

**Paket TOOLS - Zusätzliche Werkzeuge zum  
Debugging  
Version 4.0.0-testing-x86\_64-r60699**

Frank Meyer  
E-Mail: [frank@fli4l.de](mailto:frank@fli4l.de)

Das fli4l-Team  
E-Mail: [team@fli4l.de](mailto:team@fli4l.de)

12. August 2022

# Inhaltsverzeichnis

<b>1</b>	<b>Dokumentation des Paketes TOOLS</b>	<b>3</b>
1.1	TOOLS - Zusätzliche Werkzeuge zum Debugging . . . . .	3
1.1.1	Netzwerk-Tools . . . . .	3
1.1.2	Die Hardware-Erkennung . . . . .	9
1.1.3	Dateien-Tools . . . . .	10
1.1.4	Entwickler-Tools . . . . .	11
	<b>Abbildungsverzeichnis</b>	<b>12</b>
	<b>Tabellenverzeichnis</b>	<b>13</b>
	<b>Index</b>	<b>14</b>

# 1 Dokumentation des Paketes TOOLS

## 1.1 TOOLS - Zusätzliche Werkzeuge zum Debugging

Das Paket TOOLS liefert eine Reihe von Unix Programmen, die zumeist für Administrations- und Debugzwecke gedacht sind. Andere Programme wie wget werden z.B. dafür verwendet, die erste (Werbe-)Seite einiger Provider abzufangen. Mit dem Wert 'yes' wird das jeweilige Programm auf den fli4l-Router kopiert. Die Standardeinstellung ist 'no'. Die Programme werden nur kurz vorgestellt, wie sie zu bedienen sind, entnehmen man bitte den man Pages einer beliebigen Unix/ Linux Distribution oder online unter: <http://www.linuxmanpages.com>

### 1.1.1 Netzwerk-Tools

#### **OPT\_BMON** Bandbreitenmonitor

Das Programm *bmon* ist ein Monitor- und Debugging-Werkzeug um netzwerkbezogene Statistiken zu erheben und diese in einer nutzerfreundlichen Weise visuell aufbereitet anzuzeigen. Es werden verschiedene Ausgabemethoden unterstützt, unter anderem auch eine auf curses basierende interaktive Schnittstelle und eine programmierbare Ausgabe für Skripte.

#### **OPT\_CURL** Datenübertragungswerkzeug

Das Programm *curl* erlaubt den Datentransfer von oder zu einem Server mit einer Reihe von unterstützten Protokollen. Dazu gehören u.a. FTP, HTTP(S), SCP, SFTP und TFTP.

Außerdem bietet das Programm Unterstützung für Benutzer-Authentifizierung, Datentransfer via Proxy, FTP-Upload, HTTP-POST-Anfragen, SSL-Verbindungen, Cookies, Wiederaufnahme abgebrochener Dateiübertragungen und mehr.

**Wichtig:** Um mit *curl* TLS-Verbindungen aufbauen zu können, sollten im Paket *CERT* via *CERT\_X509\_MOZILLA='yes'* die Mozilla-X.509-Wurzelzertifikate installiert werden.

#### **OPT\_DIG** Schweizer Taschenmesser fürs DNS

Der Befehl *dig* erlaubt es, vielfältige DNS-Abfragen durchzuführen.

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

Mit dem Programm *ftp* können eine FTP-Verbindung zu einem FTP-Server aufgebaut und Dateien zwischen Router und FTP-Server übertragen werden.

**FTP\_PF\_ENABLE\_ACTIVE** Die Einstellung *FTP\_PF\_ENABLE\_ACTIVE='yes'* fügt dem Paketfilter eine Regel hinzu, die auf dem Router initiiertes aktives FTP ermöglicht. Bei *FTP\_PF\_ENABLE\_ACTIVE='no'* muss eine solche Regel (falls gewünscht) manuell zum

PF\_OUTPUT\_%-Array hinzugefügt werden, ein Beispiel ist in diesem Abschnitt (Seite ??) zu finden.

Passives FTP ist immer möglich, hierfür ist weder diese Variable noch eine explizite Paketfilter-Regel notwendig.

#### **OPT\_IFTOP** Netzwerküberwachung

Mit dem Programm iftop wird eine Auflistung aller aktiven Netzwerkverbindungen und deren Durchsatz direkt auf dem fli4l angezeigt.

Das Programm iftop wird nach dem Anmelden auf dem fli4l-Router durch Eingabe von iftop gestartet.

#### **OPT\_IMONC** Textorientiertes Steuerprogramm für imond

Dieses Programm liefert ein textorientiertes Frontend für den Router, um den imond zu steuern.

#### **OPT\_IPERF** Performancemessung im Netzwerk

Mit dem Programm iperf kann eine Performancemessung des Netzwerks durchgeführt werden. Dazu wird das Programm auf den beiden beteiligten Testsystemen gestartet. Auf dem Server wird das Programm mit

```
fli4l-server 4.0.0-testing-x86_64-r60699~# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
```

gestartet. Der Server wartet dann auf eine Verbindung vom Client. Der Client wird durch

```
fli4l-client 4.0.0-testing-x86_64-r60699~# 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
```

gestartet. Sofort startet die Performancemessung und zeigt die ersten Ergebnisse an. iperf kennt noch eine Reihe weiterer Optionen, für Details schauen Sie sich bitte die Informationen auf der Homepage <http://iperf.sourceforge.net/> an.

#### **OPT\_LNSTAT** Erweiterte Netzwerk-Informationen

Dieses OPT installiert `lnstat`, ein Werkzeug zum Aufbereiten der Informationen in `/proc/net/stat`. Der Name steht für "Linux Network Statistics".

Beispiele für `lnstat` aus dem Handbuch:

```
# 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_conntrack
    Use only the specified file for statistics.
# lnstat -s 0
    Do not print a header at all.
# lnstat -s 20
    Print a header at start and every 20 lines.
# lnstat -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** Übertragen von Daten an TCP basierte Server

**OPT\_NETIO** Performancemessung im Netzwerk

Analog zu “iperf” kann mit “netio” eine Performancemessung des Netzwerks durchgeführt werden. Dazu wird das Programm auf den beiden beteiligten Testsystemen gestartet. Auf dem Server wird das Programm via `netio -s -t` (für Datenaustausch via TCP) bzw. `netio -s -u` (für Datenaustausch via UDP) gestartet. Auf dem Client wird dann via `netio <Server> -t` bzw. `netio <Server> -u` der zugehörige “netio“-Clientprozess gestartet, der dann das Programm auf dem Server kontaktiert und eine Messung vornimmt.

**OPT\_NGREP** Ein grep der direkt auf einem Netzwerkdevice arbeiten kann.

**OPT\_NMAP** Portscanner

Mithilfe von nmap können Betriebssysteme auf offene Ports untersucht werden. Weiterhin liefert das Programm auf Wunsch weitere Informationen z.B. zu MAC-Adressen oder dem verwendeten Betriebssystem.

**OPT\_NTTCP** Netzwerktest

Mit dem Programm NTTCP kann man die Netzwerkgeschwindigkeit testen. Dazu wird auf einer Seite ein Server gestartet und auf einer anderen Seite ein entsprechende Client.

Den Server startet man durch Eingabe von `nttcp -i -v`. Der Server wartet dann auf eine Testanforderung des Clients. Um jetzt z.B die Geschwindigkeit zu testen gibt man auf dem Client `nttcp -t <IP Adresse des Servers>` ein.

So sieht ein gestarteter nttcp Server aus:

```
fli4l-server 4.0.0-testing-x86_64-r60699~# nttcp -i -v
nttcp-l: nttcp, version 1.47
nttcp-l: running in inetd mode on port 5037 - ignoring options beside -v and -p
```

So sieht ein Test mit einem nttcp Client aus:

```
fli4l-client 4.0.0-testing-x86_64-r60699~# 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
```

Die Hilfeseite von nttcp zeigt alle weiteren Parameter:

```
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** Installiert ein tool, dass Änderungen der Routingtabelle überwacht. Primäre Verwendung: Debugging

**OPT\_SOCAT** Das Programm “socat” ist quasi eine verbesserte und mit mehr Funktionen “vollgestopfte” Version des “netcat”-Programms (Seite 5). Mit “socat” können nicht nur diverse Netzwerk-Verbindungen aufgebaut bzw. entgegengenommen werden, sondern auch Daten an UNIX-Sockets, Geräte, FIFOs etc. gesandt bzw. von dort ausgelesen werden. Insbesondere können Quellen und Ziele *verschiedener* Typen miteinander verbunden werden: Ein Beispiel wäre etwa ein via TCP auf einem Port horchender Netzwerk-Server, der empfangene Daten in einen lokalen FIFO schreibt bzw. Daten aus dem FIFO ausliest und diese dann übers Netzwerk an den Client schickt. Siehe <http://www.dest-unreach.org/socat/doc/socat.html> für mehr Informationen sowie Anwendungsbeispiele.

**OPT\_TCPDUMP** debug

Mit dem Programm `tcpdump` kann Netzwerkverkehr beobachtet, ausgewertet mitgeschnitten werden. Mehr dazu unter z.B. Google mit den Suchworten “tcpdump man“

`tcpdump <parameter>`

**OPT\_TRACEPATH** Ermittlung der PMTU

Mit dem Programm `tracepath` kann die so genannte “Path MTU” ermittelt werden. Das ist die maximal verwendbare Paketgröße auf dem Pfad vom fli4l-Router zum Zielhost. Größere Pakete müssen entweder fragmentiert werden (IPv4) oder werden verworfen (IPv6). Typischerweise sorgt sich der Linux-Kernel um die Ermittlung der korrekten Path MTU (Stichwort “Path MTU Discovery”). Gelegentlich ist es jedoch nützlich, diese Path MTU herauszufinden, um Probleme im Netzwerk zu finden.

Ein Beispiel für 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
```

Hier erfolgt die Internet-Anbindung über eine DSL-Verbindung zwischen einer AVM Fritz!Box und einem BRAS des Internet-Providers htp. Da für DSL in Deutschland PPP-over-Ethernet (PPPoE) verwendet wird, ist die Path MTU nur 1492 Byte groß.

Ein Beispiel für 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
```

Hier erfolgt die Internet-Anbindung über einen 6in4-Tunnel von SixXS. In der Konfiguration des Tunnels ist eine Tunnel-MTU von 1280 fest eingestellt. Da dies die kleinste erlaubte MTU für IPv6-Verbindungen ist, wird sie nirgendwo anders auf dem Pfad zum Zielhost weiter reduziert.

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

Mit dem Programm dhcpdump können DHCP Pakete genauer analysiert werden. Das Programm setzt auf tcpdump auf und erzeugt leicht lesbare Ausgaben.

Benutzung:

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

Gestartet wird das Programm also bspw. mit folgendem Aufruf:

```
dhcpdump -i eth0
```

Falls gewünscht, kann mit Hilfe regulärer Ausdrücke auch direkt auf eine bestimmte MAC-Adresse gefiltert werden. Der Aufruf sieht dann so aus:

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

Die Ausgabe könnte dann z.B. so aussehen:

```
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

Mit dem Programm wget können Daten von einem Webserver im Batch abgerufen werden. Praktisch ist aber (und deswegen ist wget im fli4l-Paket dabei), dass man damit Umlenkungen des Providers auf den eigenen Webserver nach einem Verbindungsaufbau



auf einfache Weise abfangen kann, z.B. für Freenet. Wie das geht, hat Steffen Peiser in einem Mini-HowTo erklärt.

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

**Wichtig:** Um mit *wget* TLS-Verbindungen aufbauen zu können, sollten im Paket *CERT* via *CERT\_X509\_MOZILLA='yes'* die Mozilla-X.509-Wurzelzertifikate installiert werden.

### 1.1.2 Die Hardware-Erkennung

Oftmals weiß man nicht genau, welche Hardware im eigenen Rechner steckt bzw. welche Treiber man nun genau für seine Netzwerkkarte oder seinen USB-Chipsatz verwenden soll. Die Hardware kann an der Stelle helfen. Sie liefert eine Liste von Geräten im Rechner und wenn möglich den dazugehörigen Treiber. Man kann dabei auswählen, ob die Erkennung gleich beim Booten erfolgen soll (was sich vor einer Erstinstallation empfiehlt) oder später bei laufendem Rechner bequem über das Web-Interface getriggert werden soll. Die Ausgabe könnte dabei z.B. wie folgt aussehen:

```
fli4l 4.0.0-testing-x86_64-r60699 # 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'
```

Hier stecken also im wesentlichen 3 Netzwerkkarten drin, die vom 'via\_rhine'-Treiber verwaltet werden und eine Atheros-Wlan-Karte, die vom madwifi-Treiber verwaltet wird (der Name wird noch nicht korrekt aufgelöst).

**OPT\_HW\_DETECT** Diese Variable sorgt dafür, dass die für die Hardware-Erkennung Dateien auf dem Router landen. Man kann sich die Ergebnisse dann entweder nach dem Booten auf der Konsole ansehen, wenn man *HW\_DETECT\_AT\_BOOTTIME* auf 'yes' gesetzt hat oder im Web-Interface ansehen, wenn man *OPT\_HTTPD* (Seite ??) auf 'yes' gesetzt hat.

Im Web-Interface kann man sich natürlich auch den Inhalt von `/bootmsg.txt` ansehen, wenn man schon einen funktionierenden Netzzugang hat.

**HW\_DETECT\_AT\_BOOTTIME** Startet die Hardware-Erkennung beim Booten. Die Erkennung läuft im Hintergrund (sie dauert ein wenig) und schreibt dann ihre Ergebnisse auf die Konsole und nach `/bootmsg.txt`.

**OPT\_LSPCI** Auflisten aller PCI-Geräte

**OPT\_I2CTOOLS** Tools für I<sup>2</sup>C Zugriffe.

**OPT\_IWLEEPROM** Tool zum Zugriff auf das EEPROM von Intel und Atheros WLAN Karten.

Wird benötigt um z.B. bei ath9k Karten die Reg-Domain passend zu setzen (siehe <http://blog.asiantuntijakaveri.fi/2014/08/one-of-my-atheros-ar9280-minipcie-cards.html>).

**OPT\_ATH\_INFO** Tool zur Hardwarediagnose von WLAN Karten mit Atheros Chipsatz.

Mithilfe dieses Tools können z.B. bei ath5k WLAN Karten detaillierte Informationen über die verwendete Hardware gewonnen werden. Dazu gehören z.B. der verwendete Chipsatz oder Angaben zur Kalibrierung.

**OPT\_FLASHROM** Tool zum flashen von Chipsätzen.

Mithilfe dieses Tools können z.B. PC Engines Boards mit einem neuen BIOS oder einer neuen Firmware versorgt werden. Näheres dazu findet sich auf <http://www.flashrom.org>.

### 1.1.3 Dateien-Tools

**OPT\_E3** Ein Editor für fli4l

Dies ist ein sehr kleiner, in Assembler geschriebener Editor. Er stellt verschiedene Editor-Modi zur Verfügung, die andere („große“) Editoren nachstellen. Um einen bestimmten Modus zu wählen, reicht es e3 mit dem richtigen Befehl zu starten. Eine Kurzübersicht der Tastenbelegung bekommt man, wenn man e3 ohne Parameter startet oder Alt+H drückt (außer im VI-Modus, dort muß man im CMD-Modus „:h“ eintippen). Zu beachten ist auch, dass das Caret-Zeichen (^) für die Ctrl-/Strg-Taste steht.

Befehl	Modus
e3 / e3ws	WordStar, JOE
e3vi	VI, VIM
e3em	Emacs
e3pi	Pico
e3ne	NEdit

**OPT\_MTOOLS** Die mtools stellen eine Reihe von DOS-ähnlichen Befehlen zum vereinfachten Umgang (Kopieren, Formatieren, etc.) mit DOS-Datenträgern bereit.

Die genaue Syntax der Befehle kann in der Dokumentation von mtools nachgeschlagen werden:

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

**OPT\_SHRED** Installiert das Programm *shred* auf dem Router, ein Programm zum gründlichen Löschen von Blockgeräten.

**OPT\_YTREE** Datei-Manager

Installiert Datei-Manager Ytree auf dem Router.

#### 1.1.4 Entwickler-Tools

**OPT\_OPENSSL** Mit dem Programm openssl können z.B. Test der Cryptobeschleuniger durchgeführt werden.

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

**OPT\_STRACE** debug

Mit dem Programm strace können die Funktionsaufrufe, der Ablauf eines Programmes beobachtet werden

```
strace <programm>
```

**OPT\_REAVER** Brute force Angriff aus Wifi WPS PINs

Testet alle möglichen WPS PINS aus um das WPA Passwort zu ermitteln. Details für die Verwendung auf der Kommandozeile bitte nachlesen unter:

<http://code.google.com/p/reaver-wps/>

**OPT\_VALGRIND** Installiert Valgrind auf dem Router.

# **Abbildungsverzeichnis**

# **Tabellenverzeichnis**

# Index

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

HW\_DETECT\_AT\_BOOTTIME, [10](#)

OPT\_ATH\_INFO, [10](#)

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, [10](#)

OPT\_IFTOP, [4](#)

OPT\_IMONC, [4](#)

OPT\_IPERF, [4](#)

OPT\_IWLEEPROM, [10](#)

OPT\_LNSTAT, [4](#)

OPT\_LSPCI, [10](#)

OPT\_MTOOLS, [10](#)

OPT\_NETCAT, [5](#)

OPT\_NETIO, [5](#)

OPT\_NGREP, [5](#)

OPT\_NMAP, [5](#)

OPT\_NTTCP, [5](#)

OPT\_OPENSSL, [11](#)

OPT\_REAVER, [11](#)

OPT\_RTMON, [6](#)

OPT\_SHRED, [10](#)

OPT\_SOCAT, [6](#)

OPT\_STRACE, [11](#)

OPT\_TCPDUMP, [6](#)

OPT\_TRACEPATH, [7](#)

OPT\_VALGRIND, [11](#)

OPT\_WGET, [8](#)

OPT\_YTREE, [11](#)