ELECTRONIC NOTIFICATION HOWTO

You probably receive and send your ordinary email via the mail agent associated with your web browser or by logging into a University of Washington Unix system such as homer.u.washington.edu and running **pine**. There is another type of mail which is important for anyone who writes operational software and would like to have their programs or scripts send email to interested parties in case something extraordinary occurs. This document explains how to set up electronic notification email for a RedHat 8.0 Linux system and provides some examples of how to use it.

Table of Contents

A Few Examples **1**
System Administration **1**
Notifications from Cron **4**
Setting Up Pine to Read Your Notifications **4**

A Few Examples

Let us start with a simple example.

```
if ! mkdir dog; then
    echo "Could not make dog" | mail -s "Dog creation failed" alan
fi
```

This script attempts to create a new directory named `dog`. If it fails, a mail message is sent to the user with name alan. Because the recipient address is just a username, the mail will not be sent out over the Internet. Instead, it will simply be sent the the user's mail file on the local machine. If user alan has one or more shell windows open, the shell will notify him that he has received new mail the next time that he enters any command on the command-line. This is often all that is needed.

The script could just as well have sent the notification to the user's Internet email address:

```
if ! mkdir dog; then
    echo "Could not make dog" | mail -s "Dog creation failed" afw@u.washington.edu
fi
```

If the examples above were run from the command line, the email notification would normally not be required. On the other hand, the scripts could be run automatically at some predetermined time by the **cron** daemon.

In fact, it is sometimes useful to utilize **cron** simply to send reminder messages. For example, the following line inserted into a `crontab` table would generate a reminder to change the backup tape in the tape drive every Friday at 3:00 PM:

```
0 15 * * 5   echo "Change the backup tape now!" | mail -s "Change tape" alan
```

System Administration

The configuration of the **sendmail** mail transfer agent is enormously complex. Fortunately this process has been made much simpler by the use of configuration macros. We will make one small change to the arguments to one of these macros.

Sendmail Configuration Change

First we need to install the files that will let us change the **sendmail** configuration. We will download the package and use the **rpm** command to install it. As an ordinary user, download the package from whitechuck:

```
cd /tmp
scp 'whitechuck:/opt/download/sendmail-cf*' .
```

Then become the root user, provide the password when asked, and run the **rpm** command:

```
su -
cd /tmp
rpm -Uhv sendmail-cf*
```

For the remainder of this configuration process, we will continue to need root root privilege.

The principal **sendmail** configuration file is called `sendmail.cf`. The corresponding macro file is `sendmail.mc`. We will make one small change to `sendmail.mc` and then use the **m4** macro processor to create a new `sendmail.cf`.

Before changing `sendmail.mc`, we should make a backup copy. Rather than simply copying it to some file with a slightly different name, we will place it in a local RCS archive:

```
cd /etc/mail
ci -l sendmail.mc
```

Next we will use our favorite text editor to change one line in `sendmail.mc`.

Replace the line

```
dnl define('SMART_HOST','smtp.your.provider')
```

with the line

```
dnl define('SMART_HOST','smtp.washington.edu')
```

If we then run the command

```
rcsdiff sendmail.mc
```

we should see, as confirmation, the output

```
RCS file: sendmail.mc,v
retrieving revision 1.1
diff -r1.1 sendmail.mc
17c17
< dnl define('SMART_HOST','smtp.your.provider')
```

```
---
> dnl define('SMART_HOST','smtp.washington.edu')
```

To complete the change to the configuration, we run **m4** to generate a new `sendmail.cf`:

```
m4 sendmail.mc > sendmail.cf
```

Restarting the Sendmail Daemon

The **sendmail** daemon must be restarted (or started, if not currently running) in order to utilize the new configuration):

```
/etc/rc.d/init.d/sendmail restart
```

At this point it, we will also insure that **sendmail** is started automatically, each time the system reboots:

```
chkconfig --level 2345 sendmail on
```

TCP Wrappers Configuration

TCP Wrappers is one of several layers of security which protect a Linux computer from malevolent attack via the Internet. In order to use **sendmail** for sending notifications, we have to open up security so that at least our own workstation can receive messages from itself.

On the Internet, the computer that we are logged into has several aliases:

- An alias host name: localhost
- An alias IP address: 127.0.0.1.

We will now add the alias IP address to the list of hosts that are allowed to communicate with our machine. First we make a copy of the file that we are about to change, by putting it under the control of RCS:

```
cd /etc
ci -l hosts.allow
```

Using our favorite text editor, we add the following line to the file `/etc/hosts.allow`

```
ALL: 127.0.0.1
```

then use the **rcsdiff** command to verify the result.

Testing the Results

Now we should be able to test the results. Both as the root user and as an ordinary user try sending messages using the following patterns:

```
echo "a message" | mail -s "a subject line" user
echo "a message" | mail -s "a subject line" user@host
```

Notifications from Cron

The **cron** batch scheduling daemon does not run from your desktop and is not attached to any terminal or shell window. If you have **cron** run a command which writes output to *standard output* or *standard error*, there will be no place for the output to go unless you add redirection to the command. Without redirection, **cron** will send the output to your email in-box. If you allow this to happen, your in-box will get very large, very quickly.

If you do not care to ever see the output of a command run from cron, simply redirect all of the output to the null file. The following example shows a way (no longer the best way) to set your computer's Internal clock from a time standard via the Internet. In this case, you do not care to receive email each night telling you that your clock has been adjusted. The command redirects standard output to `/dev/null` and then redirects file descriptor 2, which is standard error, onto file descriptor 1, which is standard input, so that both of these streams are taken care of. (If this seems a bit esoteric, you are right. Just remember the pattern. It works.)

```
30 22 * * * rdate -s bigben.cac.washington.edu > /dev/null 2>&1
```

You usually do not want your **cron** jobs to run in complete silence. This example will send you mail telling of success or failure.

Here is a script to use tar to backup the home directory of a user called alan:

```
if tar cf home.tgz /home/alan > /dev/null > 2>&1 ; then
    echo "Tar was successful" -s | mail -s "Tar succeeded" alan
else
    echo "Tar had problems"   -s | mail -s "Tar failed"    alan
fi
```

The following addition to `crontab` will cause **cron** to run the backup script, `/home/alan/bin/shell/backup`, once a week at 3:15 AM on Sunday:

```
15 3 * * 7 /home/alan/bin/shell/backup
```

Setting Up Pine to Read Your Notifications

You can use the Unix **mail** command to read your local notifications (those which were sent to your username rather than your Internet email address). You may find it more convenient, however to use **pine**.

To configure **pine**, start it up from a shell window as an ordinary user. Then follow this menu sequence:

```
S                           (setup)
C                           (config)
inbox-path
Enter                       (press the enter key)
/var/spool/mail/username
E                           (exit setup)
Yes                         (commit changes)
```

where *username* is your own username.