The Linux Tips HOWTO

Table of Contents

The Linux Tips HOWTO

Paul Anderson, paul@geeky1.ebtech.net

v3.6, June 1998

This HOWTO contains those hard to find hints and tweekings that make Linux a bit nicer.

1.Introduction

2.Short Tips

3.Detailed Tips

1.Introduction

Welcome to the **Linux Tips HOWTO**, a list of neato tricks and optimizations that make Linux more fun. All I have in here right now are tips off of the top of my head, and tips from the old Tips−HOWTO (Why take out decent tips, right?). So send all your favorite hints and tips to me so I can put them in the next Tips−HOWTO.

Paul Anderson *Maintainer−−Linux TIPS HOWTO*

`panderso@ebtech.net`

2.Short Tips

2.1 Handy Syslog Trick *Paul Anderson, Tips−HOWTO maintainer*

Edit your /etc/syslog.conf, and put in the following line:

```
# Dump everything on tty8
*.*                                     /dev/tty8
```

One caveat: *REMEMBER TO USE TABS!* syslog doesn't like spaces...

2.2 Script to view those compressed HOWTOs. *Didier Juges,*`dj@destin.nfds.net`

From a newbie to another, here is a short script that eases looking for and viewing howto documents. My howto's are in /usr/doc/faq/howto/ and are gzipped. The file names are XXX−HOWTO.gz, XXX being the subject. I created the following script called "howto" in the /usr/local/sbin directory:

```
#!/bin/sh
if [ "$1" = "" ]; then
    ls /usr/doc/faq/howto | less
else
    gunzip -c /usr/doc/faq/howto/$1-HOWTO.gz | less
fi
```

When called without argument, it displays a directory of the available howto's. Then when entered with the first part of the file name (before the hyphen) as an argument, it unzips (keeping the original intact) then displays the document.

For instance, to view the Serial−HOWTO.gz document, enter:

$ howto Serial

2.3 Is there enough free space??? *Hans Zoebelein,*`zocki@goldfish.cube.net`

Here comes a short script which will check from time to time that there is enough free space available on anything which shows up in mount (disks, cdrom, floppy...)

If space runs out, a message is printed every X seconds to the screen and 1 mail message per filled device is fired up.

```
#!/bin/sh

#
# $Id: check_hdspace,v 1.18 1996/12/11 22:33:29 root Exp root $
#

#
# Since I got mysterious error messages during compile when
# tmp files filled up my disks, I wrote this to get a warning
# before disks are full.
#
# If this stuff saved your servers from exploding,
# send praising email to zocki@goldfish.cube.net.
```

```
# If your site burns down because of this, sorry but I
# warned you: no comps.
# If you really know how to handle sed, please forgive me :)
#

#
# Shoot and forget: Put 'check_hdspace &' in rc.local.
# Checks for free space on devices every $SLEEPTIME sec.
# You even might check your floppies or tape drives. :)
# If free space is below $MINFREE (kb), it will echo a warning
# and send one mail for each triggering device to $MAIL_TO_ME.
# If there is more free space than trigger limit again,
# mail action is also armed again.
#

# TODO: Different $MINFREE for each device.
# Free /*tmp dirs securely from old junk stuff if no more free space.


DEVICES='/dev/sda2 /dev/sda8 /dev/sda9'         # device; your put disks here
MINFREE=20480                                   # kb; below this do warning
SLEEPTIME=10                                    # sec; sleep between checks
MAIL_TO_ME='root@localhost'                     # fool; to whom mail warning


# ------- no changes needed below this line (hopefully :) -------

MINMB=0
ISFREE=0
MAILED=""
let MINMB=$MINFREE/1024         # yep, we are strict :)

while [ 1 ]; do
        DF="`/bin/df`"
                for DEVICE in $DEVICES ; do
                ISFREE=`echo $DF | sed s#.\*$DEVICE" "\*[0-9]\*""\*[0-9]\*" "\*## | sed s#" ".\*##`

                if [ $ISFREE -le $MINFREE ] ; then
                        let ISMB=$ISFREE/1024
                        echo  "WARNING: $DEVICE only $ISMB mb free." >&2
                        #echo "more stuff here" >&2
                        echo -e "\a\a\a\a"

                        if [ -z  "`echo $MAILED | grep -w $DEVICE`" ] ; then
                                echo "WARNING: $DEVICE only $ISMB mb free.      (Trigger is set to $MINMB mb)" \
                                | mail -s "WARNING: $DEVICE only $ISMB mb free!" $MAIL_TO_ME
                                MAILEDH="$MAILED $DEVICE"
                                MAILED=$MAILEDH
                                # put further action here like cleaning
                                # up */tmp dirs...
                        fi
                        elif [ -n  "`echo $MAILED | grep -w $DEVICE`" ] ; then
                                # Remove mailed marker if enough disk space
                                # again. So we are ready for new mailing action.
                                MAILEDH="`echo $MAILED  | sed s#$DEVICE##`"
                                MAILED=$MAILEDH
                        fi

                done
                sleep $SLEEPTIME

done
```

2.4 Util to clean up your logfiles. *Paul Anderson, Tips−HOWTO Maintainer>*

If you're like me, you have a list with 430 subscribers, plus 100+ messages per day coming in over UUCP. Well, what's a hacker to do with these huge logs? Install chklogs, that's what. Chklogs is written by Emilio Grimaldo, `grimaldo@panama.iaehv.nl`, and the current version 1.8 available from ftp.iaehv.nl:/pub/users/grimaldo/chklogs−1.8.tar.gz. It's pretty self explanatory to install(you will, of course, check out the info in the doc subdirectory). Once you've got it installed, add a crontab entry like this:

```
# Run chklogs at 9:00PM daily.
00 21 * * *       /usr/local/sbin/chklogs -m
```

While you're at it, mention to the author how nice a peice of software this is:)

2.5 Handy Script to Clean Up Corefiles. *Otto Hammersmith,*`ohammers@cu-online.com`

Create a file called rmcores(the author calls it handle−cores) with the following in it:

```
#!/bin/sh
USAGE="$0 <directory> <message-file>"

if [ $# != 2 ] ; then
        echo $USAGE
        exit
fi

 echo Deleting...
find $1 -name core -atime 7 -print -type f -exec rm {} \;

echo e-mailing
for name in `find $1 -name core -exec ls -l {} \; | cut -c16-24`
do
        echo $name
        cat $2 | mail $name
done
```

And have a cron job run it every so often.

2.6 Moving directories between filesystems. *Alan Cox,*`A.Cox@swansea.ac.uk`

Quick way to move an entire tree of files from one disk to another

```
(cd /source/directory && tar cf - . ) | (cd /dest/directory && tar xvfp -)
```

[Change from cd /source/directory; tar....etc. to prevent possibility of trashing directory in case of disaster. Thanks to Jim Dennis, jim@starshine.org, for letting me know. −Maint.]

2.7 Finding out which directories are the largest. *Mick Ghazey,*`mick@lowdown.com`

Ever wondered which directories are the biggest on your computer? Here's how to find out.

```
du -S | sort -n
```

2.8 The Linux Gazette

Kudos go to John Fisk, creator of the Linux Gazette. This is an excellent e−zine plus, it's **FREE!!!** Now what more could you ask? Check it out at:

```
http://www.linuxgazette.com
```

BTW, It turns out that (1) LG is now out on a monthly basis, and (2) John Fisk no longer maintains it, the fellows at SSC do.

2.9 Pointer to patch for GNU Make 3.70 to change VPATH behavior.*Ted Stern,*`stern@amath.washington.edu`

I don't know if many people have this problem, but there is a "feature" of GNU make version 3.70 that I don't like. It is that VPATH acts funny if you give it an absolute pathname. There is an extremely solid patch that fixes this, which you can get from Paul D. Smith `<psmith@wellfleet.com>`. He also posts the documentation and patch after every revision of GNU make on the newsgroup "gnu.utils.bug" Generally, I apply this patch and recompile gmake on every system I have access to.

2.10 How do I stop my system from fscking on each reboot? *Dale Lutz,*`dal@wimsey.com`

Q: How do I stop e2fsck from checking my disk every time I boot up.

A: When you rebuild the kernel, the filesystem is marked as 'dirty' and so your disk will be checked with each boot. The fix is to run:

rdev –R /zImage 1

This fixes the kernel so that it is no longer convinced that the filesystem is dirty.

Note: If using lilo, then add `read-only`*to your linux setup in your lilo config file (Usually /etc/lilo.conf)*

2.11 How to avoid fscks caused by "device busy" at reboot time. *Jon Tombs,*`jon@gtex02.us.es`

If you often get device busy errors on shutdown that leave the filesystem in need of an fsck upon reboot, here is a simple fix:

To `/etc/rc.d/init.d/halt` or `/etc/rc.d/rc.0`, add the line

```
mount -o remount,ro /mount.dir
```

for all your mounted filesystems except /, before the call to umount –a. This means if, for some reason, shutdown fails to kill all processes and umount the disks they will still be clean on reboot. Saves a lot of time at reboot for me.

2.12 How to find the biggest files on your hard–drive.

Simon Amor,`simon@foobar.co.uk`

```
ls -l | sort +4n
```

Or, for those of you really scrunched for space this takes awhile but works great:

```
cd /
ls -lR | sort +4n
```

2.13 How to print pages with a margin for hole punching. *Mike Dickey,* `mdickey@thorplus.lib.purdue.edu`

```
        #!/bin/sh
        # /usr/local/bin/print
        # a simple formatted printout, to enable someone to
        # 3-hole punch the output and put it in a binder

        cat $1 | pr -t -o 5 -w 85 | lpr
```

2.14 A way to search through trees of files for a particular regular expression. *Raul Deluth Miller,* `rockwell@nova.umd.edu`

I call this script 'forall'. Use it like this:

```
forall /usr/include grep -i ioctl
forall /usr/man grep ioctl
```

Here's forall:

```
#!/bin/sh
if [ 1 = `expr 2 \> $#` ]
then
        echo Usage: $0 dir cmd [optargs]
        exit 1
fi
dir=$1
shift
find $dir -type f -print | xargs "$@"
```

2.15 A script for cleaning up after programs that create autosave and backup files. *Barry Tolnas,* `tolnas@nestor.engr.utk.edu`

Here is a simple two–liner which recursively descends a directory hierarchy removing emacs auto–save (#) and backup (~) files, .o files, and TeX .log files. It also compresses .tex files and README files. I call it 'squeeze' on my system.

```
#!/bin/sh
#SQUEEZE removes unnecessary files and compresses .tex and README files
#By Barry tolnas, tolnas@sun1.engr.utk.edu
#
echo squeezing $PWD
find  $PWD \( -name \*~ -or -name \*.o -or -name \*.log -or -name \*\#\) -exec
rm -f {} \;
find $PWD \( -name \*.tex -or -name \*README\* -or -name \*readme\* \) -exec gzip -9 {} \;
```

2.16 How to find out what process is eating the most memory. *Simon Amor,*`simon@foobar.co.uk`

```
ps -aux | sort +4n
```

–OR–

```
ps -aux | sort +5n
```

2.17 Rigging vi for C programming, *Paul Anderson,*`Tips-HOWTO Maintainer`

I do a lot of C programming in my spare time, and I've taken the time to rig vi to be C friendly. Here's my .exrc:

```
set autoindent
set shiftwidth=4
set backspace=2
set ruler
```

What does this do? autoindent causes vi to automatically indent each line following the first one indented, shiftwidth sets the distance of ^T to 4 spaces, backspace sets the backspace mode, and ruler makes it display the line number. Remember, to go to a specific line number, say 20, use:

```
vi +20 myfile.c
```

2.18 Using ctags to ease programming.

Most hackers already have ctags on their computers, but don't use it. It can be very handy for editing specific functions. Suppose you have a function, in one of many source files in a directory for a program you're writing, and you want to edit this function for updates. We'll call this function foo(). You don't where it is in the source file, either. This is where ctags comes in handy. When run, ctags produces a file named tags in the current dir, which is a listing of all the functions, which files they're in and where they are in said files. The tags file looks like this:

```
ActiveIconManager       iconmgr.c       /^void ActiveIconManager(active)$/
AddDefaultBindings      add_window.c    /^AddDefaultBindings ()$/
AddEndResize    resize.c        /^AddEndResize(tmp_win)$/
AddFuncButton   menus.c /^Bool AddFuncButton (num, cont, mods, func, menu, item)$/
AddFuncKey      menus.c /^Bool AddFuncKey (name, cont, mods, func, menu, win_name, action)$/
AddIconManager  iconmgr.c       /^WList *AddIconManager(tmp_win)$/
AddIconRegion   icons.c /^AddIconRegion(geom, grav1, grav2, stepx, stepy)$/
AddStartResize  resize.c        /^AddStartResize(tmp_win, x, y, w, h)$/
AddToClientsList        workmgr.c       /^void AddToClientsList (workspace, client)$/
AddToList       list.c  /^AddToList(list_head, name, ptr)$/
```

To edit, say AddEndResize() in vim, run:

```
vim -t AddEndResize
```

This will bring the appropriate file up in the editor, with the cursor located at the beginning of the function.

2.19 Why does sendmail hang for 5 minutes on startup with RedHat? *Paul Anderson*,`paul@geeky1.ebtech.net`

This is a fairly common problem, almost to the point of being a FAQ. I don't know if RedHat has fixed this bug in their distribution, but you can repair it yourself. If you look in your /etc/hosts file, you will find it looks something like:

```
127.0.0.1               localhost       yourbox
```

When sendmail starts, it does a lookup on your hostname(in this example, yourbox). It then finds that the IP for yourbox is 127.0.0.1, sendmail doesn't like this, so it does the lookup again. It continues with this for a while until it eventually gives up and exits. Fixing the problem is extremely easy, edit your /etc/hosts file and change it to something like this:

```
127.0.0.1            localhost
10.56.142.1          yourbox
```

2.20 How do I configure RedHat for using color−ls? *Paul Anderson,*`paul@geeky1.ebtech.net`

RedHat's distribution comes with color−ls, however why they don't configure it for colour use by default is beyond me. Here's to fix it.

First, type eval `DIRCOLORS`

Next, alias ls='ls −−color=auto'

And put the 'alias.....' in your /etc/bashrc

2.21 How do I find which library in /usr/lib holds a certain function? *Pawel Veselow,*`vps@unicorn.niimm.spb.su`

What if you're compiling and you've missed a library that needed linking in? All gcc reports are function names... Here's a simple command that'll find what you're looking for:

```
for i in *; do echo $i:;nm $i|grep tgetnum 2>/dev/null;done
```

Where tgetnum is the name of the function you're looking for.

2.22 I compiled a small test program in C, but when I run it, I get no output!

You probably compiled the program into a binary named test, didn't you? Linux has a program called test, which tests if a certain condition is true, it never produces any output on the screen. Instead of just typing test, try: ./test

3.Detailed Tips

3.1 Sharing swap partitions between Linux and Windows. *Tony Acero*,`ace3@midway.uchicago.edu`

1. Format the partition as a dos partition, and create the Windows swap file on it, but don't run windows yet. (You want to keep the swap file completely empty for now, so that it compresses well).
2. Boot linux and save the partition into a file. For example if the partition was /dev/hda8:

   ```
   dd if=/dev/hda8 of=/etc/dosswap
   ```
3. Compress the dosswap file; since it is virtually all 0's it will compress very well

   ```
   gzip -9 /etc/dosswap
   ```
4. Add the following to the /etc/rc file to prepare and install the swap space under Linux: *XXXXX is the number of blocks in the swap partition*

   ```
   mkswap /dev/hda8 XXXXX
   swapon -av
   ```

   Make sure you add an entry for the swap partition in your /etc/fstab file
5. If your init/reboot package supports /etc/brc or /sbin/brc add the following to /etc/brc, else do this by hand when you want to boot to dos|os/2 and you want to convert the swap partition back to the dos/windows version:

```
swapoff -av
zcat /etc/dosswap.gz | dd of=/dev/hda8 bs=1k count=100
```

Note that this only writes the first 100 blocks back to the partition. I've found empirically that this is sufficient

>> What are the pros and cons of doing this?

Pros: you save a substantial amount of disk space.

Cons: if step 5 is not automatic, you have to remember to do it by hand, and it slows the reboot process by a nanosecond :–)

3.2 Desperate Undelete. *Michael Hamilton*,`michael@actrix.gen.nz`

Here's a trick I've had to use a few times.

Desperate person's text file undelete.

If you accidentally remove a text file, for example, some email, or the results of a late night programming session, all may not be lost. If the file ever made it to disk, ie it was around for more than 30 seconds, its contents may still be in the disk partition.

You can use the grep command to search the raw disk partition for the contents of file.

For example, recently, I accidentally deleted a piece of email. So I immediately ceased any activity that could modify that partition: in this case I just refrained from saving any files or doing any compiles etc. On other occasions, I've actually gone to the trouble of bring the system down to single user mode, and unmounted the filesystem.

I then used the egrep command on the disk partition: in my case the email message was in /usr/local/home/michael/, so from the output from df, I could see this was in /dev/hdb5

```
sputnik3:~ % df
  Filesystem         1024-blocks  Used Available Capacity Mounted on
  /dev/hda3              18621    9759     7901     55%   /
  /dev/hdb3             308852  258443    34458     88%   /usr
  /dev/hdb5             466896  407062    35720     92%   /usr/local

  sputnik3:~ % su
  Password:
  [michael@sputnik3 michael]# egrep -50 'ftp.+COL' /dev/hdb5 > /tmp/x
```

Now I'm ultra careful when fooling around with disk partitions, so I paused to make sure I understood the command syntax BEFORE pressing return. In this case the email contained the word 'ftp' followed by some text followed by the word 'COL'. The message was about 20 lines long, so I used −50 to get all the lines around the phrase. In the past I've used −3000 to make sure I got all the lines of some source code. I directed the output from the egrep to a different disk partition − this prevented it from over writing the message I was looking for.

I then used strings to help me inspect the output

```
strings /tmp/x | less
```

Sure enough the email was in there.

This method can't be relied on, all, or some, of the disk space may have already been re−used.

This trick is probably only useful on single user systems. On multi−users systems with high disk activity, the space you free'ed up may have already been reused. And most of use can't just rip the box out from under our users when ever we need to recover a file.

On my home system this trick has come in handy on about three occasions in the past few years − usually when I accidentally trash some of the days work. If what I'm working survives to a point where I feel I made significant progress, it get's backed up onto floppy, so I haven't needed this trick very often.

3.3 How to use the immutable flag. *Jim Dennis,*`jadestar@rahul.net`

Use the Immutable Flag

Right after you install and configure your system go through the /bin, /sbin/, /usr/bin, /usr/sbin and /usr/lib (and a few of the other usual suspects and make liberal use of the 'chattr +i command'. Also add that to the the kernel files in root. Now 'mkdir /etc/.dist/' copy everything from /etc/ on down (I do this in two steps using /tmp/etcdist.tar to avoid recursion) into that directory. (Optionally you can just create /etc/.dist.tar.gz) −− and mark that as immutable.

The reason for all of this is to limit the damage that you can do when logged in as root. You won't overwrite files with a stray redirection operator, and you won't make the system unusable with a stray space in an 'rm −fr' command (you might still do alot of damage to your data −− but your libs and bins will be safer.

This also makes a variety of security and denial of service exploits either impossible or more difficult (since many of them rely on overwriting a file through the actions of some SUID program that *isn't providing an arbitrary shell command*).

The only inconvenience of this is when building and doing your 'make install' on various sorts of system binaries. On the other hand it also prevents the 'make install' from over−writing the files. When you forget to read the Makefile and chattr −i the files that are to be overwritten (and the directories to which you want to add files) −− the make fails, you just use the chattr command and rerun it. You can also take that opportunity to move your old bin's, libs, or whatever into a .old/ directory or rename or tar them or whatever.

3.4 A suggestion for where to put new stuff.*Jim Dennis,*`jadestar@rahul.net`

All new stuff starts under /usr/local! or /usr/local/`hostname`

If your distribution is one that leaves /usr/local empty then just create your /usr/local/src, /usr/local/bin etc and use that. If your distribution puts things in the /usr/local tree than you may want to 'mkdir /usr/local/`hostname`' and give the 'wheel' group +w to it (I also make it SUID and SGID to insure that each member of the wheel group can only mess with their own files thereunder, and that all files created will belong to the 'wheel' group.

Now discipline yourself to *ALWAYS! ALWAYS! ALWAYS!* put new packages under /usr/local/src/.from/$WHEREVER_I_GOT_IT/ (for the .tar or whatever files) and build them under

/usr/local/src (or .../$HOSTNAME/src). Make sure that it installs under the local hierarchy. If it *absolutely must* be installed back in /bin or /usr/bin or somewhere else −− put a symlink from the local heirarchy to each element that when anywhere else.

The reason for this −− even though it's more work −− is that it helps isolate what has to be backed up and restored or reinstalled in the event of a full re−install from the distribution medio (usually CD these days). By using a /usr/local/.from directory you also keep an informal log of where your sources are coming from −− which helps when you're looking for new updates −− and may be critical when monitoring the security announcement lists.

One of my systems at home (the one I'm calling from) was put together before I adopted these policies for myself. I still don't "know" all the ways that it differs from the stock "as installed" system. This is despite the fact that I've done very little with my home system's configuration and I'm the *only* person who ever uses it.

By contrast the systems I've set up at work (when I was thrust into the role of system administrator there) have all been configured this way −− have been administered by many contractors and other MIS people, and have had a large number of upgrades and package installations. Nonetheless I have a very good idea which precise elements were put in *after* the initial installation and configuration.

3.5 Converting all files in a directory to lowercase. *Justin Dossey,*`dossey@ou.edu`

I noticed a few overly difficult or unnecessary procedures recommended in the 2c tips section of Issue 12. Since there is more than one, I'm sending it to you:

```
#!/bin/sh
         # lowerit
         # convert all file names in the current directory to lower case
         # only operates on plain files--does not change the name of directories
         # will ask for verification before overwriting an existing file
         for x in `ls`
           do
           if [ ! -f $x ]; then
             continue
             fi
           lc=`echo $x  | tr '[A-Z]' '[a-z]'`
           if [ $lc != $x ]; then
             mv -i $x $lc
           fi
           done
```

Wow. That's a long script. I wouldn't write a script to do that; instead, I would use this command:

```
for i in * ; do [ -f $i ] && mv -i $i `echo $i | tr '[A-Z]' '[a-z]'`;
done;
```

on the command line.

The contributor says he wrote the script how he did for understandability (see below).

On the next tip, this one about adding and removing users, Geoff is doing fine until that last step. Reboot? Boy, I hope he doesn't reboot every time he removes a user. All you have to do is the first two steps. What sort of processes would that user have going, anyway? An irc bot? Killing the processes with a simple

```
kill -9 `ps -aux |grep ^<username> |tr -s " " |cut -d " " -f2`
```

Example, username is foo

```
kill -9 `ps -aux |grep ^foo |tr -s " " |cut -d " " -f2`
```

That taken care of, let us move to the forgotten root password.

The solution given in the Gazette is the most universal one, but not the easiest one. With both LILO and loadlin, one may provide the boot parameter "single" to boot directly into the default shell with no login or password prompt. From there, one may change or remove any passwords before typing "init 3" to start multiuser mode. Number of reboots: 1 The other way Number of reboots: 2

Justin Dossey

3.6 How To Upgrade Sendmail*Paul Anderson,*`paul@geeky1.ebtech.net`

We're starting from raw, clean source. First, obtain the sendmail source code. I've d/led version 8.9.0, which is, as you will notice, bleeding edge. I grabbed it from ftp.sendmail.org:/pub/sendmail/sendmail.8.9.0.tar.gz

It's about 1Meg, and considering I'm running 8.7.6, I think it's worth the effort. If this works, you'll undoubtedly hear about it, elsewise I can't get the new HOWTO versions out without e−mail:)

Now, once you've got the source d/led, unpack it. It'll create a dir called `sendmail-8.9.0` in the current directory. Change into that directory, read the README and RELEASE_NOTES files(and be amazed at the updates they've done). Now, cd in src. This is where most of your work will be done.

A quick note: Sendmail is a small, powerful and well−written program. The sendmail binary itself compiled in less than 5 minutes on my 5x86 133 with 32Megs RAM! The entire compile and install(sans config) took under 15 minutes!

I don't normally run BIND on my system, so I found the lines:

```
# ifndef NAMED_BIND
```

```
#  define NAMED_BIND    1       /* use Berkeley Internet Domain Server */
# endif
```

and changed the 1 to a 0, ala:

```
# ifndef NAMED_BIND
#  define NAMED_BIND    0       /* use Berkeley Internet Domain Server */
# endif
```

On Debian 1.3.1, db.h is by default installed in /usr/include/db, instead of /usr/include, where sendmail hopes to find it. Change to the src, mailstats, makemap, praliases, rmail and smrsh directories and execute the following command:

```
 ./Build -I/usr/include/db
```

Once you've done that, cd .. and type make install. There! Sendmail version 8.9.0 should now be installed! This is, of course, assuming you already have your original configuration. For everything to work smoothly on my system, since I host free mailing lists for people using majordomo, I had to add the following to the beginning of my /etc/sendmail.cf:

```
O DontBlameSendmail=forwardfileinunsafedirpath, forwardfileinunsafedirpathsafe
```

Sendmail 8.9.0 is rather pedantic about directory and file permissions these days, and will complain about dirs and files in aliases or .forward files that are group or world writeable. While it's not a good idea to disable this pedantry, I am only running with a single person at the console and I felt it was okay to allow this minor security hole. YMMV.

3.7 Some tips for new sysadmins.*Jim Dennis,*`jadestar@rahul.net`

Create and maintain a /README.`hostname` and/or a /etc/README.`hostname` *[Or possibly /usr/local/etc/README.`hostname` −Maint.]*

Absolutely, from *day one* of administering a system take notes in an online log file. You might make "vi /README.$(hostname)" a line in root's /bash_logout. Another way to do this is to write an su or a sudo script that does something like:

```
function exit \
        { unset exit; exit; \
          cat ~/tmp/session.$(date +%y%m%d) \
          >> /README.$(hostname) && \
          vi /README.$(hostname)
          }
script -a ~/tmp/session.$(date +%y%m%d)
/bin/su.org -
```

(use the typescript command to create a session log and create a function to automate appending and updating the log).

I'll admit that I haven't implemented this automation of policy –– I've just relied on self–discipline so far. However I have been toying with the idea (even to the point of prototyping the scripts and shell functions as you see them). One thing that holds me back on this is the 'script' command itself. I think I'll have to grab the sources and add a couple of command line parameters (to pause/stop the script recording from the command line) before I commit to using this).

My last suggestion (for this round):

Root's path should consist of 'PATH= /bin'

That's it. Nothing else on root's path. Everything root does is provided by a symlink from /bin or by an alias or shell function, or is a script or binary in /bin, or is typed out with an explicit path.

This makes anyone running as root aware (sometimes painfully so) of how he or she is trusting binaries. The wise admin of a multi–user host will periodically look through his or here /bin and /.*history files to look for patterns and loopholes.

The really motivated admin will spot sequences that can be automated, places where sanity checks can be inserted, and tasks for which "root" privileges should be temporarily eschewed (launching editors, MTA's and other large interactive programs with elaborate scripting features that *might* be embedded in transparent or data files –– like the infamous vi ./.exrc and emacs ./.emacs and the even more insidous $EXINIT and the embedded header/footer macros). Naturally those sorts of commands can be run with something like:

```
cp $data $some_users_home/tmp
su -c $origcommand $whatever_switches
cp $some_users_home/tmp $data
```

(...where the specifics depend on the command).

Mostly these last sorts of precautions are overboard for the home or "single" user workstation –– but they are very good policy the admin of a multi–user –– particular a publicly exposed system (like the one's at netcom).

3.8 How to configure xdm's chooser for host selection. *Arrigo Triulzi*,`a.triulzi@ic.ac.uk`

1. Edit the file that launches xdm most likely /etc/rc/rc.6 or /etc/rc.local) so that it contains the following lines in the xdm startup section.

   ```
   /usr/bin/X11/xdm
   exec /usr/bin/X11/X -indirect hostname
   ```

2. Edit /usr/lib/X11/xdm/Xservers and comment out the line which starts the server on the local machine (i.e. starting 0:)
3. Reboot the machine and you're home and away.

I add this because when I was, desperately, trying to set it up for my own subnet over here it took me about a week to suss out all the problems.

Caveat: with old SLS (1.1.1) for some reason you can leave a −nodaemon after the xdm line −− this does **NOT** work for later releases.