

Nagios-Workshop:

Eigene Plugins in Perl entwickeln

Workshop \neq Schulung

- Workshop = mehrstündiger Vortrag/ Diskussion, um in ein Thema tiefer einsteigen zu können, als das in einem einstündigen Vortrag der Fall ist.
- keine Schulung = "selbst tun" steht nicht im Vordergrund – wer möchte, kann das aber durchaus tun.

Inhalt

- Standard-Anforderungen an ein Plugin
- Das Perl-Modul Nagios::Plugin
- Auf der Kommandozeile: GetOpt::Long
- Online-Hilfe: Pod::Usage
- Die Sache mit dem Timeout
- Formate für Schwellwerte: Thresholds
- Ausgabe von Performancedaten
- Konfigurationsdateien für Plugins?
- Trubelschießen ('trouble shooting' ;-)
- ePN: Nagios und der embedded Perl Interpreter

Standardanforderungen im Überblick

- (1) Batch-tauglich
- (2) Rückgabewert (Returncode)
- (3) Textausgabe für den Admin
- (4) Online-Hilfe
- (5) Reservierte Optionen
- (6) Schwellwerte/Thresholds
- (7) Timeout
- (8) Performancedaten
- (9) Copyright-Info

Anforderungen I

(1) Batch-tauglich:

- keine graphische Oberfläche
- keine feste Bindung an ein tty
- keine interaktive Eingabe (nur Optionen)

(2) Rückgabewert/Return-Code

Returncode	Service-Check	Host-Check
0	OK	UP
1	WARNING	(*)
2	CRITICAL	DOWN/UNREACHABLE
3	UNKNOWN	DOWN/UNREACHABLE

Anforderungen II

- (3) Textausgabe für den Admin:
 - einzeilig auf STDOUT (ab Nagios 3.0 auch mehrzeilig, aber Rücksicht auf Nagios 2.x nehmen!)
 - Format: *Name_des_Checks Status – Text*
 - z.B.: CHECK_DU OK – /usr (520 MBytes)
- (4) Online-Hilfe
 - anstelle einer Manpage
 - bei Fehlbedienung und mit Option `-h|--help`

Anforderungen III

(5) Reservierte Optionen

Kurzform	Langform	Beschreibung
-h	--help	Online-Hilfe, ggf. kurze und lange Version
-V	--version	Ausgabe der Plugin-Version
-V	--verbose	erhöht die Geschwätzigkeit
-h	--hostname	Hostname (IP-Adresse)
-t	--timeout	Timeout für den Abbruch des Checks
-w	--warning	Threshold für Warnschwelle
-c	--critical	Threshold für kritische Schwelle
-4	--use-ipv4	verwende IPv4
-6	--use-ipv6	verwende IPv6
-C	--community	SNMP-Community-String bei SNMP-Abfragen
-u	--user oder --url	User (für Authentifikation) oder URL
-p	--port oder --password	TCP/UDP-Port oder Passwort für Authentifikation
-a	--authentication	REALM (z.B. Kerberos, SNMPv3) oder Passwort
-l	--logname	Login-Name

Anforderungen IV

(6) Thresholds:

Fehler, falls außerhalb eines Bereiches

<code>[@]start:end</code>	Beschreibung
<code>start ≤ end</code>	start muss kleiner gleich end sein!
<code>end</code>	start = 0
<code>start:</code>	Bereich geht bis Unendlich
<code>~:end</code>	~ bedeutet negatives Unendlich
<code>@start:end</code>	Bereich wird negiert
<code>-w 1:2 -c 1:5</code>	Wert < 1 oder > 5: CRITICAL; 3-5: WARNING
<code>-c @10:20</code>	CRITICAL, falls $10 \leq \text{Wert} \leq 20$

! Thresholds kennen keine Unit of Measurements!

Anforderungen V

(7) Timeout:

- Jedes Plugin soll nach einem Timeout den Check hart abbrechen (Default: 10 sec.)
- Verhindert Scheduling-Probleme

(8) Performancedaten:

text | var=value[uom];warn;crit;min;max

- UOM = Unit Of Measurement:
 - nichts
 - s - Sekunden: s, ms, us, ...
 - % - Prozentangaben (0-100)
 - B - Bytes (auch: KB, MB, GB)
 - c - Counter (stetig wachsender Zähler)

Das Modul Nagios::Plugin

- Autor: Ton Voon
- aktuelle Version: 0.21 (2007-09-24)
- Installation:
 - perl -MCPAN -e 'install Nagios::Plugin'
 - ggf. vor der ersten CPAN-Benutzung das Modul 'CPAN' installieren (Bundle)
 - alternativ: \geq nagios-plugins-1.4.10
- Manpage: man Nagios::Plugin
- **Demo**

Nagios::Plugin ohne CPAN

- Installation aus den nagios-plugins:

```
cd perlmods  
make && make install
```

- Modul wird installiert nach:

```
/usr/local/nagios/bin/perl
```

- Verwendung im Plugin:

```
use FindBin;  
use lib "$FindBin::Bin/../../perl/lib";  
use Nagios::Plugin;
```

- Suchpfad ist nun relativ zum Plugin-Verzeichnis

Nagios::Plugin

- Konstanten:
 - OK, WARNING, CRITICAL, UNKNOWN
- Constructor (`new`)
- Optionen: getopt-like
- Exit-Funktionen:
 - `nagios_exit(status, text)`
 - `nagios_die(text [,status])`
- Thresholds
- Performancedaten
- Message-Behandlung (experimentell)

Projekt check_du.pl

```
% du -cs /var/log /var/spool
731468  /var/log
26448  /var/spool
757916 total
```

- Summe aller Dateien ausgeben
- prüfen, ob Schwellwerte überschritten werden
- Performance-Daten
- Online-Hilfe

check_du.pl: Rumpf

```
# -- main
open( OUT, "LANG=C; /usr/bin/du -cs $what 2>&1 |" )
    or $np->nagios_die( "can't start /usr/bin/du" );

while (<OUT>) {
    print "$_" if ($verbose);
    chomp $_;

    $denied++ if ( /Permission denied/i );
    if ( /^(\\d+)\\s+total$/i ) { # last line
        $size = $1;
        last;
    }
}
close (OUT);
```

Nagios::Plugin->new

```
#!/usr/bin/perl -w
use strict;
use warnings;
# -- ggf. FindBin
use Nagios::Plugin;

my $np = Nagios::Plugin->new(
    shortname => "CHECK_DU"
);
```

- Weitere Optionen: **siehe manpage**

Getopt::Long

```
use Getopt::Long qw(:config
    no_ignore_case bundling);

# -- GetOpt
GetOptions(
    "P|path=s"      => \$what,
    "w|warning=s"   => \$warn_threshold,
    "c|critical=s"  => \$crit_threshold,
    "h|help"        => \$help,
    "V|version"     => \$printversion,
    "v|verbose+"    => \$verbose,
) or exit_with_unknown_and_help;

argumente_auf_stichhaltigkeit_pruefen;
```

Getopt/Hilfe: entweder/oder

- **Entweder** nur die Funktionen von Nagios::Plugin verwenden
- **Oder** nur Getopt::Long einsetzen
- Keinesfalls mischen!
- Dasselbe gilt für die Online-Hilfe

POD - Perl Online Doku

- Inline-Dokumentation, steht direkt im Skript selbst
- formatierte Ausgabe als Manpage, usw.
- Aufruf: `perldoc skript`
- für die Batch-Verarbeitung
 - `pod2html`, `pod2latex`, `pod2man`
 - `pod2text`, `pod2usage`
- `man perlpod`; `man perldoc`; `man man`

POD - Anweisungen

```
=pod  
=head1 Heading Text  
=head2 Heading Text  
=head3 Heading Text  
=head4 Heading Text  
=over indentlevel  
=item stuff  
=back  
=begin format  
=end format  
=for format text...  
=encoding type  
=cut
```

Demo

POD - everywhere

```
#!/usr/bin/perl -w
=head1 NAME
...
=cut
use strict;
...
=head1 OPTIONS
...
=cut
... perl code ...
=head1 AUTHOR
...
=cut
```

Modul Pod::Usage

```
pod2usage (  
    -msg          => $message_text ,  
    -exitval     => $exit_status  ,  
    -verbose     => $verbose_level,  
    -output      => $filehandle  );
```

- Verbose:
 - 0: SYNOPSIS
 - 1: SYNOPSIS|OPTIONS|ARGUMENTS
 - 2: Alles
 - 99: über "-sections" frei wählbar
- Output:
 - stdout: bei verbose = 0 oder 1
 - stderr: bei verbose = 2

Pod::Usage in check_du.pl

```
GetOptions(...) or pod2usage(  
    -exitval => UNKNOWN,  
    -verbose => 0,  
    -msg      => "**unknown argument found**" );  
  
pod2usage(-verbose => 2,  
    -exitval => UNKNOWN,  
    -output  => \*STDOUT) if ( $help );  
  
pod2usage(-msg => "\n$0 -- version: $version\n",  
    -verbose => 0,  
    -exitval => UNKNOWN) if ( $printversion );  
  
pod2usage(-msg => "**no path/pattern specified",  
    -verbose => 0,  
    -exitval => UNKNOWN) if ( "$what" eq "" );
```

Thresholds

```
# -- thresholds
$np->set_thresholds(
    warning    => $warn_threshold,
    critical   => $crit_threshold
);

# -- check
$result = $np->check_threshold($size);

$np->nagios_exit(
    $result, "check size: $size kByte"
);
```

Messages in Nagios::Plugin

- noch sehr experimentell und veränderlich, daher nicht produktiv benutzen!
- ermitteln aus vorhandenem Text einen Fehlercode:
 - `add_message(<CODE>, $message);`
 - `check_messages()`
- Details siehe Manpage

Performancedaten

```
$np->add_perfdata(  
    label      => "size",  
    value      => $size,  
    uom        => "kB",  
    threshold  => $np->threshold()  
);  
  
# -- Ausgabe  
$np->nagios_exit(  
    $result, "check size: $size kByte");
```

```
CHECK_DU WARNING - check size: 100 kByte |\  
size=100kB;200:;100:
```

Timeout

```
# ... getopt ...
$SIG{ALRM} = sub {
    $np->nagios_die("Timeout reached");
};
alarm($timeout);
# ... eigentlicher Code ...
alarm(0);    # timeout stoppen
```

Achtung: alarm() nicht zusammen mit sleep() verwenden!

Konfigurationsdateien

```
$Config = Nagios::Plugin::Config->read(
    '/etc/nagios/plugins.ini'
);
$Config = Nagios::Plugin::Config->read();

$rootproperty = $Config->{$_}->{rootproperty};
$pi           = $Config->{mathe}->{pi};
$euler        = $Config->{mathe}->{euler};

# /etc/nagios/plugins.ini - windows ini style
rootproperty=10.0

[mathe]
pi=3.1415
euler=2.78
```

Troubleshooting

- Pragmas zur Fehlervermeidung/-Suche:

```
#!/usr/bin/perl -w
use strict;
use warnings;
...
```

- Deparse:

```
perl -MO=Deparse plugin.pl
```

- Debugging-Ausgaben:

```
print "xyz\n" if ($debug);
```

Literatur/Bücher

- Larry Wall, Tom Christiansen, JonOrwant, Randal Schwartz: **Programmieren in Perl** (O'Reilly)
- Tom Christiansen, Nathan Torkington: **Perl Kochbuch** (O'Reilly)
- cromatic: **Perl Hacks** (O'Reilly)
- Damian Conway: **Perl Best Practices** (O'Reilly)
- Fahrid Hajji: **Perl** (Addison-Wesley)
- Johan Vromans: **Perl 5 kurz & gut**

Literatur/Internet

- www.cpan.org
- www.annocpan.org
- perldoc.perl.org
- nagios.sourceforge.net/docs/3_0
- nagiosplug.sourceforge.net
- nagiosplugins.org (**neu**)