wget code

Below is the code in perl for the function that gathers the data from PingER Measurement Agents (MAs). See around lines 39-47 and 343. It uses wget (see for example http://perlmaven.com/simple-way-to-fetch-many-web-pages)

```
#!/usr/local/bin/perl -w
#The following code is placed at the top to ensure we are able to use perl -d
#and stop things before they call other things.
my $debug; #For cronjobs use -1, for normal execution from command line use 0,
           #for debugging information use > 0, max value = 3.
$|=1;
if (-t STDOUT) {$debug=0;}
else           {$debug=-1;} #script executed from cronjob
my $user=scalar(getpwuid($<));
########################################
(my $progname = $0) =~ s'^.*/'';   # strip path components, if any
#my $version="1.1, 6/6/07, Cottrell";
#my $version="2.1, 9/21/07, Cottrell";
#my $version="2.2, 3/16/08, Cottrell";#Replace Lynx with wget for timeout
#my $version="2.3, 2/7/09, Cottrell"; #Reduced progress messages
#my $version="2.4, 6/4/2011, Cottrell";#Clean up of 3 command loops, add STDOUT & target, speed up
#my $version="2.5, 11/12/2011, Cottrell";#Allow for gzip to be at /bin/gzip.
#my $version="2.6. 7/4/2014, Cottrell";#Don't replace old data if there is no new
my $version="2.7. 1/2/2015, Cottrell";#Make IPv6 comliant.
#my $version="2.8. 1/2/2015, Cottrell";#make data all lower case
use strict;
use lib "/afs/slac/package/pinger";
use PingIt qw(pingit);
my $base_dir="/afs/slac/package/pinger";
my $base_data="/nfs/slac/g/net/pinger/pingerdata/hep";
my $timeout=60;#Default timeout for wget
#use Sys::Hostname;
#my $ipaddr=gethostbyname(hostname());
#my ($a, $b, $c, $d)=unpack('C4',$ipaddr);
#my ($hostname,$aliases, $addrtype, $length, @addrs)=gethostbyaddr($ipaddr,2);
use Net::Domain qw(hostname hostfqdn hostdomain);
my $hostname = hostfqdn();
unless(($hostname=~/(([a-z0-9]+|([a-z0-9]+[-]+[a-z0-9]+))[.])+/)){#Name
  print "hostname=$hostname, not a valid IP name\n";
  exit 101;
}
use Socket;
my $ipaddr=inet_ntoa(scalar(gethostbyname($hostname||'localhost')));
my $wget0='/usr/local/bin/wget';
if(-x $wget0){}
else{$wget0='/usr/bin/wget';}
unless(-x $wget0) {
  print "$progname: $wget0 not executable on $hostname by $user\n";
  exit 101;
}
my $wget="$wget0 --no-check-certificate --no-verbose --tries=1 "
        . '--quiet -O - ';
#pinger.new.cf is a configuration file giving the path for various files.
our %PATH;  #is provided by require "$base_dir/pinger.new.cf";
my $config="$base_dir/pinger.new.cf";
require "$config";
#nodes.cf gives the hosts configurations file
my $nodes="$base_dir/nodes.cf";
our %NODE_DETAILS; our %data_servers; #these are provided by require $nodes
require "$nodes"; #Fill NODE_DETAILS hash with NODEDETAILS/Guthrie database info
########################################################
my $USAGE="
$progname gathers the data from the monitoring sites worldwide,
 usually run once a day from a cronjob
 in pinger\@pinger.slac.stanford.edu/.trs/crontab
 It calls the pingER ping_data.pl CGI script at the monitoring site via wget.
To call getdata.pl:
 getdata.pl hostname [yyyy-mm-dd[ debug[ STDOUT[ target]]]
 the hostname may be 'all' to indicate to gather data from all sites
   (this is the default).
```

```
 if yyyy-mm-dd is not specified then yesterday's data will be gathered
    Possible other values apart from yyyy-mm-dd are today or yesterday
 the default debug=$debug, it is an integer, larger values give more debugging
 if STDOUT is specified then the output will go to STDOUT instead of to a file
 if target is specified then it will only return data for that target
Security:
 Unless run from the pinger account or the debug option is set to -1 (e.g. if
 it is run as a batch or cron job),
 then it will request confirmation before replacing
 an existing .gz file, e.g. you will get a request of the form:
 gzip: /nfs/slac/g/net/pinger/pingerdata/hep/data/multivac.sdsc.edu/ping-2011-01-16.txt.gz already exists; do
you wish to overwrite (y or n)?
It uses:
  $wget
  $nodes
  $config
  $base_data
  PingIt
  Socket
  Net::Domain

Input:
  Configuration data is read from $config
  The Node meta data is from $nodes
Output to files of the form:
 /nfs/slac/g/net/pinger/pingerdata/hep/data/pcgiga.cern.ch/ping-2006-09-28.txt.gz
Output format:
 pinger.slac.stanford.edu 134.79.240.30 ping.slac.stanford.edu 134.79.18.21 100 1152489601 10 10 0.255 0.341
0.467 0 1 2 3 4 5 6 7 8 9 0.287 0.380 0.467 0.391 0.327 0.387 0.291 0.332 0.255 0.299
Example calls:
 getdata.pl pinger.slac.stanford.edu #gets yesterday's data
 getdata.pl pinger.slac.stanford.edu 2006-08-11 2
 getdata.pl | tee /afs/slac/g/www/www-iepm/pinger/slaconly/getdata.err
 getdata.pl all 2006-09-11 | tee /afs/slac/g/www/www-iepm/pinger/slaconly/getdata.err
 getdata.pl -v
 getdata.pl pinger.slac.stanford.edu 2011-06-04 1
 getdata.pl pinger.slac.stanford.edu 2011-06-04 0 STDOUT access.sprace.org.br
 getdata.pl pinger.ictp.it 2011-06-04 -1 STDOUT www.kek.jp
 getdata.pl all 2012-02-11 1
Use:
It is run from a trscrontab and starts running at 32 minutes after
midnight Pacific time each night. It gathers data from about 90 monitored
hosts (Beacons) for most monitoring hosts.  Pakistani monitoring hosts
have more monitored hosts since they also monitor Pakstani hosts
(as well as Beacons).  It takes about 135 minutes to gather the data from
all the monitors.
For each monitoring node the archive node(e.g. pinger.slac.stanford.edu)
makes 3 wgets each getting 8 hours of data from yesterday. Typically in a
day there can be ~14000 records(i.e. for each host monitored by the
monitoring host (e,g, airuniversity.seecs.edu.pk, there are about 140
Beacons plus Pakistani hosts. There is a line each 30 mins, i.e 48 lines
and two times that for 100 and 1000Byte pings,  and each record is say
200 bytes long, ie 2,800,000 Bytes. For airuniversity.seecs.edu.pk
111.68.96.101 each of the 3  wgets from pinger.slac.stanford.edu take
~ 7 secs.  Thus the data rate for about 20 secs is about 1.4Mbits/sec
Version $version
";
########################################################################
###### Set some local variables #####
my $start_time=time();
umask 0002;
# open(STDERR, '>&STDOUT');# Redirect stderr onto stdout
$|=1;
if(!-e "$base_data"){mkdir("$base_data",0775);}
if(!-e "$base_data/data"){mkdir("$base_data/data",0775);}
########################################################################
##### Look to see if command line options were given & set defaults #####
if(!defined($ARGV[0])) {$ARGV[0]="all";}
my $specified_site=$ARGV[0];
if($ARGV[0] eq "-v") {print "$USAGE"; exit 100;};
$specified_site="all" unless $specified_site=~/\w/;
```

```
my $specified_date=$ARGV[1];
if(!($specified_date)) {$specified_date="yesterday";}
if(defined($ARGV[2])) {
  $debug=$ARGV[2];
}
my $debug0=$debug;
my $datahandle='DATA';
if(defined($ARGV[3])) {
  $datahandle=$ARGV[3];
}
my $target="";
if(defined($ARGV[4])) {
  $target="sites=$ARGV[4]&";
}
#########################################################################
##### If a date has been specified, check it is the correct format #####
##### If not the correct format, complain and exit.                #####
my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst);
if($specified_date=~/today/) {
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=gmtime();
  $year+=1900;
  $mon=sprintf "%2d", ++$mon; $mon=~s/ /0/;
  $mday=sprintf "%2d", $mday; $mday=~s/ /0/;
}
elsif($specified_date=~/yesterday/) {
  $specified_date="yesterday";
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=gmtime(time-24*3600);
  $year+=1900;
  $mon=sprintf "%2d", ++$mon; $mon=~s/ /0/;
  $mday=sprintf "%2d", $mday; $mday=~s/ /0/;
}
else {
  ($year, $mon, $mday)=split(/-/, $specified_date);
  unless($year=~/\d{4}/ && $mon=~/\d{2}/ && $mday=~/\d{2}/) {
    print "Invalid date date=$specified_date, site=$specified_site\n"
        . "$USAGE";
    exit 101;
  }
}
#############################################################
my $total_lines=0;
my $ikey=0;#Counter of number of monitor processed so far
my $matches=0;
my $working=0;
my $ndisabled=0;
my $nodata=0;
my $nmon=scalar(keys(%data_servers));
##########Main loop through all sites to gather data
foreach my $hep_site (sort keys %data_servers) {
  $ikey++;
  $hep_site=~s/\s//;
  if($data_servers{$hep_site}) {
    if($hep_site =~ /www.iiu.edu.pk/) {
      my $a=1;
    }
    if($specified_site=~/^all$/ || $hep_site=~/^$specified_site$/) {
      $matches++;
      my $time_so_far=time()-$start_time;
      print STDERR "($matches/$ikey/".scalar(keys(%data_servers))
          . ")$0 $hep_site $specified_date"
          . "($year-$mon-$mday), "
          . "so far=$time_so_far"."s.\n" if ($debug>0);
      my $time=time();
      my $nline=0;
      if(($hep_site eq "pinger.arn.dz"
       || $hep_site eq "pinger.uum.edu.my"
       || $hep_site eq "pinger.sesame.org.jo")  #Special case since it works but does not ping
       ||(PingIt::pingit($hep_site) >0)) {#Site pings so probably available
        $nline=&get_data("$base_data/data",$hep_site);#Get the data
      }
      else{
```

```
        if($debug>=0) {
          print STDERR "$hep_site($ikey) does not ping.\n";
        }
      }
      $total_lines=$total_lines+$nline;
      if($nline>0) {$working++;}
      $time=time()-$time;
      if($hep_site =~ /\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/) {
        if($debug>=0) {
          print "$progname: monitor site $hep_site is an IP address "
               . "should be a name.\n";
          my $asn="/afs/slac/g/scs/net/netmon/bin/asn.pl";
          #my $asn="/afs/slac/package/pinger/asn.pl";
          my @ans=`$asn $hep_site`;
          print "@ans";
        }
      }
      $time_so_far=time()-$start_time;
      print STDERR "(transfer_time=$time"."s.(timeout=$timeout"."s.))"
          . " $0 transferred $nline lines\n"
          . "  to $base_data/data/$hep_site\n"
          if ($debug>0);
    }
  }
  else {
    print "$progname: unable to determine data server for $hep_site. "
        . "Check it is defined in configuration file.\n";
    next;
  }
}#End foreach my $hep_site (sort keys %data_servers) {
if($total_lines<=0) {
  print "$progname @ARGV: unable to gather any data for monitor=$specified_site "
      . "to $target for date=$specified_date!\n";
  if($matches<=0) {
    print "$progname: unable to match hostname $specified_site, "
        . "hostname not in list of known monitors in $base_dir/nodes.cf\n";
    if($debug>=0) {
      foreach my $hep_site (sort keys %data_servers) {
        print "  $hep_site\n";
      }
    }
  }
}
$start_time=time()-$start_time;
if($debug>=0) {
  print STDERR localtime()
                  ." $progname done, took $start_time seconds for "
                  ."$0 @ARGV to collect $total_lines lines "
                  ."from $matches monitors, of which $working are working\n"
                  #."Disabled=$ndisabled, No data=$nodata\n"
                  ."$total_lines lines of data written to $base_data/data/\n";
}
if($debug==-0.5) {
  print "Gathered $total_lines";
}
exit();
#############################################################################
sub get_data {
  #Gets data from $hep_site and writes it line by line to $base_data/data
  #Example:
  # $result=&get_data("$base_data/data",$hep_site);
  # Uses wget to get data from source host(s). Command is of form:
  #/usr/local/bin/lynx -source   "http://www.slac.stanford.edu/cgi-wrap/ping_data.pl\
  #?in_form=1&begin_hour=00&begin_min=00&begin_sec=00&begin_day=29&begin_month=01\
  #&begin_year=2007&begin_offset=&begin_point=y&end_hour=23&end_min=59&end_sec=59\
  #&end_day=29&end_month=01&end_year=2007&end_offset=&end_point=y"
  #The return value is the number of lines read for the $hep_site
  my $dir=$_[0];
  my $site=$_[1];
#  if ($site eq 'pingermtn.pern.edu.pk') {
#    $debug0=$debug;
```

```
#    $debug=1;
#  }
#  else {$debug=$debug0;}
  ##################Check the site#######################
  if(!defined($NODE_DETAILS{$site}[8])) {
    if($debug>=0) {
      print "Project-type (NODE_DETAILS{$site}[8]) is undefined "
          . "(probably Disabled) in  $nodes";
    }
    return(0);
  }
  if($NODE_DETAILS{$site}[8]=~/D/) {#Monitoring host disabled
    $ndisabled++;
    if($debug>=0) {
      print "--$site ($ndisabled) has been disabled\n";
    }
    return(0);
  }
  if($site =~ /www\.iiu\.edu\.pk/) {
    my $dbug=1;
  }
  ###################Format of %data_servers############
  #%data_servers=(
  #  "brunsvigia.tenet.ac.za" => ["http://brunsvigia.tenet.ac.za/cgi-bin/ping_data.pl?", ],
  #  "cithep130.ultralight.org" => ["http://cithep130.ultralight.org/cgi-bin/ping_data.pl?", ],
  #  "davinci.ampath.net" => ["http://davinci.ampath.net/cgi-bin/ping_data.pl?", ],
  #  ...
  #);
  my $data_dir="$dir/$site";
  my $file_name="ping-$year-$mon-$mday.txt";
  my $failures="$base_data/failures";
  if($debug>0) {
    print STDERR "<br>============================<br>\n".scalar(localtime())
        . " $progname(debug=$debug): Start get_data $_[1] $year-$mon-$mday> $_[0]\n";
  }
  my ($reply, $msg)=("", "");
  if($data_servers{$site}[0]=~/^http/) {
    mkdir("$failures", 0775) unless(-e $failures);
    if($data_servers{$site}[0] !~/\?$/){#add the ? mark to the URL if missing
      $data_servers{$site}[0] .= '?';
    }
    unless ($user eq 'apache') {
      open (FAILED, ">>$failures/log-$mon-$mday")
        or print "$progname: can't open FAILED >>$failures/log-$mon-$mday "
               . "for $user: $!\n";
    }
    ###########################################################
    #Create commands to read the data in 8 hour chunks.
    my @cmd;
    for (my $i=0;$i<3;$i++) {
      my $begin_hour=($i*8);
      my $end_hour  =(($i+1)*8)-1);
      my $timeo="--timeout=$timeout";
      if($target ne ""){#If only getting data for 1 remote host ($target)
                        #then do not need to break into 3 gets
        $end_hour=23;
        $timeo="--timeout=20";#Also use a short timeout
      }
      $cmd[$i]="$wget $timeo '$data_servers{$site}[0]in_form=1&$target"
             .
"begin_hour=$begin_hour&begin_min=00&begin_sec=00&begin_day=$mday&begin_month=$mon&begin_year=$year&"
             . "begin_offset=&begin_point=y"
             . "&end_hour=$end_hour&end_min=59&end_sec=59&end_day=$mday&end_month=$mon&end_year=$year&"
             . "end_offset=&end_point=y'";
      #/usr/bin/wget --no-check-certificate --no-verbose --tries=1 --quiet -O -  --timeout=20 'http://pinger.
ictp.it/cgi-bin/ping_data.pl?in_form=1&sites=www.kek.
jp&begin_hour=0&begin_min=00&begin_sec=00&begin_day=04&begin_month=06&begin_year=2011&begin_offset=&begin_point=
y&end_hour=23&end_min=59&end_sec=59&end_day=04&end_month=06&end_year=2011&end_offset=&end_point=y'
      if($target ne "") {last;}
    }#End for (my $i=0;$i<3;$i++) {
    ##Done with creating commands to get data in 8 hour chunks.
```

```
    #Now get the data in 8 hour chunks
    my $i=0;
    my @reply=();
    $msg='';
    foreach my $cmd (@cmd) {
      print STDERR scalar(localtime()). " $progname: executing step $cmd\n" if ($debug>0);
      $reply[$i]=`$cmd`;
      if($reply[$i] eq "" || $reply[$i] =~ /No data/) {
        $msg .= "wget returned null result or 'No data' for $data_servers{$site}[0] for "
             .  "hours ".($i*8)." through ".((($i+1)*8)-1)." hrs of $year-$mon-$mday\n";
        if($debug>0) {
          $msg .=" from $cmd with result=$reply[$i]\n";
        }
      }
      else {$reply.=$reply[$i];}
      $i++;
    }#End foreach my $cmd (@cmd){
    #Done getting data in 8 hour chunks
    if(($msg ne '') && ($debug>=0)) {
      if(PingIt::pingit($site) <=0) {
        $msg .= "  $site does not ping.\n";
      }
      print STDERR scalar(localtime())
          . " $progname: $site $year-$mon-$mday: cmd=$cmd[0]\n$msg";
    }
    unless (($user eq 'apache')
          ||($user eq 'nobody')){#user apache does not have privs to this file
      print FAILED "$year-$mon-$mday\n$reply\n";
      close FAILED or print "Cannot close FAILED $failures/log-$mon-$mday: $!\n";
    }
  }#END if($data_servers{$site}[0]=~/^http/) {
  else {
    if(-e "$data_servers{$site}[0]-$file_name"){
      print STDERR "$data_servers{$site}[0]-$file_name exists, reading that file\n" if ($debug>0);
      my $cmd="cat $data_servers{$site}[0]-$file_name";
      $reply=`$cmd`;
      if($debug>1) {
        print STDERR "reply=$reply\n";
      }
    }
    else {
      print "$site has invalid NODEDETAILS dataserver=$data_servers{$site}[0] for filename=$file_name\n";
      return(0);
    }
  }
  print STDERR "start of reply: ", substr($reply,0,100), "\n" if ($debug>0);
  my @tokens;
  if(!defined($reply) || $reply eq "") {
    $nodata++;
    if($debug>=0) {
      print STDERR "!reply (nodata=$nodata, data=$working, monitors=$nmon) is empty for $site for $mon/$mday
/$year\n";
    }
  }
  elsif($reply !~ /^$site/) {
    my ($line1,undef)=split(/\n/,$reply);
    if($reply=~/No data/) {
      print STDERR "!!reply got 'No data' for $mon/$mday/$year, "
          . "1st line follows:\n $line1\n";
    }
    else {
      @tokens=split(/\s+/,$line1);
      if(!defined($tokens[0]) || $tokens[0] eq '') {
        print "tokens[0] not defined in $line1, site=$site, dir=$dir, "
            . "for $mon/$mday/$year, reply=$reply\n ";
      }
      my (undef,$domain_rep)=split(/\./,$tokens[0],2);
      my (undef,$domain_req)=split(/\./,$site,2);
      if(!defined($domain_rep) || !defined($domain_req) || $domain_rep eq "") {
        print "!!$progname: wget reply from site $site does not have a '.' "
            . "in hostname '$tokens[0]' for $mon/$mday/$year, "
```

```
          . "1st lines follows:\n"
          . "$line1\n";
    }
    elsif($domain_rep ne $domain_req) {
      print "!!$progname: reply does not start with domain $domain_req in "
          . "site $site for $mon/$mday/$year, 1st line follows:\n "
          . "$line1\n";
    }
    else {
      if(lc($domain_rep) eq lc($domain_req)) {
        if($debug>0) {
          print STDERR "Case does not match between request=$domain_req "
                     . "and data read in $line1\n";
        }
      }
      else {
        print STDERR "$progname: warning reply host ($tokens[0]) "
            . "not equal to requested host($site) but domains are "
            . "equal=$domain_req, 1st line read follows:\n$line1\n";
      }
    }
  }
  @tokens = split(/\s+/,$reply . '1');
  if(($tokens[0] ne "") && ($tokens[0] ne "1")) {
    if($tokens[0] eq "<!DOCTYPE") {
      print "$progname: $site did not provide valid data!\n$reply";
      return(0);
    }
    my ($name1,$net1)=split(/\./,$tokens[0],2);
    my ($name2,$net2)=split(/\./,$site,2);
    if(!defined($net1) || !($net1 eq $net2)) {#Eliminate probable aliases with same network address
      if($tokens[0] eq "No") {
        print "progname: no data for $site for $mon/$mday/$year\n";
      }
      else {
        print STDERR "Warning: found $tokens[0] as monitoring host in "
            . "data gathered from monitoring site at $site.\n"
            . "  There may be an alias between $tokens[0] and $site,\n"
            . "  or you may need to inform the contact at $site\n"
            . "  that the pinger.xml in the configuration file is incorrect\n"
            . "  (e.g. the monitoring site is using its IP address "
            . "rather than its name in the <SrcName> declaration)\n"
            . "  First line: $line1\n";
        if(defined($tokens[5]) && $tokens[5]=~/^\d+$/) {
          print "  ".scalar(gmtime($tokens[5]))." GMT for $tokens[5]\n";
        }
        else {
          print "Invalid Unix GMT time stamp in line\n";
        }
      }
    } # if not defined $net1 or $net1 == $net2
  } # tokens[0] ne "" and tokens[0] ne 1
} #End  if(!defined($reply) || $reply eq "") {
#########################################################
#Validate each line in turn and write good data to DATA
print STDERR "Looking for $data_dir\n" if ($debug>0);
mkdir("$data_dir", 0775) unless (-e $data_dir);
my $dataopen=0;#Set to 1 when the DATA fie is opened. It is not opened
               #Until some good data is found (to avoid wiping out
               #data got at an earlier time.
my $nline=0;
my %seen;
my $skip=0;
my $gt10_valid=0;
my $undef_rcvd=0;
my $datetime;
my $nrecords=0;
my $nbad=0;
foreach my $line (split /\n/, $reply) {
  #Typical $line is:
  #access.sprace.org.br 200.136.80.8 frcu.eun.eg 193.227.1.1 100 1260144888 10 10 297.409 298.677 301.917 1 2
```

```
3 4 5 6 7 8 9 10 298 299 297 297 301 297 297 299 299 297
    $nline++;
    chomp $line;
    if($line eq "") {next;}
    $line=lc($line);
    (my @dataline)=split /\s/, $line;
    ###################Check data validity####################
    if($#dataline<7) {
      $nbad++;
      print "$progname: $site has incomplete record (needs > 7 tokens), "
          . "skipping (line $nline):\n  $line\n";
      next;
    }
    if(!defined($dataline[1]) || $dataline[1]!~/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/ ) {#probably could not get
data
      $nbad++;
      print "$progname: $site has bad monitor IP address($dataline[1]), "
          . "skipping (line $nline):\n  $line\n";
      next;
    }
    if($dataline[2]!~/\w/) {
      $nbad++;
      print "$progname: $site has bad remote name ($dataline[2]): "
          . "skipping (line $nline):\n  $line\n";
      next;
    }
    if(!defined($dataline[5]) || $dataline[5]!~/^\d+$/) {
      $nbad++;
      if(defined($dataline[5])) {
        print "$progname: $site has bad Unix epoch time(token[5]=$dataline[5]):";
      }
      else {
        print "$progname: $site has missing Unix epoch time(token[5]):";
      }
      print "skipping (line $nline):\n  $line\n";
      next;
    }
    if(!defined($dataline[6]) || $dataline[6]!~/^\d+$/){
      $nbad++;
      print "$progname: $site has non-numerical ping packets sent:"
          . "skipping (line $nline):\n  $line\n";
      next;
    }
    #################Data validation completed####################
    my $pair="$site $dataline[0]";
    $datetime = ($dataline[5] ? scalar(gmtime($dataline[5])) : '0');
    if(lc($dataline[0]) ne $site &&
      !defined($seen{$pair}) &&         #already notified?
      $#dataline > 6) {                 #does the line contain data (vs a warning)?
      if(!$NODE_DETAILS{$site}[8]=~/D/){#Only give warning for no Disabled monitoring hosts.
        print "$progname: monitor site $site labels itself as $dataline[0] in(line $nline at $datetime GMT):\n
$line\n";
        $seen{$pair}++;
      }
    }
    if(!defined($dataline[6])) {$undef_rcvd++; next;}
    if($dataline[6]<10) {
      $skip++;
      if($debug>1) {
        print STDERR "$progname: Warning ($skip) data skipped, monitor site $site "
            . "sent < 10 packets in (line $nline at $datetime GMT):\n  $line\n";
      }
      next;
    }
    elsif($dataline[6]!=10) {
      if (defined($dataline[7])) {
        if ($dataline[7]!~/^\d+$/) {
          print "$progname: bad data ($dataline[7]) in (line $nline): $line\n";
        }
        elsif($dataline[7]!=0){#simple unavailable are trivial so ignore & focus on some data received
          $gt10_valid++;
```

```
        if($debug>1) {
          print STDERR "$progname: Warning ($gt10_valid) $site to $dataline[2] "
              . "responded, sent $dataline[6] != 10 pings in "
              . "(line $nline at $datetime GMT):\n  $line\n";
        }
      }
    }
  }
  $nrecords++;
  if($datahandle eq 'STDOUT') {print "$line\n";}
  else {
    unless($dataopen) {
      print STDERR "Opening for 1st line $data_dir/$file_name\n" if ($debug>0);
      open (DATA, ">$data_dir/$file_name") or die("Cannot open $datahandle >$data_dir/$file_name: $!");
    }
    $dataopen++;
    print DATA "$line\n";
  }
}#End foreach my $line (split /\n/, $reply) {
#Done validating and writing the good lines of data to DATA
if($skip>0) {
  if($debug>0) {
    print STDERR "$progname: Warning $site skipped $skip/$nline records with < 10 "
        . "packets sent!\n";
  }
}
if($gt10_valid>0) {
  if($debug>0) {
    print STDERR "$progname: Warning $site sent > 10 & got non zero back $gt10_valid "
        . "times/$nline!\n";
  }
}
if($undef_rcvd>0) {
  if($debug>0) {
    print "$progname: $site skipped $undef_rcvd/$nline records "
        . "with undefined number of received packets!\n";
  }
}
if($nbad>0) {
  print "$progname: $site has $nbad/$nline invalid records!!\n";
}
if($datahandle ne 'STDOUT') {
  unless($dataopen) {
    print STDERR "Opening empty file $data_dir/$file_name\n" if ($debug>=0);
    open (DATA, ">$data_dir/$file_name") or die("Cannot open $datahandle >$data_dir/$file_name: $!");
    $dataopen++;
  }
  unless(close(DATA)) {
    print "progname: can't close ($ikey/"
        . scalar(keys(%data_servers))
        . ") $data_dir/$file_name with $nrecords records\n";
    my @ans=`ls -l $data_dir/$file_name`;
    print @ans;
    return 0;
  }
  print STDERR "$progname: Closed ($ikey/"
      . scalar(keys(%data_servers))
      . ") $data_dir/$file_name with $nrecords records\n" if ($debug>=0);
}
my $rc;
if (-e "$data_dir/$file_name.gz" && ($user eq 'pinger'
 || $debug<0)) {
  if($debug>=0){print "$progname: $PATH{'RM'} $data_dir/$file_name.gz\n";}
  $rc=0xffff & system("$PATH{'RM'}","$data_dir/$file_name.gz");
  if($rc != 0) {
    print "$progname user=$user: can't $PATH{'RM'} $data_dir/$file_name.gz ($rc)\n";
    my @ans=`ls -l $data_dir/$file_name.gz`;
    print @ans;
    return "0";
  }
}
```

```
  unless($datahandle eq 'STDOUT') {
    unless(-x $PATH{'GZIP'}) {$PATH{'GZIP'}='/bin/gzip';}#try another path
    $rc=0xffff & system("$PATH{'GZIP'}",'-f',"$data_dir/$file_name");
    if($rc != 0) {
      print "$progname: can't $PATH{'GZIP'} $data_dir/$file_name ($rc): $?\n";
      my @ans=`ls -l $data_dir/$file_name.*`;
      print @ans;
      return "0";
    }
  }
  return($nline);
}
__END__
```