

**Package TOOLS - Additional Tools For  
Debugging  
Version 4.0.0-stable-x86\_64-r60603**

Frank Meyer  
email: [frank@fli41.de](mailto:frank@fli41.de)

The fli41-Team  
email: [team@fli41.de](mailto:team@fli41.de)

March 7, 2022

# Contents

<b>1</b>	<b>Documentation Of Package TOOLS</b>	<b>3</b>
1.1	TOOLS - Additional Tools For Debugging . . . . .	3
1.1.1	Networking-Tools . . . . .	3
1.1.2	Hardware Identification . . . . .	8
1.1.3	File Management Tools . . . . .	10
1.1.4	Developer-Tools . . . . .	10
	<b>List of Figures</b>	<b>12</b>
	<b>List of Tables</b>	<b>13</b>
	<b>Index</b>	<b>14</b>

# 1 Documentation Of Package TOOLS

## 1.1 TOOLS - Additional Tools For Debugging

Package TOOLS provides some Unix programs mostly used for administration and debugging. Other ones like wget are used i.e. for catching the first (ad-)page of some providers. By setting the value 'yes' the mentioned program is copied to the fli4l router. Default setting is 'no'. The programs are only described in short, how to use them is described in their respective man pages which can be found in your favourite linux distribution or online at: <http://www.linuxmanpages.com>

### 1.1.1 Networking-Tools

#### **OPT\_BMON** Bandwidth Monitor

bmon is a monitoring and debugging tool to capture networking related statistics and prepare them visually in a human friendly way. It features various output methods including an interactive curses user interface and a programmable text output for scripting.

#### **OPT\_CURL** Data Transfer Tool

The program curl allows the transfer of data to or from a server running a number of supported protocols. These include, amongst others, FTP, HTTP(S), SCP, SFTP, and TFTP.

The program also provides support for user authentication, data transfer via proxy, FTP upload, HTTP POST requests, SSL connections, cookies, resumption of interrupted file transfers and more.

**Important:** *In order to establish TLS connections, the Mozilla X.509 root certificates should be installed by setting `CERT_X509_MOZILLA='yes'` in the CERT package.*

#### **OPT\_DIG** DNS Multitool

The command dig allows to execute various DNS queries.

#### **OPT\_FTP** FTP-Client

The ftp program can connect fli4l to a FTP server to move files between the two of them.

**FTP\_PF\_ENABLE\_ACTIVE** The setting `FTP_PF_ENABLE_ACTIVE='yes'` adds a rule to the packet filter that enables initiated active FTP. For `FTP_PF_ENABLE_ACTIVE='no'` such a rule has to be added to the `PF_OUTPUT_%-array` manually (if needed), for an example look here (Page ??).

Passive FTP is always possible, neither this variable nor an explicit packet filter rule is needed then.

### **OPT\_IFTOP** Netzwerküberwachung

The iftop program lists all active network connections and their throughput directly on the fli4l router.

After login to the router iftop is simply started by executing it on the console.

### **OPT\_IMONC** Text based control program for imond

This program is a text based frontend for the router to control imond.

### **OPT\_IPERF** Performance checks for the network

The command iperf can do performance measuring for networks. The program has to be started on the two machines involved. On the iperf server execute

```
fli4l-server 4.0.0-stable-x86_64-r60603~# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
```

to start. The server will wait for a connection from an iperf client then. Start the client on the second machine with

```
fli4l-client 4.0.0-stable-x86_64-r60603~# iperf -c 1.2.3.4
-----
Client connecting to 1.2.3.4, TCP port 5001
TCP window size: 16.0 KByte (default)
-----
[ 3] local 1.2.3.5 port 50311 connected with 1.2.3.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec   985 MBytes  826 Mbits/sec
```

A performance check will start immediately showing first results. Some options can be added to iperf, for details evaluate the informations on its homepage <http://iperf.sourceforge.net/>.

### **OPT\_LNSTAT** Advanced network informations

This OPT installs lnstat, a tool for processing informations from /proc/net/stat. The name stands for “Linux Network Statistics”.

Examples from lnstat’s manual:

```
# lnstat -d
    Get a list of supported statistics files.
# lnstat -k arp_cache:entries,rt_cache:in_hit,arp_cache:destroys
    Select the specified files and keys.
# lnstat -i 10
    Use an interval of 10 seconds.
# lnstat -f ip_contrack
    Use only the specified file for statistics.
# lnstat -s 0
```

```
Do not print a header at all.
# linstat -s 20
    Print a header at start and every 20 lines.
# linstat -c -1 -i 1 -f rt_cache -k entries,in_hit,in_slow_tot
    Display statistics for keys entries, in_hit and in_slow_tot of field
    rt_cache every second.
```

**OPT\_NETCAT** Transfer data to TCP based servers

**OPT\_NETIO** Network performance test

Just like “iperf” the program “netio” can do network performance tests. The program has to be started on both systems taking part in the test, one acting as the server, one as the client side. On the server side the program has to be started via `netio -s -t` (for data transfer via TCP) resp. `netio -s -u` (for data transfer via UDP). Via `netio <server> -t` resp. `netio <server> -u` the “netio” client process is started on the client side, contacting the server side and starting the performance measurement.

**OPT\_NGREP** A grep that is able to work directly with network devices.

**OPT\_NMAP** Port Scanner

By the help of nmap operating systems can be tested for open ports. In addition the program provides further information, e.g. about MAC addresses or the operating system used.

**OPT\_NTTCP** Network checks

The program NTTCP can check network speed. On one side a server is started and on the other side the client.

Start the server by executing `nttcp -i -v`. The server will wait for client requests. To test i.e. speed execute `nttcp -t <IP Adress of the server>` on the client.

This is how a started nttcp server looks like:

```
fli4l-server 4.0.0-stable-x86_64-r60603~# nttcp -i -v
nttcp-l: nttcp, version 1.47
nttcp-l: running in inetd mode on port 5037 - ignoring options beside -v and -p
```

This is how a test with a nttcp client looks like:

```
fli4l-client 4.0.0-stable-x86_64-r60603~# nttcp -t 192.168.77.77
1~~8388608~~~~4.77~~~~0.06~~~~~14.0713~~~1118.4811~~~~2048~~~~429.42~~~34133.3
1~~8388608~~~~4.81~~~~0.28~~~~~13.9417~~~239.6745~~~~6971~~~1448.21~~~24896.4
```

The nttcp help shows all further parameters:

```
Usage: nttcp [local options] host [remote options]
local/remote options are:
    -t          transmit data (default for local side)
```

```

-r      receive data
-l#     length of bufs written to network (default 4k)
-m      use IP/multicasting for transmit (enforces -t -u)
-n#     number of source bufs written to network (default 2048)
-u      use UDP instead of TCP
-g#us   gap in micro seconds between UDP packets (default 0s)
-d      set SO_DEBUG in sockopt
-D      don't buffer TCP writes (sets TCP_NODELAY socket option)
-w#     set the send buffer space to #kilobytes, which is
        dependent on the system - default is 16k
-T      print title line (default no)
-f      give own format of what and how to print
-c      compares each received buffer with expected value
-s      force stream pattern for UDP transmission
-S      give another initialisation for pattern generator
-p#     specify another service port
-i      behave as if started via inetd
-R#     calculate the getpid()/s rate from # getpid() calls
-v      more verbose output
-V      print version number and exit
-?      print this help
-N      remote number (internal use only)
default format is: %9b%8.2rt%8.2ct%12.4rbr%12.4cbr%8c%10.2rcr%10.1ccr

```

**OPT\_RTMON** Installs a tool that will track changes in routing tables. Primary used for debugging.

**OPT\_SOCAT** The program “socat” is more or less an enhanced version of the “[netcat program](#)” (Page 5) with more functionality. By using “socat” you may not only establish or accept various types of network connections, but also sent data to or read data from UNIX sockets, devices, FIFOs, and so on. In addition sources and destinations of *different* types may be connected: An example would be a Network server listening on a TCP port and then writing received data to a local FIFO or reading data from the FIFO and then transmitting it over the network to a client. See <http://www.dest-unreach.org/socat/doc/socat.html> for more information and application examples.

**OPT\_TCPDUMP** debug

The program `tcpdump` can watch, interpret and record network traffic. Find more on this by feeding Google with “tcpdump man”

`tcpdump <parameter>`

**OPT\_TRACEPATH** Determining the PMTU

The program `tracert` is used to determine the so-called “Path MTU”. This is defined as the maximum usable packet size for the path used from the fl4l router to the destination host. Larger packets must either be fragmented (IPv4) or discarded (IPv6). Typically,

the Linux kernel cares for the determination of the correct Path MTU (“ Path MTU Discovery ”). Occasionally, however, it is useful to find out this Path MTU to diagnose problems in the network.

An example for IPv4:

```
sandbox 4.0.0-r46077M # tracepath -4 fli4l.de
1?: [LOCALHOST] pmtu 1500
1: fritz.box 0.703ms
1: fritz.box 0.588ms
2: a89-182-53-190.net-htp.de 0.702ms pmtu 1492
2: a81-14-248-243.net-htp.de 33.692ms
3: a81-14-249-82.net-htp.de 32.089ms asymm 4
4: xe-4-1-2.edge4.Berlin1.Level3.net 35.936ms
5: SYSELEVEN-G.edge4.Berlin1.Level3.net 74.944ms asymm 8
6: ecix.dus.octalus.in-berlin.de 49.693ms asymm 7
7: virtualhost.in-berlin.de 50.269ms reached
Resume: pmtu 1492 hops 7 back 57
```

In this example the Internet connection is established via a DSL connection between an AVM Fritz! Box and a BRAS belonging to the Internet provider htp. As for DSL in Germany PPP over Ethernet (PPPoE) is used, the Path MTU is only 1492 bytes in size.

An example for IPv6:

```
sandbox 4.0.0-r46077M # tracepath -6 fli4l.de
1?: [LOCALHOST] 0.046ms pmtu 1280
1: gw-1362.ham-01.de.sixxs.net 43.586ms
1: gw-1362.ham-01.de.sixxs.net 42.832ms
2: 2001:6f8:862:1::c2e9:c729 43.565ms asymm 1
3: 2001:6f8:862:1::c2e9:c72c 44.313ms asymm 2
4: 30gigabitethernet4-3.core1.fra1.he.net 64.501ms asymm 6
5: no reply
6: virtualhost.in-berlin.de 65.949ms reached
Resume: pmtu 1280 hops 6 back 56
```

Here, the Internet connection is established via a 6in4 tunnel by SixXS. In the configuration of the tunnel a tunnel MTU of 1280 is fixed. Since this is the smallest possible MTU for IPv6 connections, it is not further reduced on the path to the destination host.

## **OPT\_DHCPDUMP** DHCP packet dumper

The program “dhcpcdump” can be used to analyse DHCP packets more accurately. It is based on tcpdump and produces easily readable output.

Usage:

```
dhcpcdump -i interface [-h regular-expression]
```

The program can be executed using the following command:

```
dhcpcdump -i eth0
```

If desired, the analysis can be cut down to a specific MAC address using regular expressions. The command would be:

```
dhcpcdump -i eth0 -h ^00:a1:c4
```

The output could look like this:

```
TIME: 15:45:02.084272
  IP: 0.0.0.0.68 (0:c0:4f:82:ac:7f) > 255.255.255.255.67 (ff:ff:ff:ff:ff:ff)
  OP: 1 (BOOTPREQUEST)
HTYPE: 1 (Ethernet)
HLEN: 6
HOPS: 0
  XID: 28f61b03
SECS: 0
FLAGS: 0
CIADDR: 0.0.0.0
YIADDR: 0.0.0.0
SIADDR: 0.0.0.0
GIADDR: 0.0.0.0
CHADDR: 00:c0:4f:82:ac:7f:00:00:00:00:00:00:00:00:00:00
  SNAME: .
  FNAME: .
OPTION: 53 ( 1) DHCP message type          3 (DHCPREQUEST)
OPTION: 54 ( 4) Server identifier          130.139.64.101
OPTION: 50 ( 4) Request IP address         130.139.64.143
OPTION: 55 ( 7) Parameter Request List     1 (Subnet mask)
                                           3 (Routers)
                                           58 (T1)
                                           59 (T2)
```

## OPT\_WGET http/ftp Client

The program wget can fetch data from web servers in batch mode. More useful is (and that is why wget is included here) that it can catch redirections to your provider's own webserver while establishing a connection to the internet in a simple way i.e. for Freenet. A (german) Mini-HowTo on this topic can be found here:

<http://www.fli4l.de/hilfe/howtos/einsteiger/wget-und-freenet/>

**Important:** *In order to establish TLS connections, the Mozilla X.509 root certificates should be installed by setting `CERT_X509_MOZILLA='yes'` in the `CERT` package.*

### 1.1.2 Hardware Identification

Often you do not know exactly what hardware is in your own computer or which driver you should use for i.e. your network card or the USB chipset. The hardware itself can help here. It provides a list of devices in the computer and the associated driver if possible. You can choose



whether the recognition is done at boot (which is recommended before the first installation) or later when the computer is running, comfortably triggered from the Web interface. The output might look like this:

```
fli4l 4.0.0-stable-x86_64-r60603 # cat /bootmsg.txt
#
# PCI Devices and drivers
#
Host bridge: Advanced Micro Devices [AMD] CS5536 [Geode companion] Host Bridge (rev 33)
Driver: 'unknown'
Entertainment encryption device: Advanced Micro Devices [AMD] Geode LX AES Security Block
Driver: 'geode_rng'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: Atheros Communications, Inc. AR5413 802.11abg NIC (rev 01)
Driver: 'unknown'
ISA bridge: Advanced Micro Devices [AMD] CS5536 [Geode companion] ISA (rev 03)
Driver: 'unknown'
IDE interface: Advanced Micro Devices [AMD] CS5536 [Geode companion] IDE (rev 01)
Driver: 'amd74xx'
USB Controller: Advanced Micro Devices [AMD] CS5536 [Geode companion] OHC (rev 02)
Driver: 'ohci_hcd'
USB Controller: Advanced Micro Devices [AMD] CS5536 [Geode companion] EHC (rev 02)
Driver: 'ehci_hcd'
```

In essential 3 network cards are in this machine, driven by 'via\_rhine' and an Atheros-WiFi card driven by madwifi (the driver name is not resolved correctly here).

**OPT\_HW\_DETECT** This variable triggers that the files needed for hardware identification are copied to the router. Results can either be found on the console while booting if **HW\_DETECT\_AT\_BOOTTIME** is set to 'yes' or in the web interface if **OPT\_HTTPD** (Page ??) was set to 'yes'. The content of '/bootmsg.txt' can of course be checked in the web interface as well if you have a working network connection to the router.

**HW\_DETECT\_AT\_BOOTTIME** Starts hardware identification start on boot. Recognition will take place in background (it will take some time) and then writes its output to console and '/bootmsg.txt'.

**OPT\_LSPCI** Listing of all PCI devices

**OPT\_I2CTOOLS** Tools for I<sup>2</sup>C access.

**OPT\_IWLEEPROM** Tool to access the EEPROM of Intel and Atheros WLAN cards.

Needed e.g. to reprogram the regulatory domain on ath9k cards (see <http://blog.asiantuntijakaveri.fi/2014/08/one-of-my-atheros-ar9280-minipcie-cards.html>).

**OPT\_ATH\_INFO** Hardware diagnosis tool for WLAN cards based on Atheros chipset.

This tool can extract detailed information about the hardware used in Atheros wireless cards, e.g. ath5k. These include, amongst others, the chipset used or calibration information.

#### **OPT\_FLASHROM** Tool to Flash Chipsets.

This tool may provide a new BIOS or a new Firmware for example for PC Engines mainboards. Details can be found at <http://www.flashrom.org>.

### **1.1.3 File Management Tools**

#### **OPT\_E3** An editor for fli4l

E3 is a very small editor written in Assembler. It mimics various editor modes known from other („full size”) editors. To choose a mode e3 only has to be started with the right command. A short key overview is given by e3 starting without parameter or by pressing Alt+H (except in VI-mode, press „h” in CMD mode then). Caret (^) stands for the Ctrl-/Strg key.

Command	Mode
e3 / e3ws	WordStar, JOE
e3vi	VI, VIM
e3em	Emacs
e3pi	Pico
e3ne	NEdit

#### **OPT\_MTOOLS** mtools provide some DOS-like commands for simpler handling of DOS media (copying, formatting, a.s.o.).

Exact syntax of the commands can be found in the mtools documentation:

<http://www.gnu.org/software/mtools/manual/mtools.html>

#### **OPT\_SHRED** Installs the program *shred* on the router which is used for secure erasing of block devices.

#### **OPT\_YTREE** File Manager

Installs the File Manager Ytree on the router.

### **1.1.4 Developer-Tools**

#### **OPT\_OPENSSL** The tool openssl can i.e. be used to test crypto accelerator devices.

```
openssl speed -evp des -elapsed
openssl speed -evp des3 -elapsed
openssl speed -evp aes128 -elapsed
```

#### **OPT\_STRACE** debug

By using the program strace you can trace system calls and signals or watch function calls and runtime behavior of a program.

```
strace <program>
```

**OPT\_REAVER** Brute force attacks on Wifi WPS PINs

Tests all possible WPS PINs to find the WPA password. For details on usage see:  
<http://code.google.com/p/reaver-wps/>.

**OPT\_VALGRIND** Installs valgrind on the router.

## List of Figures

# List of Tables

# Index

FTP\_PF\_ENABLE\_ACTIVE, [3](#)

HW\_DETECT\_AT\_BOOTTIME, [9](#)

OPT\_ATH\_INFO, [9](#)

OPT\_BMON, [3](#)

OPT\_CURL, [3](#)

OPT\_DHCPDUMP, [7](#)

OPT\_DIG, [3](#)

OPT\_E3, [10](#)

OPT\_FLASHROM, [10](#)

OPT\_FTP, [3](#)

OPT\_HW\_DETECT, [9](#)

OPT\_I2CTOOLS, [9](#)

OPT\_IFTOP, [3](#)

OPT\_IMONC, [4](#)

OPT\_IPERF, [4](#)

OPT\_IWLEEPROM, [9](#)

OPT\_LNSTAT, [4](#)

OPT\_LSPCI, [9](#)

OPT\_MTOOLS, [10](#)

OPT\_NETCAT, [5](#)

OPT\_NETIO, [5](#)

OPT\_NGREP, [5](#)

OPT\_NMAP, [5](#)

OPT\_NTTCP, [5](#)

OPT\_OPENSSL, [10](#)

OPT\_REAVER, [10](#)

OPT\_RTMON, [6](#)

OPT\_SHRED, [10](#)

OPT\_SOCAT, [6](#)

OPT\_STRACE, [10](#)

OPT\_TCPDUMP, [6](#)

OPT\_TRACEPATH, [6](#)

OPT\_VALGRIND, [11](#)

OPT\_WGET, [8](#)

OPT\_YTREE, [10](#)