(out)
   net.swrite0(out)
   net.swrite1(out)
end
--------------------------------------------------------------------
--
-- function ON_DROP()    - Run at network drop
--
-- INPUT:   NONE
--
-- OUTPUT:  NONE
--
--
---------------------------------------------------------------------
function ON_DROP()
   local out
   out = string.format("Connection DOWN")
   print(out)
end
-- These globals used by ON_PERIODIC
count = 0
nettime = 0
--------------------------------------------------------------------
--
-- function ON_PERIODIC()    - Run every second
--
-- INPUT:   NONE
--
-- OUTPUT:  NONE
--          Print connection status and time
--
---------------------------------------------------------------------
function ON_PERIODIC()
   local out
   count = count + 1
   if( count > 5 )
   then
       if( net.cstat() )
       then
           out = string.format("Periodic Message [NET UP %d sec]",
           nettime = nettime + 5
       else
           out = string.format("Periodic Message")
           nettime = 0
       end

       print(out)
       count = 0
   end
end
```

Index

References