

Table des matières

Tester un port en TCP ou UDP 3
Vérifier si un fichier a été modifié depuis n secondes 3

Tester un port en TCP ou UDP

```
#!/bin/perl
use Net::Ping;
my ($host,$port)=(shift,shift);
my $timeout=5;
$sp = Net::Ping->new("tcp", $timeout);
$sp->{port_num} = $port;
$ret=$sp->ping($host);
# 0=ok, 1=non joignable
undef($sp);
exit($ret==0);
```

ou

```
#!/usr/bin/perl -w
use strict;
use Socket;
use File::Basename;

my ($remote,$port,$iaddr,$paddr,$proto,$ligne,@server);
my $fic = ("serv.txt");
open(FIC,"<$fic") or die "Impossible d'ouvrir $fic";
while(<FIC>)
{
    $ligne =~ "$_";
    chomp($ligne);
    @server = split(/;/,$ligne);
    $remote = $server[0];
    $port = $server[1];
    if ($port =~ /\D/) { $port = getservbyname($port, 'tcp') }
    die "No port" unless $port;
    $iaddr = inet_aton($remote);
    $paddr = sockaddr_in($port,$iaddr);
    $proto = getprotobyname('tcp');
    socket(SOCK,PF_INET,SOCK_STREAM,$proto);
    print "\n$remote $port :";
    connect(SOCK,$paddr) || next;
    print "OK";
    close(SOCK) || die "close: $!";
}
close(FIC);
print "\n";
```

On crée un fichier serv.txt qui a pour format :

server;port



Aucun soucis en TCP, par contre en UDP ça peut ne pas fonctionner (dépend du serveur et du service en face).

Vérifier si un fichier a été modifié depuis n secondes

```
use File::Basename;
use File::Path;-br/>
$FILE_FAX="d:\\apps\\cft\\temp\\fax.zip";
$FILE_RTG="c:\\apps\\cft\\temp\\rtg.zip";
$FILE_LOG="c:\\apps\\log\\exploit\\surv_seekfax.log";-br/>
@temps = localtime(time);
$jour = sprintf("%02d",$temps[3]);
$mois = sprintf("%02d",$temps[4]+1);
$annee = sprintf("%02d",$temps[5]+1900);
$heure = sprintf("%02d",$temps[2]);
$min = sprintf("%02d",$temps[1]);

sub test_fax()
{
    ($dev,$ino,$mode,$nlink,$uid,$gid,$rdev,$size,$time,$mtime,$ctime,$blksize,$blocks) = stat($FILE_FAX);
    ($now) = time();

    ($delay_fax) = $now - $mtime;
```

```

if ($delay_fax > 600) {
    print FICLOG "$jour/$mois/$annee $heure:$min ERROR SEEKFAX_____Un fichier fax.zip est present depuis plus de 5 minutes : $FILE_FAX\n";
}
}

open(FICLOG,">>$FILE_LOG") || die "Impossible d'ouvrir $!";
if (-e $FILE_FAX) { test_fax } else { print FICLOG "$jour/$mois/$annee $heure:$min OK pas de fichier fax.zip\n"; }
if (-e $FILE_RTG) { test_rtg } else { print FICLOG "$jour/$mois/$annee $heure:$min OK pas de fichier rtg.zip\n"; }
close(FICLOG);

```

vxtree.pl

```

#!/usr/bin/perl
#
# vxtree.pl - provides a tty version of the "view layout" option in vmsa.
#             best viewed in wide xterm
#
# Aug 2003 - Richard Quixley JPMorgan
#
#
# Identifies the following volume types:
#
# 1. concat
# 2. stripe = (raid-0)
# 3. mirror = (raid-1)
# 4. raid = (raid-5)
# 5. mirrored-stripe = (raid-0+1)
# 6. [layered volume] - striped-mirror ("striped-pro") = (raid-1+0 or raid-10)
# 7. [layered volume] - concat-mirror ("concatenated-pro")
#
# USAGE:
# vxtree.pl -a [dg1 dg2 dg3...] - display all disks and volumes
# vxtree.pl -d [dg1 dg2 dg3...] - display all disks
# vxtree.pl -g [dg1 dg2 dg3...] - display all volumes
# vxtree.pl -v [vol1 vol2 vol3...] - display all listed volumes
# vxtree.pl -s [dg1 dg2 dg3...] - display summary
# vxtree.pl -V - display vxvm version

use Getopt::Std;

sub check_vxvm {
    ( `swlist VRTSvxvm` ) || die "\nERROR: No Veritas packages installed - exiting...\n\n";
}

sub get_dg {
    @dg_list=();
    @dg_list=`/usr/sbin/vxprint -thf|grep "^dg" 2> /dev/null`;
    my $n=0;
    foreach $item (@dg_list) {
        my @f = split(" ", $item, 9999) ;
        $dg_list[$n] = $f[1];
        ++$n;
    }
}

sub check_root {
    my $id=`/usr/xpg4/bin/id -u`;
    if ( $id > 0 ) {
        print "\nWARNING: disk group free space info not available when run as non-root\n";
    }
}

# do disks
sub get_disks {
    &check_root;
    my $arg;
    $arg=scalar (@ARGV);
    if ( $arg < 1 ) {
        &get_dg;
    } else {
        @dg_list=@ARGV;
    }
}

```



```

@sub_vol_child_volname_array=();

foreach $vol (@dg_vols) {
    @dg_vol_array='/usr/sbin/vxprint -thfg $dg -v $vol 2> /dev/null';
    @format=();
    @format='/usr/sbin/vxprint -lvg $dg -v $vol 2> /dev/null';
    my @format2=split("=", $format[4], 9999);
    my @format_pl=split(" ", @format2[1], 9999);
        $k=0;          # count volumes
        $pr=0;        ### print array
        @print_array=(); # zero array
        $sub_vol_flag="false"; # starting condition
        # *** $dg_vol_array is last index of array
        # *** whereas "scalar (@dg_vol_array)" is the number of indices
        while ($k <= $dg_vol_array) {
            @vol_elements = split(" ", $dg_vol_array[$k], 9999);
            if ($vol_elements[0] eq "v") {
                $reset_plex="true";
                $vol_size=($vol_elements[5])/2048000;
                $remember_vol="$vol_elements[1]";
                $remember_vol_array_no="$pr";
                # test whether volume is present in sub-vol array, if it is we won't output it yet
                foreach $item (@sub_vol_child_volname_array) {
                    if ($vol_elements[1] eq $item) {
                        $sub_vol_flag="true";
                        last;
                    }
                }
            }
            } elsif ($vol_elements[0] eq "pl") {
                if ( $reset_plex ne "false" ) {
                    $vol_format=($vol_elements[6]);
                    if ( $vol_format eq "CONCAT" ) {
                        $vol_format="concat"
                    } elsif ( $vol_format eq "STRIPE" ) {
                        $vol_format="stripe"
                    } elsif ( $vol_format eq "RAID" ) {
                        $vol_format="raid"
                    }
                }
                $reset_plex="false";
                if (scalar (@format_pl) >=2 && $vol_format ne "raid") {
                    $vol_mir_format="mirrored-";
                }
                # now overwrite our vol line now we know its format
                $print_array[$remember_vol_array_no]=sprintf " |\n%14s%-20s%50s%-9s%-15s%11s%6.2f Gb\n",'|___ VOL ', $remember_vol , '----- layout=', $vol_mir_format, $vol_format, '----- size
', $vol_size;

            } else {
                $vol_mir_format="";
                $print_array[$remember_vol_array_no]=sprintf " |\n%14s%-20s%50s%-24s%11s%6.2f Gb\n",'|___ VOL ', $remember_vol , '----- layout=', $vol_mir_format, $vol_format, '----- size ',
$vol_size;
            }
            unless ( $Options{s} ) { $print_array[$pr]=sprintf "%25s%-85s[%6.2f Gb]\n",'|___ PLEX ', $vol_elements[1], $vol_size;
                $remember_plex="$vol_elements[1]";
                $get_plex_log="$vol_elements[4]";
            }
            } elsif ($vol_elements[0] eq "sv") {
                $sub_vol_flag="true";
                $sub_vol_child_volname="$vol_elements[3]";
                $sub_vol_parent_plex="$remember_plex";
                $sub_vol_parent_vol="$remember_vol";
                push(@sub_vol_parent_volname_array, $sub_vol_parent_vol);
                push(@sub_vol_parent_plexname_array, $sub_vol_parent_plex);
                push(@sub_vol_child_volname_array, $sub_vol_child_volname);
                push(@remember_vol, $sub_vol_child_volname);
            }
            } elsif ($vol_elements[0] eq "sd") {
                $size=($vol_elements[5])/2048000;
                if ($vol_elements[6] eq "LOG" || $get_plex_log eq "LOG" ) { $log="DRLog" } else { $log=" " }
                unless ( $Options{s} ) { $print_array[$pr]=sprintf "%40s%-19s%-10s%-24s[%6.3f Gb]\n",'|___ sub-disk ', $vol_elements[1], $vol_elements[7], $log, $size;
            }
            }
            # if vol_elements
            ++$k; # k=number of volumes
            ++$pr; # pr=load print array
        } # while $k <= $dg_vol_array

# print out our data...
$l=0;
while ($l <= $pr) {
    if ( $sub_vol_flag ne "true" && $print_array[$l] ) {
        print "$print_array[$l]";
    }
}

```

```

    }
    ++$l; # l=unload print array
    } # while $l <= $pr

} # foreach @dg_vols

# now process all data for layered volumes...

%seen = ();
@unique_sub_vol_parent_volname_array=();
foreach $item (@sub_vol_parent_volname_array) {
    push(@unique_sub_vol_parent_volname_array, $item) unless $seen{$item}++;
} # foreach item

%seen = ();
@unique_sub_vol_parent_plexname_array=();
foreach $item (@sub_vol_parent_plexname_array) {
    push(@unique_sub_vol_parent_plexname_array, $item) unless $seen{$item}++;
} # foreach item

if ( scalar (@unique_sub_vol_parent_volname_array) >= 1 ) {
    $mm=0;
    foreach $svp (@unique_sub_vol_parent_volname_array) {
# print "\n";
        my $nn=0;
        @svp_vol_array=();
        @svp_vol_array='/usr/sbin/vxprint -thfg $dg -v $svp 2> /dev/null`;
        while ($nn <= $#svp_vol_array) {
            @vol_elements3 = split(" ", $svp_vol_array[$nn], 9999);
            if ( scalar (@$svp) == 1 ) {
                $format="concat-mirror";
            } else {
                $format="striped-mirror";
            }
            my $size=($vol_elements3[5]/2048000);
            if ($vol_elements3[0] eq "v") {
                printf "\n%14s%-20s%50s%-15s%20s%6.2f Gb\n",'|___ VOL ', $vol_elements3[1], '----- layout=', $format, '----- size ', $size;
            } elsif ($vol_elements3[0] eq "pl") {
                unless ( $Options{s} ) {printf "%25s%-85s[%6.2f Gb]\n",'|___ PLEX ', $vol_elements3[1], $size;}
            }
        }
        ++$nn;
    }

    foreach $svv (@$svp) {
        $n=0;
        @sv_vol_array=();
        @sv_vol_array='/usr/sbin/vxprint -thfg $dg -v $svv 2> /dev/null`;
        while ($n <= $#sv_vol_array) {
            @vol_elements2 = split(" ", $sv_vol_array[$n], 9999);
            my $size=($vol_elements2[5]/2048000);
            if ($vol_elements2[5] eq "LOGONLY" || $vol_elements2[6] eq "LOG" ) { $log="DRLog " } else { $log="    " }
            if ($vol_elements2[0] eq "v") {
                printf "%37s%-68s(%6.2f Gb)\n",'|___ S-VOL ', $vol_elements2[1], $size;
            } elsif ($vol_elements2[0] eq "pl") {
                unless ( $Options{s} ) {printf "%48s%-25s%-26s[%6.2f Gb]\n",'|___ PLEX ', $vol_elements2[1], $log, $size;}
            } elsif ($vol_elements2[0] eq "sd") {
                #if ($vol_elements2[6] eq "LOG" ) { $log="DRLog " } else { $log="    " }
                unless ( $Options{s} ) {printf "%63s%-15s%-10s%-6s[%6.3f Gb]\n",'|___ sub-disk ', $vol_elements2[1], $vol_elements2[7], $log, $size;}
            } # if
        }
        ++$n;
    } # while
} # foreach $svv (@$svp)
++$mm;
} # foreach unique_sub_vol_parent_volname_array
} # if scalar unique_sub_vol_parent_volname_array
} # foreach (@dg_list)
} # sub get_data

sub do_usage {
    print "\nUSAGE:\n\n";
    print " \vxtree.pl -flag [optional args]\n\n";
    print " \vxtree.pl -a [dg1 dg2 dg3...]\n - display all disks and volumes\n";
    print " \vxtree.pl -d [dg1 dg2 dg3...]\n - display all disks\n";
    print " \vxtree.pl -g [dg1 dg2 dg3...]\n - display all volumes\n";
    print " \vxtree.pl -v [vol1 vol2 vol3...]\n - display all listed volumes\n";
}

```

```

print " \vxtree.pl -s [dg1 dg2 dg3...]" - display summary\n";
print " \vxtree.pl -V" - display Veritas vxvm version\n";
print " \vxtree.pl -h" - help\n";
print "\n";
}

sub do_help {
print "\nvxtree identifies the following volume types:\n=====\n";
print "\n1. concat\n";
print "eg. |__ VOL vol1 ----- layout=concat ----- size 0.10 Gb\n";
print " |__ PLEX vol1-01 [ 0.10 Gb]\n";
print " |__ sub-disk testdg03-01 c3t3d0 { 0.102 Gb}\n";
print "\n2. stripe = (raid-0)\n";
print "eg. |__ VOL vol2 ----- layout=stripe ----- size 0.10 Gb\n";
print " |__ PLEX vol2-01 [ 0.10 Gb]\n";
print " |__ sub-disk testdg04-01 c3t4d0 { 0.052 Gb}\n";
print " |__ sub-disk testdg03-03 c3t3d0 { 0.052 Gb}\n";
print "\n3. mirror = (raid-1)\n";
print "eg. |__ VOL vol3 ----- layout=mirrored-concat ----- size 0.10 Gb\n";
print " |__ PLEX vol3-01 [ 0.10 Gb]\n";
print " |__ sub-disk testdg04-03 c3t4d0 { 0.102 Gb}\n";
print " |__ PLEX vol3-02 [ 0.10 Gb]\n";
print " |__ sub-disk testdg03-04 c3t3d0 { 0.102 Gb}\n";
print "\n4. raid = (raid-5)\n";
print "eg. |__ VOL vol4 ----- layout=raid ----- size 0.10 Gb\n";
print " |__ PLEX vol4-01 [ 0.10 Gb]\n";
print " |__ sub-disk testdg04-04 c3t4d0 { 0.052 Gb}\n";
print " |__ sub-disk testdg03-05 c3t3d0 { 0.052 Gb}\n";
print " |__ sub-disk testdg01-01 c2t0d0 { 0.052 Gb}\n";
print "\n5. mirrored-stripe = (raid-0+1)\n";
print "eg. |__ VOL vol5 ----- layout=mirrored-stripe ----- size 0.10 Gb\n";
print " |__ PLEX vol5-01 [ 0.10 Gb]\n";
print " |__ sub-disk testdg04-05 c3t4d0 { 0.052 Gb}\n";
print " |__ sub-disk testdg03-06 c3t3d0 { 0.052 Gb}\n";
print " |__ PLEX vol5-02 [ 0.10 Gb]\n";
print " |__ sub-disk testdg01-03 c2t0d0 { 0.052 Gb}\n";
print " |__ sub-disk testdg02-03 c3t0d0 { 0.052 Gb}\n";
print "\n6. [layered volume] - striped-mirror (\\"striped-pro"\") = (raid-1+0 or raid-10)\n";
print "eg. |__ VOL vol6 ----- layout=striped-mirror ----- size 0.10 Gb\n";
print " |__ PLEX vol6-03 [ 0.10 Gb]\n";
print " |__ S-VOL vol6-L01 ( 0.05 Gb)\n";
print " |__ PLEX vol6-P01 [ 0.05 Gb]\n";
print " |__ sub-disk testdg04-06 c3t4d0 { 0.050 Gb}\n";
print " |__ PLEX vol6-P02 [ 0.05 Gb]\n";
print " |__ sub-disk testdg01-04 c2t0d0 { 0.050 Gb}\n";
print " |__ S-VOL vol6-L02 ( 0.05 Gb)\n";
print " |__ PLEX vol6-P03 [ 0.05 Gb]\n";
print " |__ sub-disk testdg03-07 c3t3d0 { 0.050 Gb}\n";
print " |__ PLEX vol6-P04 [ 0.05 Gb]\n";
print " |__ sub-disk testdg02-04 c3t0d0 { 0.050 Gb}\n";
print "\n7. [layered volume] - concat-mirror (\\"concatenated-pro"\")\n";
print "eg. |__ VOL vol7 ----- layout=concat-mirror ----- size 0.10 Gb\n";
print " |__ PLEX vol7-03 [ 0.10 Gb]\n";
print " |__ S-VOL vol7-L01 ( 0.10 Gb)\n";
print " |__ PLEX vol7-P01 [ 0.10 Gb]\n";
print " |__ sub-disk testdg04-07 c3t4d0 { 0.100 Gb}\n";
print " |__ PLEX vol7-P02 [ 0.10 Gb]\n";
print " |__ sub-disk testdg03-08 c3t3d0 { 0.100 Gb}\n";
}

sub get_bighelp {
man vxintro`;
}

sub get_version {
$version=`swlist -l product VRTSvxvm | grep VRTSvxvm | awk '{print $2}' | cut -d, -f1|cut -d: -f2`;
print "\nVeritas Volume Manager version: $version\n";
}

&check_vxvm;

%Options;
getopts('vdgashVH', \%Options);

if ( $Options{v} ) {

```

```
    &get_version;
    &get_dg;
    &get_vols;
} elsif ( $Options{d} ) {
    &get_version;
    &get_disks;
} elsif ( $Options{g} ) {
    &get_version;
    &get_dg;
    &get_vols;
} elsif ( $Options{a} ) {
    &get_version;
    &get_disks;
    &get_vols;
} elsif ( $Options{s} ) {
    &get_version;
    &get_disks;
    &get_vols;
} elsif ( $Options{h} ) {
    &do_help;
} elsif ( $Options{H} ) {
    &get_bighelp;
} elsif ( $Options{V} ) {
    &get_version;
} else {
    &do_usage;
}

#####
### This script is submitted to BigAdmin by a user of the BigAdmin community.
### Sun Microsystems, Inc. is not responsible for the
### contents or the code enclosed.
###
### Copyright Sun Microsystems, Inc. ALL RIGHTS RESERVED
### Use of this software is authorized pursuant to the
### terms of the license found at
### http://www.sun.com/bigadmin/common/berkeley_license.jsp
#####
```

From:
<https://unix-bck.ndlp.info/> - **Where there is a shell, there is a way**

Permanent link:
https://unix-bck.ndlp.info/doku.php/informatique:langages:scripts_perl

Last update: **2016/10/10 15:41**