The Linux Tips HOWTO

Paul Anderson, panderso@ebtech.net v2.0, August 1996

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

Contents

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 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 -)
```

Changefromcd/source/directory;tar....etc.topreventpossibilityoftrashingdirectoryincaseofdisaster.ThankstoJimDennis, jadestar@rahul.net, forlettingmeknow.− Maint.

2.3 Finding out which directories are the largest. *Mick Ghazey,* `mghazey@miso.lowdown.com`

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

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

2.4 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.redhat.com/lg
```

While you're there, drop John Fisk a note telling him how wonderful an e-zine LG is.

2.5 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.

2.6 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)*

2.7 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/brc` or `/sbin/brc`, 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

2.8 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
```

2.9 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
```

2.10 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 $dir -type f -print | xargs "$@"
```

2.11 A script for cleaning up after programs that creat 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 {} \;
```

2.12 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
```

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 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.

3.3 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. 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.

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

Create and maintain a /README.‘hostname‘ and/or a /etc/README.‘hostname‘
Orpossibly/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).

3.5 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.