Open Computing ‘‘Hands-On’’: ‘‘Wizard’s Grabbag’’ Column: October 1994: Listings

Listing: The `chkaddrs` program summarizes Sendmail address processing.

A. Listing of `chkaddrs`:

```
#!/usr/local/bin/perl
# @(#) chkaddrs     Check sendmail address processing
# Based on checksendmail, which was written by
# Gene Kim, Rob Kolstad, and Jeff Polk, July 1990
# Repackaged by Becca Thomas, July 1994

# Configuration section:
$=              = 24;                     # default report page length
$|              = 1;                      # flush output after print,write
$addr_file      = "address.resolve";      # default address input file
$config_file    = "/etc/sendmail.cf";     # default sendmail config file
$tmpfile        = "/tmp/chkaddrs.$$";     # temporary file
$usage = "Usage: $0 [ -a addr-file ] [ -c config-file ] [ -q queuedir ]
    -a address-file         File containing addresses for testing
    -c configuration-file   Alternate sendmail configuration file
    -q queue directory      Alternate mail queue directory\n";

# Process command-line options:
require 'getopts.pl';   # library to process command-line options
&Getopts('a:c:q:') || die "$usage"; # opt x arg into opt_x variable

# Process option arguments:
if ($opt_a) { $addr_file = $opt_a; }    # user-specified addr file
if ($opt_c) { $config_file = $opt_c; }  # alternate config file

# Determine queue directory specified in sendmail.cf config file:
chop($queue_dir = `grep ^OQ $config_file`); # get dir value from file
$queue_dir =~ s/^OQ//;                    # store default for later use

if ($opt_q) { $queue_dir = $opt_q; }    # overwrite default value

# Check for existence of these files and directory:
die "Can't find address file, $addr_file\n"   unless -e $addr_file;
die "Can't find config file $config_file\n"   unless -e $config_file;
die "Can't find queue directory $queue_dir\n" unless -e $queue_dir;

# Make sure the script user can access the queue directory:
if ((! -r $queue_dir) || (! -x _) || (! -w _)) {
    die "$0: Aborting, can't access queue directory, $queue_dir!\n";
}

# Display status information:
chop($hostname = `hostname`);
chop($pwd = `pwd`);
print "This system: $hostname\nCurrent directory: $pwd\n";
print "Configuration file: $config_file\nAddress file: $addr_file\n";
print "Queue directory: $queue_dir\n\n\n";

# Trap keyboard-generated signals:
sub handler {
```

```
    local($sig) = @_;           # first argument is signal name
    print STDERR "Caught a SIG$sig--shutting down\n";
    unlink($tmpfile);
    exit(0);
}
$SIG{’INT’} = ’handler’;
$SIG{’QUIT’} = ’handler’;

# REPORT FORMAT SPECIFICATIONS:
# Override default screen length with LINES environment variable.
($= = $ENV{"LINES"}) if defined($ENV{"LINES"});

# Current line count must be decremented because
# print() already has been used to display five lines:
$- = -5;

format MAILERDEFINITIONS_TOP =
Delivery   Input                      Dest            Dest
Agent      Address                    Host            User
------------------------------------------------------------------------
.

format MAILERDEFINITIONS =
@<<<<<<<<< @<<<<<<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<
$mailer,   $input,                    $dest_host,     $dest_user
.

format ADDRESSREWRITING_TOP =
Delivery   Input                      Output
Agent      Address                    Address
------------------------------------------------------------------------
.

format ADDRESSREWRITING =
@<<<<<<<<< @<<<<<<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$mailer,   $input,                    $output
.

# SUBROUTINE DEFINITIONS:
# Parse output of sendmail name resolution

sub parseaddress
{
    local($mailer, $intfile) = @_;      # store intermediate-file name
    local($input, $output);         # no conflict with global var.

    open(INTFILE, $intfile) || die "Exiting, can’t open $intfile\n";

    $~ = ’ADDRESSREWRITING’;
    while () {
            if (/^ADDRESS TEST MODE|^Enter /) { next; } # skip
            chop;
            if (/^>/) {                         # lines of interest
                if ($output) {                  #  contains output addr
                    $output =~ s/^.*returns:  *//; # remove prefix
                    $output =~ s/[ "]//g;       # no spaces, no quotes
                    write;                      # mailer: input=>output
                }
                s/>.*input: *//; s/[ "]//g;     # isolate address
                $input = $_;                    # save input address
            }
```

```
            $output = $_;                       # save previous line
    }
    close(INTFILE);
}

sub address_processing {
    ($type) = @_;                   # either "Sender" or "Recipient"
    $~ = 'STDOUT';                          # back to standard output
    printf("\n%s-address rewriting:\n", $type);
    $- = 0;                                 # force header display
    $^ = 'ADDRESSREWRITING_TOP';            # new header format
    write;                                  # write header
    foreach $mailer (keys %mailers) {       # for each mailer
        next if $mailer eq "error";         # ignore error mailer
        if (($recvrules{$mailer} == 0) || ($sendrules{$mailer} == 0)) {
            print STDERR "Ignoring $mailer: Can't locate rule set.\n";
            next;
        }
        open(SENDMAIL,
        "|/usr/lib/sendmail -bt -C$config_file -oQ$queue_dir >$tmpfile")
            || die "Can't run /usr/lib/sendmail program.\n";
         open(ADDRESSFILE, $addr_file) || die "Can't open $addr_file";
        while () {
            next if /^$/;                       # skip blank lines
            $address = $_;                      # save address
            print SENDMAIL "2,$recvrules{$mailer},4 $address\n"
                if $type eq "Recipient";
            print SENDMAIL "1,$sendrules{$mailer},4 $address\n"
                if $type eq "Sender";
        }
        close(SENDMAIL); close(ADDRESSFILE);    # close, saving data
        &parseaddress ($mailer, $tmpfile);      # display result
    }
}

# MAIN PROGRAM starts here:

# Prefix each line of address file with "0 " so
# sendmail will process the addresses with rule set zero:
open(ADDRESSFILE, $addr_file)
    || die "Can't open $addr_file for reading\n";
open(TMPFILE, ">$tmpfile")  || die "Can't open $tmpfile for writing\n";
while () {
    print TMPFILE "0 $_" unless $_ eq "\n";     # skip blank lines
}
close(ADDRESSFILE); close(TMPFILE);

open(SENDMAIL, # open to read from temp file containing "0 addr" lines.
    "/usr/lib/sendmail -bt -C$config_file -oQ$queue_dir < $tmpfile|") ||
    die "Can't exec /usr/lib/sendmail program...\n";

# Get the mailers used from the Rule 0 tests and populate the
# the mailers associative array keyed by mailer, value arbitrary
$- = 0;                                     # force header display
$^ = 'MAILERDEFINITIONS_TOP';               # header format
write;                                      # display header
$~ = 'MAILERDEFINITIONS';                   # data format
while () {                        # Get next line (into $_)
    chop;                                   # remove the newline from $_
    if (/^ADDRESS TEST MODE|^Enter /) { next; } # skip prompts
    if (/^>/) {                             # only process these lines
```

```
        if ($prevline) {                    # prev line had mailer defn.
            $prevline =~ s/^.*returns:  *//; # remove prefix text
            $prevline =~ s/[ "]//g;         # no spaces, no quotes
            # $prevline may contain: $#mailer$@dest_host$:dest_user
            @t = split(/\$/, $prevline);    # divide into fields at "$"
            $mailer = $dest_user = $dest_host = "XXX";  # defaults
            for ($i = 1; $i <= $#t; $i++) { # for all fields
                if ($t[$i] =~ /^#(.*)/) {       # was $#mailer
                    $mailer = $1;
                } elsif ($t[$i] =~ /^@(.*)/) {  # was $@dest_host
                    $dest_host = $1;
                } elsif ($t[$i] =~ /^:(.*)/) {  # was $:dest_user
                    $dest_user = $1;
                }
            }
            write;                          # display formatted report
            if ($mailer ne "XXX") {
                $mailers{$mailer} = 1;      # add to list of mailers
            }
        }   # no previous line means current line has input address
        s/>.*input: *//; s/[ "]//g;         # isolate address
        $input = $_;                        # save input address
    }   # end of if (/^>)
    $prevline = $_;                         # store previous line
}
close(SENDMAIL);

# Get delivery-agent specific sender and recipient rule-set numbers:
# Typical input line: Mether, P=[IPC], F=msDFMueCX, S=11, R=21, A=IPC $h
open(CONFIGFILE, "grep ^M $config_file|");  # mailer defn. lines
while ()
{
    ($mailer)   = /^M(\w+),.*$/;            # get mailer name
    ($sendrule) = /^.*S=(\d+),.*$/;         # get sender rule number
    ($recvrule) = /^.*R=(\d+),.*$/;         # and recipient rule number
    $sendrules{$mailer} = $sendrule;        # sender and recipient rules
    $recvrules{$mailer} = $recvrule;        # keyed by mailer name
}
close(CONFIGFILE);

# Display recipient-address processing:
&address_processing(Recipient);

# Display sender-address processing:
&address_processing(Sender);

unlink($tmpfile);                           # done so clean up
```

B. General command-line format for invoking `chkaddrs`:

chkaddrs [-a *addr-file*] [-c *config-file*] [-q *queuedir*]

C. Sample input address file:

```
beccat
beccat@yang
kolstad@bsdi.com
```

D. Report generated by `chkaddrs` on a ‘‘client’’ machine using the addresses shown in Part C:

```
# chkaddrs -a addressformats
This system: yin
Current directory: /usr/local/scripts
Configuration file: /etc/sendmail.cf
Address file: threeaddr
Queue directory: /usr/spool/mqueue


Delivery    Input                      Dest            Dest
Agent       Address                    Host            User
--------------------------------------------------------------------
local       beccat                     XXX             beccat
local       beccat@yang                XXX             beccat
ddn         kolstad@bsdi.com           bsdi.com        kolstad<@bsdi.com>

Recipient-address rewriting:
Delivery    Input                      Output
Agent       Address                    Address
--------------------------------------------------------------------
ddn         beccat                     beccat@magicats.org
ddn         beccat@yang                beccat@yang.magicats.org
ddn         kolstad@bsdi.com           kolstad@bsdi.com
local       beccat                     beccat
local       beccat@yang                beccat@yang
local       kolstad@bsdi.com           kolstad@bsdi.com

Sender-address rewriting:
Delivery    Input                      Output
Agent       Address                    Address
--------------------------------------------------------------------
ddn         beccat                     beccat@magicats.org
ddn         beccat@yang                beccat@yang.magicats.org
ddn         kolstad@bsdi.com           kolstad@bsdi.com
local       beccat                     beccat
local       beccat@yang                beccat@yang
local       kolstad@bsdi.com           kolstad@bsdi.com

# []
```

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com

Go To Content   Search

Last Modified: Tuesday, 22-Aug-95 16:18:51 PDT