LINUX 101 HACKS

Practical Examples to Build a Strong Foundation in Linux

Ramesh Natarajan
www.thegeekstuff.com

Table of Contents

Introduction

> *"There are only 10 types of people in the world — those who understand binary, those who don't, and those who understand gray code"*
>
> ***— Geek***

There are total of 101 hacks in this book that will help you build a strong foundation in Linux. All the hacks in this book are explained with appropriate Linux command examples that are easy to follow.

This book contains 12 chapters. Hacks mentioned in 6 chapters are based on the articles that I've already posted on my blog. Hacks mentioned in rest of the 6 chapters are brand new.

I'm Ramesh Natarajan, author of The Geek Stuff blog and this eBook.

I have done intensive programming on several languages and C is my favorite. I have done lot of work on the infrastructure side including Linux system administration, DBA, Networking, Hardware and Storage (EMC).

I have also developed Password Dragon — free, easy and secure password manager that runs on Windows, Linux and Mac.

If you have any feedback about this eBook, please use this contact form to get in touch with me.

Foreword

Another collection of hacks? Yes! If you have just completed your first admin course or looking for better ways to get the job done the "Linux 101 Hacks" eBook is a good point to start. These useful tips are concise, well written and easy to read.

Well done - I will recommend this eBook to my students.

--Prof. Dr. Fritz Mehner, FH Südwestfalen, Germany

(Author of several Vim plugins, including bash-support vim plugin)

Version

Version	Date	Revisions
1.0	12-Feb-2009	First Edition

Download the latest version of the book here.

Chapter 1: Powerful CD Command Hacks

cd is one of the most frequently used commands during a UNIX session. The 6 cd command hacks mentioned in this chapter will boost your productivity instantly and make it easier to navigate the directory structure from command line.

Hack 1. Use CDPATH to define the base directory for cd command

If you are frequently performing cd to subdirectories of a specific parent directory, you can set the CDPATH to the parent directory and perform cd to the subdirectories without giving the parent directory path as explained below.

```
[ramesh@dev-db ~]# pwd
/home/ramesh

[ramesh@dev-db ~]# cd mail
-bash: cd: mail: No such file or directory

[Note: This is looking for mail directory under current
directory]

[ramesh@dev-db ~]# export CDPATH=/etc
[ramesh@dev-db ~]# cd mail
/etc/mail

[Note: This is looking for mail under /etc and not
under current directory]

[ramesh@dev-db /etc/mail]# pwd
/etc/mail
```

To make this change permanent, add export CDPATH=/etc to your ~/.bash_profile

Similar to the PATH variable, you can add more than one directory entry in the CDPATH variable, separating them with : , as shown below.

```
export CDPATH=.:~:/etc:/var
```

This hack can be very helpful under the following situations:

- Oracle DBAs frequently working under $ORACLE_HOME, can set the CDPATH variable to the oracle home
- Unix sysadmins frequently working under /etc, can set the CDPATH variable to /etc
- Developers frequently working under project directory /home/projects, can set the CDPATH variable to /home/projects
- End-users frequently accessing the subdirectories under their home directory, can set the CDPATH variable to ~ (home directory)

Hack 2. Use cd alias to navigate up the directory effectively

When you are navigating up a very long directory structure, you may be using cd ..\..\ with multiple ..\'s depending on how many directories you want to go up as shown below.

```
# mkdir -p
/tmp/very/long/directory/structure/that/is/too/deep

# cd /tmp/very/long/directory/structure/that/is/too/deep

# pwd
/tmp/very/long/directory/structure/that/is/too/deep

# cd ../../../../
```

```
# pwd
/tmp/very/long/directory/structure
```

Instead of executing cd ../../../.. to navigate four levels up, use one of the following three alias methods:

Method 1: Navigate up the directory using "..n"

In the example below, ..4 is used to go up 4 directory level, ..3 to go up 3 directory level, ..2 to go up 2 directory level. Add the following alias to your ~/.bash_profile and re-login.

```
alias ..="cd .."
alias ..2="cd ../.."
alias ..3="cd ../../.."
alias ..4="cd ../../../.."
alias ..5="cd ../../../../.."
```

```
# cd
/tmp/very/long/directory/structure/that/is/too/deep

# ..4
[Note: use ..4 to go up 4 directory level]

# pwd
/tmp/very/long/directory/structure/
```

Method 2: Navigate up the directory using only dots

In the example below, ... (five dots) is used to go up 4 directory level. Typing 5 dots to go up 4 directory structure is really easy to remember, as when you type the first two dots, you are thinking "going up one directory", after that every additional dot, is to go one level up. So, use .... (four dots) to go up 3 directory level and .. (two dots) to go up 1 directory level. Add the following alias to your ~/.bash_profile and re-login for the ..... (five dots) to work properly.

```
alias ..="cd .."
```

```
alias ...="cd ../.."
alias ....="cd ../../.."
alias .....="cd ../../../.."
alias ......="cd ../../../../.."
```

```
# cd /tmp/very/long/directory/structure/that/is/too/deep

# .....
[Note: use ..... (five dots) to go up 4 directory level]

# pwd
/tmp/very/long/directory/structure/
```

Method 3: Navigate up the directory using cd followed by consecutive dots

In the example below, cd….. (cd followed by five dots) is used to go up 4 directory level. Making it 5 dots to go up 4 directory structure is really easy to remember, as when you type the first two dots, you are thinking “going up one directory”, after that every additional dot, is to go one level up. So, use cd…. (cd followed by four dots) to go up 3 directory level and cd… (cd followed by three dots) to go up 2 directory level. Add the following alias to your ~/.bash_profile and re-login for the above cd….. (five dots) to work properly.

```
alias cd..="cd .."
alias cd...="cd ../.."
alias cd....="cd ../../.."
alias cd.....="cd ../../../.."
alias cd......="cd ../../../../.."
```

```
# cd /tmp/very/long/directory/structure/that/is/too/deep

# cd.....
[Note: use cd..... to go up 4 directory level]

# pwd
/tmp/very/long/directory/structure
```

Method 5: Navigate up the directory using cd followed by number

In the example below, cd4 (cd followed by number 4) is used to go up 4 directory level.

```
alias cd1="cd .."
alias cd2="cd ../.."
alias cd3="cd ../../.."
alias cd4="cd ../../../.."
alias cd5="cd ../../../../.."
```

Hack 3. Perform mkdir and cd using a single command

Sometimes when you create a new directory, you may cd to the new directory immediately to perform some work as shown below.

```
# mkdir -p /tmp/subdir1/subdir2/subdir3

# cd /tmp/subdir1/subdir2/subdir3

# pwd
/tmp/subdir1/subdir2/subdir3
```

Wouldn't it be nice to combine both mkdir and cd in a single command? Add the following to the .bash_profile and re-login.

```
$ vi .bash_profile

function mkdircd () { mkdir -p "$@" && eval cd
"\"\$$#\""; }
```

Now, perform both mkdir and cd at the same time using a single command as shown below:

```
# mkdircd /tmp/subdir1/subdir2/subdir3

[Note: This creates the directory and cd to it
automatically]

# pwd
/tmp/subdir1/subdir2/subdir3
```

Hack 4. Use “cd -” to toggle between the last two directories

You can toggle between the last two current directories using cd - as shown below.

```
# cd /tmp/very/long/directory/structure/that/is/too/deep

# cd /tmp/subdir1/subdir2/subdir3

# cd -

# pwd
/tmp/very/long/directory/structure/that/is/too/deep


# cd -

# pwd
/tmp/subdir1/subdir2/subdir3

# cd -

# pwd
/tmp/very/long/directory/structure/that/is/too/deep
```

Hack 5. Use dirs, pushd and popd to manipulate directory stack

You can use directory stack to push directories into it and later pop directory from the stack. Following three commands are used in this example.

- dirs: Display the directory stack
- pushd: Push directory into the stack
- popd: Pop directory from the stack and cd to it

Dirs will always print the current directory followed by the content of the stack. Even when the directory stack is empty, dirs command will still print only the current directory as shown below.

```
# popd
-bash: popd: directory stack empty

# dirs
~

# pwd
/home/ramesh
```

How to use pushd and popd? Let us first create some temporary directories and push them to the directory stack as shown below.

```
# mkdir /tmp/dir1
# mkdir /tmp/dir2
# mkdir /tmp/dir3
# mkdir /tmp/dir4

# cd /tmp/dir1
# pushd .

# cd /tmp/dir2
# pushd .

# cd /tmp/dir3
# pushd .

# cd /tmp/dir4
# pushd .
```

```
# dirs
/tmp/dir4 /tmp/dir4 /tmp/dir3 /tmp/dir2 /tmp/dir1

[Note: The first directory (/tmp/dir4) of the dir
command output is always the current directory and not
the content from the stack.]
```

At this stage, the directory stack contains the following directories:

```
/tmp/dir4
/tmp/dir3
/tmp/dir2
/tmp/dir1
```

The last directory that was pushed to the stack will be at the top. When you perform popd, it will cd to the top directory entry in the stack and remove it from the stack. As shown above, the last directory that was pushed into the stack is /tmp/dir4. So, when we do a popd, it will cd to the /tmp/dir4 and remove it from the directory stack as shown below.

```
# popd
# pwd
/tmp/dir4

[Note: After the above popd, directory Stack Contains:
/tmp/dir3
/tmp/dir2
/tmp/dir1]

# popd
# pwd
/tmp/dir3

[Note: After the above popd, directory Stack Contains:
/tmp/dir2
/tmp/dir1]

# popd
```

```
# pwd
/tmp/dir2

[Note: After the above popd, directory Stack Contains:
/tmp/dir1]

# popd
# pwd
/tmp/dir1

[Note: After the above popd, directory Stack is empty!]

# popd
-bash: popd: directory stack empty
```

Hack 6. Use "shopt -s cdspell" to automatically correct mistyped directory names on cd

Use shopt -s cdspell to correct the typos in the cd command automatically as shown below. If you are not good at typing and make lot of mistakes, this will be very helpful.

```
# cd /etc/mall
-bash: cd: /etc/mall: No such file or directory

# shopt -s cdspell

# cd /etc/mall

# pwd
/etc/mail

[Note: By mistake, when I typed mall instead of mail,
         cd corrected it automatically]
```

Chapter 2: Date Manipulation

Hack 7. Set System Date and Time

To change the system date use:

```
# date {mmddhhmiyyyy.ss}
```

- mm - Month
- dd - Date
- hh - 24 hour format
- mi - Minutes
- yyyy - Year
- ss - seconds

For example, to set system date to Jan 31st 2008, 10:19 p.m, 53 seconds

```
# date 013122192009.53
```

You can also change system date using set argument as shown below.

```
# date 013122192009.53

# date +%Y%m%d -s "20090131"

# date -s "01/31/2009 22:19:53"

# date -s "31 JAN 2009 22:19:53"

# date set="31 JAN 2009 22:19:53"
```

To set the time only:

```
# date +%T -s "22:19:53"
```

```
# date +%T%p -s "10:19:53PM"
```

Hack 8. Set Hardware Date and Time

Before setting the hardware date and time, make sure the OS date and time is set appropriately as shown in the hack#7.

Set the hardware date and time based on the system date as shown below:

```
# hwclock –systohc

# hwclock --systohc –utc
```

Use hwclock without any parameter, to view the current hardware date and time:

```
# hwclock
```

Check the clock file to verify whether the system is set for UTC:

```
# cat /etc/sysconfig/clock

ZONE="America/Los_Angeles"
UTC=false
ARC=false
```

Hack 9. Display Current Date and Time in a Specific Format

Following are different ways of displaying the current date and time in various formats:

```
$ date
Thu Jan  1 08:19:23 PST 2009

$ date --date="now"
Thu Jan  1 08:20:05 PST 2009

$ date --date="today"
Thu Jan  1 08:20:12 PST 2009

$ date --date='1970-01-01 00:00:01 UTC +5 hours' +%s
18001

$ date '+Current Date: %m/%d/%y%nCurrent Time:%H:%M:%S'
Current Date: 01/01/09
Current Time:08:21:41

$ date +"%d-%m-%Y"
01-01-2009

$ date +"%d/%m/%Y"
01/01/2009

$ date +"%A,%B %d %Y"
Thursday,January 01 2009
```

Following are the different format options you can pass to the date command:

- %D date (mm/dd/yy)
- %d day of month (01..31)
- %m month (01..12)
- %y last two digits of year (00..99)
- %a locale's abbreviated weekday name (Sun..Sat)
- %A locale's full weekday name, variable length (Sunday..Saturday)
- %b locale's abbreviated month name (Jan..Dec)

- %B locale's full month name, variable length (January..December)
- %H hour (00..23)
- %I hour (01..12)
- %Y year (1970...)

Hack 10. Display Past Date and Time

Following are various ways to display a past date and time:

```
$ date --date='3 seconds ago'
Thu Jan  1 08:27:00 PST 2009

$ date --date="1 day ago"
Wed Dec 31 08:27:13 PST 2008

$ date --date="1 days ago"
Wed Dec 31 08:27:18 PST 2008

$ date --date="1 month ago"
Mon Dec  1 08:27:23 PST 2008

$ date --date="1 year ago"
Tue Jan  1 08:27:28 PST 2008

$ date --date="yesterday"
Wed Dec 31 08:27:34 PST 2008

$ date --date="10 months 2 day ago"
Thu Feb 28 08:27:41 PST 2008
```

Hack 11. Display Future Date and Time

Following examples shows how to display a future date and time.

```
$ date
Thu Jan  1 08:30:07 PST 2009

$ date --date='3 seconds'
Thu Jan  1 08:30:12 PST 2009

$ date --date='4 hours'
Thu Jan  1 12:30:17 PST 2009

$ date --date='tomorrow'
Fri Jan  2 08:30:25 PST 2009

$ date --date="1 day"
Fri Jan  2 08:30:31 PST 2009

$ date --date="1 days"
Fri Jan  2 08:30:38 PST 2009

$ date --date="2 days"
Sat Jan  3 08:30:43 PST 2009

$ date --date='1 month'
Sun Feb  1 08:30:48 PST 2009

$ date --date='1 week'
Thu Jan  8 08:30:53 PST 2009

$ date --date="2 months"
Sun Mar  1 08:30:58 PST 2009

$ date --date="2 years"
Sat Jan  1 08:31:03 PST 2011

$ date --date="next day"
Fri Jan  2 08:31:10 PST 2009
```

```
$ date --date="-1 days ago"
Fri Jan  2 08:31:15 PST 2009

$ date --date="this Wednesday"
Wed Jan  7 00:00:00 PST 2009
```

Chapter 3: SSH Client Commands

Hack 12. Identify SSH Client Version

Sometimes it may be necessary to identify the SSH client that you are currently running and it's corresponding version number. Use ssh -V to identify the version number. Please note that Linux comes with OpenSSH.

The following example indicates that this particular system is using OpenSSH:

```
$ ssh -V
OpenSSH_3.9p1, OpenSSL 0.9.7a Feb 19 2003
```

The following example indicates that this particular system is using SSH2:

```
$ ssh -V
ssh: SSH Secure Shell 3.2.9.1 (non-commercial version)
on i686-pc-linux-gnu
```

Hack 13. Login to Remote Host using SSH

The First time when you login to a remotehost from a localhost, it will display the host key not found message and you can give "yes" to continue. The host key of the remote host will be added under .ssh2/hostkeys directory of your home directory, as shown below.

```
localhost$ ssh -l jsmith remotehost.example.com

Host key not found from database.
Key fingerprint:
xabie-dezbc-manud-bartd-satsy-limit-nexiu-jambl-title-jarde-
tuxum
You can get a public key's fingerprint by running
% ssh-keygen -F publickey.pub
```

```
on the keyfile.
Are you sure you want to continue connecting (yes/no)? Yes

Host key saved to
/home/jsmith/.ssh2/hostkeys/key_22_remotehost.example.com.pub
host key for remotehost.example.com, accepted by jsmith Mon
May 26 2008 16:06:50 -0700
jsmith@remotehost.example.com password:

remotehost.example.com$
```

The Second time when you login to the remote host from the localhost, it will prompt only for the password as the remote host key is already added to the known hosts list of the ssh client.

```
localhost$ ssh -l jsmith remotehost.example.com
jsmith@remotehost.example.com password:

remotehost.example.com$
```

For some reason, if the host key of the remote host is changed after you logged in for the first time, you may get a warning message as shown below. This could be because of various reasons such as:

- Sysadmin upgraded/reinstalled the SSH server on the remote host
- Someone is doing malicious activity etc.,

The best possible action to take before saying "yes" to the message below, is to call your sysadmin and identify why you got the host key changed message and verify whether it is the correct host key or not.

```
localhost$ ssh -l jsmith remotehost.example.com

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: HOST IDENTIFICATION HAS CHANGED!         @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-
```

```
middle attack)!
It is also possible that the host key has just been changed.
Please contact your system administrator.
Add correct host key to
"/home/jsmith/.ssh2/hostkeys/key_22_remotehost.example.com.pu
b"
to get rid of this message.
Received server key's fingerprint:
xabie-dezbc-manud-bartd-satsy-limit-nexiu-jambl-title-arde-
tuxum
You can get a public key's fingerprint by running
% ssh-keygen -F publickey.pub
on the keyfile.
Agent forwarding is disabled to avoid attacks by corrupted
servers.
Are you sure you want to continue connecting (yes/no)? yes

Do you want to change the host key on disk (yes/no)? yes

Agent forwarding re-enabled.
Host key saved to
/home/jsmith/.ssh2/hostkeys/key_22_remotehost.example.com.pub
host key for remotehost.example.com, accepted by jsmith Mon
May 26 2008 16:17:31 -0700

jsmith @remotehost.example.com's password:

remotehost$
```

Hack 14. Debug SSH Client Session

Sometimes it is necessary to view debug messages to troubleshoot any SSH connection issues. pass -v (lowercase v) option to the ssh as shown below to view the ssh debug messages.

Example without SSH client debug message:

```
localhost$ ssh -l jsmith remotehost.example.com

warning: Connecting to remotehost.example.com failed:
No address associated to the name
```

Example with SSH client debug message:

```
locaclhost$ ssh -v -l jsmith remotehost.example.com

debug:
SshConfig/sshconfig.c:2838/ssh2_parse_config_ext:
Metaconfig parsing stopped at line 3.

debug:
SshConfig/sshconfig.c:637/ssh_config_set_param_verbose:
Setting variable 'VerboseMode' to 'FALSE'.

debug:
SshConfig/sshconfig.c:3130/ssh_config_read_file_ext:
Read 17 params from config file.

debug: Ssh2/ssh2.c:1707/main: User config file not
found, using defaults. (Looked for
'/home/jsmith/.ssh2/ssh2_config')

debug: Connecting to remotehost.example.com, port 22…
(SOCKS not used)
warning: Connecting to remotehost.example.com failed:
No address associated to the name
```

Hack 15. Toggle SSH Session using SSH Escape Character

When you've logged on to the remotehost using ssh from the localhost, you may want to come back to the localhost to perform some activity and go back to remote host again. In this case, you don't need to disconnect the ssh session to the remote host. Instead, follow the steps below.

1. Login to remotehost from localhost:

```
localhost$ ssh -l jsmith remotehost
```

2. Now you are connected to the remotehost:

```
remotehost$
```

3. To come back to the localhost temporarily, type the escape character ~ and Control-Z.

When you type ~ you will not see that immediately on the screen until you press <Control-Z> and press enter. So, on the remotehost in a new line enter the following key strokes for the below to work: ~<Control-Z>

```
remotehost$ ~^Z
[1]+  Stopped  ssh -l jsmith remotehost

localhost$
```

4. Now you are back to the localhost and the ssh remotehost client session runs as a typical UNIX background job, which you can check as shown below:

```
localhost$ jobs
[1]+  Stopped  ssh -l jsmith remotehost
```

5. You can go back to the remote host ssh without entering the password again by bringing the background ssh remotehost session job to foreground on the localhost.

```
localhost$ fg %1
ssh -l jsmith remotehost

remotehost$
```

Hack 16. SSH Session Statistics using SSH Escape Character

To get some useful statistics about the current ssh session, do the following. This works only on SSH2 client.

1. Login to remotehost from localhost.

```
localhost$ ssh -l jsmith remotehost
```

2. On the remotehost, type ssh escape character ~ followed by s as shown below. This will display lot of useful statistics about the current SSH connection.

```
remotehost$  [Note: The ~s is not visible on the
command line when you type.]

        remote host: remotehost
        local host: localhost
        remote version: SSH-1.99-OpenSSH_3.9p1
        local version:  SSH-2.0-3.2.9.1 SSH Secure
Shell (non-commercial)
        compressed bytes in: 1506
        uncompressed bytes in: 1622
        compressed bytes out: 4997
        uncompressed bytes out: 5118
        packets in: 15
        packets out: 24
        rekeys: 0
        Algorithms:
        Chosen key exchange algorithm: diffie-hellman-
group1-sha1
        Chosen host key algorithm: ssh-dss
        Common host key algorithms: ssh-dss,ssh-rsa
        Algorithms client to server:
        Cipher: aes128-cbc
        MAC: hmac-sha1
        Compression: zlib
```

```
        Algorithms server to client:
        Cipher: aes128-cbc
        MAC: hmac-sha1
        Compression: zlib

localhost$
```

Additional SSH Info

On a side note, to setup SSH key based authentication, refer openSSH and SSH2 tutorials.

Chapter 4: Essential Linux Commands

Hack 17. Grep Command

grep command is used to search files for a specific text. This is incredibly powerful command with lot of options.

```
Syntax: grep [options] pattern [files]
```

How can I find all lines matching a specific keyword on a file?

In this example, grep looks for the text John inside /etc/passwd file and displays all the matching lines.

```
# grep John /etc/passwd

jsmith:x:1082:1082:John Smith:/home/jsmith:/bin/bash
jdoe:x:1083:1083:John Doe:/home/jdoe:/bin/bash
```

Option -v, will display all the lines except the match. In the example below, it displays all the records from /etc/password that doesn't match John.

Note: There are several lines in the /etc/password that doesn't contain the word John. Only the first line of the output is shown below.

```
# grep -v John /etc/passwd

jbourne:x:1084:1084:Jason Bourne:/home/jbourne:/bin/bash
```

How many lines matched the text pattern in a particular file?

In the example below, it displays the total number of lines that contains the text John in /etc/passwd file.

```
# grep -c John /etc/passwd
2
```

You can also get the total number of lines that did not match the specific pattern by passing option -cv.

```
# grep -cv John /etc/passwd
39
```

How to search a text by ignoring the case?

Pass the option -i (ignore case), which will ignore the case while searching.

```
# grep -i john /etc/passwd

jsmith:x:1082:1082:John Smith:/home/jsmith:/bin/bash
jdoe:x:1083:1083:John Doe:/home/jdoe:/bin/bash
```

How do I search all subdirectories for a text matching a specific pattern?

Use option -r (recursive) for this purpose. In the example below, it will search for the text "John" by ignoring the case inside all the subdirectories under /home/users.

This will display the output in the format of "filename: line that matching the pattern". You can also pass the option -l, which will display only the name of the file that matches the pattern.

```
# grep -ri john /home/users

/home/users/subdir1/letter.txt:John, Thanks for your
contribution.
/home/users/name_list.txt:John Smith
```

```
/home/users/name_list.txt:John Doe

# grep -ril john /root

/home/users/subdir1/letter.txt
/home/users/name_list.txt
```

Hack 18. Find Command

find is frequently used command to find files in the UNIX filesystem based on numerous conditions. Let us review some practice examples of find command.

```
Syntax: find [pathnames] [conditions]
```

How to find files containing a specific word in its name?

The following command looks for all the files under /etc directory with mail in the filename.

```
# find /etc -name "*mail*"
```

How to find all the files greater than certain size?

The following command will list all the files in the system greater than 100MB.

```
# find / -type f -size +100M
```

How to find files that are not modified in the last x number of days?

The following command will list all the files that were modified more than 60 days ago under the current directory.

```
# find . -mtime +60
```

How to find files that are modified in the last x number of days?

The following command will list all the files that were modified in the last two days under the current directory.

```
# find . –mtime -2
```

How to delete all the archive files with extension *.tar.gz and greater than 100MB?

Please be careful while executing the following command as you don't want to delete the files by mistake. The best practice is to execute the same command with ls -l to make sure you know which files will get deleted when you execute the command with rm.

```
# find / -type f -name *.tar.gz -size +100M -exec ls -l {} \;

# find / -type f -name *.tar.gz -size +100M -exec rm -f {} \;
```

How to archive all the files that are not modified in the last x number of days?

The following command finds all the files not modified in the last 60 days under /home/jsmith directory and creates an archive files under /tmp in the format of ddmmyyyy_archive.tar.

```
# find /home/jsmith -type f -mtime +60 | xargs tar -cvf
/tmp/`date '+%d%m%Y'_archive.tar`
```

On a side note, you can perform lot of file related activities (including finding files) using midnight commander GUI, a powerful text based file manager for Unix.

Hack 19. Suppress Standard Output and Error Message

Sometime while debugging a shell script, you may not want to see either the standard output or standard error message. Use /dev/null as shown below for suppressing the output.

Suppress standard output using > /dev/null

This will be very helpful when you are debugging shell scripts, where you don't want to display the echo statement and interested in only looking at the error messages.

```
# cat file.txt > /dev/null

# ./shell-script.sh > /dev/null
```

Suppress standard error using 2> /dev/null

This is also helpful when you are interested in viewing only the standard output and don't want to view the error messages.

```
# cat invalid-file-name.txt 2> /dev/null

# ./shell-script.sh 2> /dev/null
```

Hack 20. Join Command

Join command combines lines from two files based on a common field.

In the example below, we have two files - employee.txt and salary.txt. Both have employee-id as common field. So, we can use join command to combine

the data from these two files using employee-id as shown below.

```
$ cat employee.txt

100 Jason Smith
200 John Doe
300 Sanjay Gupta
400 Ashok Sharma

$ cat bonus.txt

100 $5,000
200 $500
300 $3,000
400 $1,250

$ join employee.txt bonus.txt

100 Jason Smith $5,000
200 John Doe $500
300 Sanjay Gupta $3,000
400 Ashok Sharma $1,250
```

Hack 21. Change the Case

Convert a file to all upper-case

```
$ cat employee.txt

100 Jason Smith
200 John Doe
300 Sanjay Gupta
400 Ashok Sharma

$ tr a-z A-Z < employee.txt

100 JASON SMITH
200 JOHN DOE
300 SANJAY GUPTA
```

```
400 ASHOK SHARMA
```

Convert a file to all lower-case

```
$ cat department.txt

100 FINANCE
200 MARKETING
300 PRODUCT DEVELOPMENT
400 SALES

$ tr A-Z a-z < department.txt

100 finance
200 marketing
300 product development
400 sales
```

Hack 22. Xargs Command

xargs is a very powerful command that takes output of a command and pass it as argument of another command. Following are some practical examples on how to use xargs effectively.

1. When you are trying to delete too many files using rm, you may get error message: /bin/rm Argument list too long - Linux. Use xargs to avoid this problem.

```
find ~ -name '*.log' -print0 | xargs -0 rm -f
```

2. Get a list of all the *.conf file under /etc/. There are different ways to get the same result. Following example is only to demonstrate the use of xargs. The output of the find command in this example is passed to the ls -l one by one using xargs.

```
# find /etc -name "*.conf" | xargs ls –l
```

3. If you have a file with list of URLs that you would like to download, you can use xargs as shown below.

```
# cat url-list.txt | xargs wget –c
```

4. Find out all the jpg images and archive it.

```
# find / -name *.jpg -type f -print | xargs tar -cvzf
images.tar.gz
```

5. Copy all the images to an external hard-drive.

```
# ls *.jpg | xargs -n1 -i cp {} /external-hard-
drive/directory
```

Hack 23. Sort Command

Sort command sorts the lines of a text file. Following are several practical examples on how to use the sort command based on the following sample text file that has employee information in the format:

```
employee_name:employee_id:department_name.
```

```
$ cat names.txt

Emma Thomas:100:Marketing
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

Sort a text file in ascending order

```
$ sort names.txt

Alex Jason:200:Sales
Emma Thomas:100:Marketing
Madison Randy:300:Product Development
Nisha Singh:500:Sales
Sanjay Gupta:400:Support
```

Sort a text file in descending order

```
$ sort -r names.txt

Sanjay Gupta:400:Support
Nisha Singh:500:Sales
Madison Randy:300:Product Development
Emma Thomas:100:Marketing
Alex Jason:200:Sales
```

Sort a colon delimited text file on 2nd field (employee_id)

```
$ sort -t: -k 2 names.txt

Emma Thomas:100:Marketing
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

Sort a tab delimited text file on 3rd field (department_name) and suppress duplicates

```
$ sort -t: -u -k 3 names.txt
```

```
Emma Thomas:100:Marketing
Madison Randy:300:Product Development
Alex Jason:200:Sales
Sanjay Gupta:400:Support
```

Sort the passwd file by the 3rd field (numeric userid)

```
$ sort -t: -k 3n /etc/passwd | more

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

Sort /etc/hosts file by ip-addres

```
$ sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n /etc/hosts

127.0.0.1 localhost.localdomain localhost
192.168.100.101 dev-db.thegeekstuff.com dev-db
192.168.100.102 prod-db.thegeekstuff.com prod-db
192.168.101.20  dev-web.thegeekstuff.com dev-web
192.168.101.21  prod-web.thegeekstuff.com prod-web
```

Combine sort with other commands

- **ps –ef | sort** : Sort the output of process list
- **ls -al | sort +4n** : List the files in the ascending order of the file-size. i.e sorted by 5th filed and displaying smallest files first.
- **ls -al | sort +4nr** : List the files in the descending order of the file-size. i.e sorted by 5th filed and displaying largest files first.

Hack 24. Uniq Command

Uniq command is mostly used in combination with sort command, as uniq removes duplicates only from a sorted file. i.e In order for uniq to work, all the duplicate entries should be in the adjacent lines. Following are some common examples.

1. When you have an employee file with duplicate entries, you can do the following to remove duplicates.

```
$ sort namesd.txt | uniq

$ sort –u namesd.txt
```

2. If you want to know how many lines are duplicates, do the following. The first field in the following examples indicates how many duplicates where found for that particular line. So, in this example the lines beginning with Alex and Emma were found twice in the namesd.txt file.

```
$ sort namesd.txt | uniq –c

      2 Alex Jason:200:Sales
      2 Emma Thomas:100:Marketing
      1 Madison Randy:300:Product Development
      1 Nisha Singh:500:Sales
      1 Sanjay Gupta:400:Support
```

3. The following displays only the entries that are duplicates.

```
$ sort namesd.txt | uniq –cd

      2 Alex Jason:200:Sales
      2 Emma Thomas:100:Marketing
```

Hack 25. Cut Command

Cut command can be used to display only specific columns from a text file or other command outputs.

Following are some of the examples.

Display the 1st field (employee name) from a colon delimited file

```
$ cut -d: -f 1 names.txt

Emma Thomas
Alex Jason
Madison Randy
Sanjay Gupta
Nisha Singh
```

Display 1st and 3rd field from a colon delimited file

```
$ cut -d: -f 1,3 names.txt

Emma Thomas:Marketing
Alex Jason:Sales
Madison Randy:Product Development
Sanjay Gupta:Support
Nisha Singh:Sales
```

Display only the first 8 characters of every line in a file

```
$ cut -c 1-8 names.txt

Emma Tho
Alex Jas
Madison
Sanjay G
Nisha Si
```

Misc Cut command examples

- **cut -d: -f1 /etc/passwd** Displays the unix login names for all the users in the system.
- **free | tr -s ' ' | sed '/^Mem/!d' | cut -d" " -f2** Displays the total memory available on the system.

Hack 26. Stat Command

Stat command can be used either to check the status/properties of a single file or the filesystem.

Display statistics of a file or directory.

```
$ stat /etc/my.cnf

  File: `/etc/my.cnf'
  Size: 346 Blocks: 16 IO Block: 4096   regular file
Device: 801h/2049d      Inode: 279856      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid:
(    0/    root)
Access: 2009-01-01 02:58:30.000000000 -0800
Modify: 2006-06-01 20:42:27.000000000 -0700
Change: 2007-02-02 14:17:27.000000000 -0800

$ stat /home/ramesh

  File: `/home/ramesh'
  Size: 4096            Blocks: 8          IO Block:
4096   directory
Device: 803h/2051d      Inode: 5521409     Links: 7
Access: (0755/drwxr-xr-x)  Uid: (  401/ramesh)   Gid: (
401/ramesh)
Access: 2009-01-01 12:17:42.000000000 -0800
Modify: 2009-01-01 12:07:33.000000000 -0800
Change: 2009-01-09 12:07:33.000000000 -0800
```

Display the status of the filesystem using option -f

```
$ stat -f /

  File: "/"
    ID: 0        Namelen: 255     Type: ext2/ext3
Blocks: Total: 2579457    Free: 2008027    Available:
1876998    Size: 4096
Inodes: Total: 1310720    Free: 1215892
```

Hack 27. Diff Command

diff command compares two different files and reports the difference. The output is very cryptic and not straight forward to read.

```
Syntax: diff [options] file1 file2
```

What was modified in my new file when compare to my old file?

The option -w in the diff command will ignore the white space while performing the comparison.

In the following diff output:

- The lines above ---, indicates the changes happened in first file in the diff command (i.e name_list.txt).
- The lines below ---, indicates the changes happened to the second file in the diff command (i.e name_list_new.txt). The lines that belong to the first file starts with < and the lines of second file starts with >.

```
# diff -w name_list.txt name_list_new.txt
```

```
2c2,3
< John Doe
---
> John M Doe
> Jason Bourne
```

Hack 28. Display total connect time of users

Ac command will display the statistics about the user's connect time.

Connect time for the current logged in user

With the option -d, it will break down the output for the individual days. In this example, I've been logged in to the system for more than 6 hours today. On Dec 1st, I was logged in for about 1 hour.

```
$ ac -d

Dec  1  total        1.08
Dec  2  total        0.99
Dec  3  total        3.39
Dec  4  total        4.50
Today   total        6.10
```

Connect time for all the users

To display connect time for all the users use -p as shown below. Please note that this indicates the cumulative connect time for the individual users.

```
$ ac -p

        john                                    3.64
        madison                               0.06
        sanjay                                  88.17
        nisha                                   105.92
```

```
        ramesh                                  111.42
        total 309.21
```

Connect time for a specific user

To get a connect time report for a specific user, execute the following:

```
$ ac -d sanjay

Jul  2  total       12.85
Aug 25  total        5.05
Sep  3  total        1.03
Sep  4  total        5.37
Dec 24  total        8.15
Dec 29  total        1.42
Today   total        2.95
```

Chapter 5: PS1, PS2, PS3, PS4 and PROMPT_COMMAND

Hack 29. PS1 - Default Interaction Prompt

The default interactive prompt on your Linux can be modified as shown below to something useful and informative. In the following example, the default PS1 was "\s-\v\$", which displays the shell name and the version number. Let us change this default behavior to display the username, hostname and current working directory name as shown below.

```
-bash-3.2$  export PS1="\u@\h \w> "

ramesh@dev-db ~> cd /etc/mail

ramesh@dev-db /etc/mail>

[Note: Prompt changed to "username@hostname current-
dir>" format]
```

Following PS1 codes are used in this example:

- \u - Username
- \h - Hostname
- \w - Full pathname of current directory. Please note that when you are in the home directory, this will display only ~ as shown above

Note that there is a space at the end in the value of PS1. Personally, I prefer a space at the end of the prompt for better readability.

Make this setting permanent by adding export PS1="\u@\h \w> " to either .bash_profile (or) .bashrc as shown below.

```
ramesh@dev-db ~> vi ~/.bash_profile

ramesh@dev-db ~> vi ~/.bashrc

[Note: Add export PS1="\u@\h \w> " to one of the above
files]
```

Refer to the next chapter for several practical examples of PS1 usage in detail.

Hack 30. PS2 - Continuation Interactive Prompt

A very long command can be broken down to multiple lines by giving \ at the end of the line. The default interactive prompt for a multi-line command is "> ". Let us change this default behavior to display "continue->" by using PS2 environment variable as shown below.

```
ramesh@dev-db ~> myisamchk --silent --force --fast --
update-state \
> --key_buffer_size=512M --sort_buffer_size=512M \
> --read_buffer_size=4M --write_buffer_size=4M \
> /var/lib/mysql/bugs/*.MYI

[Note: This uses the default ">" for continuation
prompt]

ramesh@dev-db ~> export PS2="continue-> "

ramesh@dev-db ~> myisamchk --silent --force --fast --
update-state \
continue-> --key_buffer_size=512M --
sort_buffer_size=512M \
continue-> --read_buffer_size=4M --write_buffer_size=4M
\
continue-> /var/lib/mysql/bugs/*.MYI

[Note: This uses the modified "continue-> " for
```

```
continuation prompt]
```

I found it very helpful and easy to read, when I break my long commands into multiple lines using \. I have also seen others who don't like to break-up long commands.

Hack 31. PS3 - Prompt used by "select" inside shell script

You can define a custom prompt for the select loop inside a shell script, using the PS3 environment variable, as explained below.

Shell script and output WITHOUT PS3:

```
ramesh@dev-db ~> cat ps3.sh
select i in mon tue wed exit
do
  case $i in
    mon) echo "Monday";;
    tue) echo "Tuesday";;
    wed) echo "Wednesday";;
    exit) exit;;
  esac
done

ramesh@dev-db ~> ./ps3.sh
1) mon
2) tue
3) wed
4) exit
#? 1
Monday
#? 4

[Note: This displays the default "#?" for select
command prompt]
```

Shell script and output WITH PS3:

```
ramesh@dev-db ~> cat ps3.sh
PS3="Select a day (1-4): "
select i in mon tue wed exit
do
  case $i in
    mon) echo "Monday";;
    tue) echo "Tuesday";;
    wed) echo "Wednesday";;
    exit) exit;;
  esac
done

ramesh@dev-db ~> ./ps3.sh
1) mon
2) tue
3) wed
4) exit
Select a day (1-4): 1
Monday
Select a day (1-4): 4

[Note: This displays the modified "Select a day (1-4):"
for select command prompt]
```

Hack 32. PS4 - Used by “set -x” to prefix tracing output

The PS4 shell variable defines the prompt that gets displayed, when you execute a shell script in debug mode as shown below.

Shell script and output WITHOUT PS4:

```
ramesh@dev-db ~> cat ps4.sh

set -x
echo "PS4 demo script"
ls -l /etc/ | wc -l
du -sh ~

ramesh@dev-db ~> ./ps4.sh
       ++ echo 'PS4 demo script'
PS4 demo script
++ ls -l /etc/
++ wc -l
243
++ du -sh /home/ramesh
48K     /home/ramesh

[Note: This displays the default "++" while tracing the
output using set -x]
```

Shell script and output WITH PS4:

The PS4 defined below in the ps4.sh has the following two codes:

- $0 - indicates the name of script
- $LINENO - displays the current line number within the script

```
ramesh@dev-db ~> cat ps4.sh

export PS4='$0.$LINENO+ '
set -x
echo "PS4 demo script"
ls -l /etc/ | wc -l
du -sh ~

ramesh@dev-db ~> ./ps4.sh
../ps4.sh.3+ echo 'PS4 demo script'
PS4 demo script
../ps4.sh.4+ ls -l /etc/
```

```
../ps4.sh.4+ wc -l
243
../ps4.sh.5+ du -sh /home/ramesh
48K     /home/ramesh

[Note: This displays the modified "{script-name}.{line-
number}+" while tracing the output using set -x]
```

Hack 33. PROMPT_COMMAND

Bash shell executes the content of the PROMPT_COMMAND just before displaying the PS1 variable.

```
ramesh@dev-db ~> export PROMPT_COMMAND="date +%k:%m:%S"

22:08:42
ramesh@dev-db ~>

[Note: This displays the PROMPT_COMMAND and PS1 output
on different lines]
```

If you want to display the value of PROMPT_COMMAND in the same line as the PS1, use the echo -n as shown below.

```
ramesh@dev-db ~> export PROMPT_COMMAND="echo -n [$(date
+%k:%m:%S)]"

[22:08:51]ramesh@dev-db ~>

[Note: This displays the PROMPT_COMMAND and PS1 output
on the same line]
```

Chapter 6: Colorful and Functional Shell Prompt Using PS1

Hack 34. Display username, hostname and basename of directory in the prompt

The PS1 in this example displays following three information in the prompt:

- \u - Username
- \h - Hostname
- \W - Base name of the current working directory

```
-bash-3.2$ export PS1="\u@\h \W> "

ramesh@dev-db ~> cd /etc/mail

ramesh@dev-db mail>
```

Hack 35. Display current time in the prompt

In the PS1 environment variable, you can directly execute any Linux command, by specifying in the format $(linux_command). In the following example, the command $(date) is executed to display the current time inside the prompt.

```
ramesh@dev-db ~> export PS1="\u@\h [\$(date
+%k:%m:%S)]> "

ramesh@dev-db [11:09:56]>
```

You can also use \t to display the current time in the hh:mm:ss format as shown below:

```
ramesh@dev-db ~> export PS1="\u@\h [\t]> "

ramesh@dev-db [12:42:55]>
```

You can also use \@ to display the current time in 12-hour am/pm format as shown below:

```
ramesh@dev-db ~> export PS1="[\@] \u@\h> "

[04:12 PM] ramesh@dev-db>
```

Hack 36. Display output of any command in the prompt

You can display output of any Linux command in the prompt. The following example displays three items separated by | (pipe) in the command prompt:

- \!: The history number of the command
- \h: hostname
- $kernel_version: The output of the uname -r command from $kernel_version variable
- \$?: Status of the last command

```
ramesh@dev-db ~> kernel_version=$(uname -r)

ramesh@dev-db ~> export PS1="\!|\h|$kernel_version|\$?> "

473|dev-db|2.6.25-14.fc9.i686|0>
```

Hack 37. Change foreground color of the prompt

Display prompt in blue color, along with username, host and current directory information

```
$ export PS1="\e[0;34m\u@\h \w> \e[m "

[Note: This is for light blue prompt]

$ export PS1="\e[1;34m\u@\h \w> \e[m "

[Note: This is for dark blue prompt]
```

- \e[- Indicates the beginning of color prompt
- x;ym - Indicates color code. Use the color code values mentioned below.
- \e[m - indicates the end of color prompt

Color Code Table:

```
Black 0;30
Blue 0;34
Green 0;32
Cyan 0;36
Red 0;31
Purple 0;35
Brown 0;33

[Note: Replace 0 with 1 for dark color]
```

Make the color change permanent by adding the following lines your ~/.bash_profile or ~/.bashrc

```
$ vi ~/.bash_profile
```

```
STARTCOLOR='\e[0;34m';
ENDCOLOR="\e[0m"
export PS1="$STARTCOLOR\u@\h \w> $ENDCOLOR"
```

Hack 38. Change background color of the prompt

Change the background color by specifying \e[{code}m in the PS1 prompt as shown below.

```
$ export PS1="\e[47m\u@\h \w> \e[m "

[Note: This is for Light Gray background]
```

Combination of background and foreground.

```
$ export PS1="\e[0;34m\e[47m\u@\h \w> \e[m "

[Note: This is for Light Blue foreground and Light Gray
background]
```

Add the following to your ~/.bash_profile or ~/.bashrc to make the above background and foreground color permanent.

```
$ vi ~/.bash_profile
STARTFGCOLOR='\e[0;34m';
STARTBGCOLOR="\e[47m"
ENDCOLOR="\e[0m"
export PS1="$STARTFGCOLOR$STARTBGCOLOR\u@\h \w>
$ENDCOLOR"
```

Play around by using the following background color and choose the one that match your taste:

- \e[40m

- \e[41m
- \e[42m
- \e[43m
- \e[44m
- \e[45m
- \e[46m
- \e[47m

Hack 39. Display multiple colors in the prompt

You can also display multiple colors in the same prompt. Add the following function to your ~/.bash_profile

```
function prompt {
  local BLUE="\[\033[0;34m\]"
  local DARK_BLUE="\[\033[1;34m\]"
  local RED="\[\033[0;31m\]"
  local DARK_RED="\[\033[1;31m\]"
  local NO_COLOR="\[\033[0m\]"
  case $TERM in
    xterm*|rxvt*)
      TITLEBAR='\[\033]0;\u@\h:\w\007\]'
      ;;
    *)
      TITLEBAR=""
      ;;
  esac
  PS1="\u@\h [\t]> "
  PS1="${TITLEBAR}\
  $BLUE\u@\h $RED[\t]>$NO_COLOR "
  PS2='continue-> '
  PS4='$0.$LINENO+ '
}
```

You can re-login for the changes to take effect or source the .bash_profile as shown below.

```
$. ./.bash_profile

$ prompt

ramesh@dev-db [13:02:13]>
```

Hack 40. Change the prompt color using tput

You can also change color of the PS1 prompt using tput as shown below:

```
$ export PS1="\[$(tput bold)$(tput setb 4)$(tput setaf
7)\]\u@\h:\w $ \[$(tput sgr0)\]“
```

tput Color Capabilities:

- tput setab [1-7] - Set a background color using ANSI escape
- tput setb [1-7] - Set a background color
- tput setaf [1-7] - Set a foreground color using ANSI escape
- tput setf [1-7] - Set a foreground color

tput Text Mode Capabilities:

- tput bold - Set bold mode
- tput dim - turn on half-bright mode
- tput smul - begin underline mode
- tput rmul - exit underline mode
- tput rev - Turn on reverse mode

- tput smso - Enter standout mode (bold on rxvt)
- tput rmso - Exit standout mode
- tput sgr0 - Turn off all attributes

Color Code for tput:

- 0 - Black
- 1 - Red
- 2 - Green
- 3 - Yellow
- 4 - Blue
- 5 - Magenta
- 6 - Cyan
- 7 - White

Hack 41. Create your own prompt using the available codes for PS1 variable

Use the following codes and create your own personal PS1 Linux prompt that is functional and suites your taste.

- \a an ASCII bell character (07)
- \d the date in "Weekday Month Date" format (e.g., "Tue May 26")
- \D{format} - the format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required
- \e an ASCII escape character (033)

- \h the hostname up to the first part
- \H the hostname
- \j the number of jobs currently managed by the shell
- \l the basename of the shell's terminal device name
- \n newline
- \r carriage return
- \s the name of the shell, the basename of $0 (the portion following the final slash)
- \t the current time in 24-hour HH:MM:SS format
- \T the current time in 12-hour HH:MM:SS format
- \@ the current time in 12-hour am/pm format
- \A the current time in 24-hour HH:MM format
- \u the username of the current user
- \v the version of bash (e.g., 2.00)
- \V the release of bash, version + patch level (e.g., 2.00.0)
- \w the current working directory, with $HOME abbreviated with a tilde
- \W the basename of the current working directory, with $HOME abbreviated with a tilde
- \! the history number of this command
- \# the command number of this command
- \$ if the effective UID is 0, a #, otherwise a $
- \nnn the character corresponding to the octal number nnn
- \\ a backslash
- \[begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt

- \] end a sequence of non-printing character

Hack 42. Use bash shell function inside PS1 variable

You can also invoke a bash shell function in the PS1 as shown below.

```
ramesh@dev-db ~> function httpdcount {
>  ps aux | grep httpd | grep -v grep | wc -l
> }

ramesh@dev-db ~> export PS1="\u@\h [`httpdcount`]> "

ramesh@dev-db [12]>

[Note: This displays the total number of running httpd
processes]
```

You can add the following line to your ~/.bash_profile or ~/.bashrc to make this change permanent:

```
$ vi .bash_profile
function httpdcount {
  ps aux | grep httpd | grep -v grep | wc -l
}
export PS1='\u@\h [`httpdcount`]> '
```

Hack 43. Use shell script inside PS1 variable

You can also invoke a shell script inside the PS1 variable. In the example below, the ~/bin/totalfilesize.sh, which calculates the total filesize of the current directory, is invoked inside the PS1 variable.

```
ramesh@dev-db ~> cat ~/bin/totalfilesize.sh
```

```
for filesize in $(ls -l . | grep "^-" | awk '{print
$5}')
do
  let totalsize=$totalsize+$filesize
done
echo -n "$totalsize"

ramesh@dev-db ~> export PATH=$PATH:~/bin

ramesh@dev-db ~> export PS1="\u@\h
[\$(totalfilesize.sh) bytes]> "

ramesh@dev-db [534 bytes]> cd /etc/mail

ramesh@dev-db [167997 bytes]>
```

[**Note**: This executes the totalfilesize.sh to display the total file size of the current directory in the PS1 prompt]

Chapter 7: Archive and Compression

Hack 44. Zip command basics

How to zip multiple files?

```
syntax: zip {.zip file-name} {file-names}
```

```
# zip var-log-files.zip /var/log/*
  adding: var/log/acpid (deflated 81%)
  adding: var/log/anaconda.log (deflated 79%)
  adding: var/log/anaconda.syslog (deflated 73%)
  adding: var/log/anaconda.xlog (deflated 82%)
  adding: var/log/audit/ (stored 0%)
  adding: var/log/boot.log (stored 0%)
  adding: var/log/boot.log.1 (deflated 40%)
  adding: var/log/boot.log.2 (deflated 42%)
  adding: var/log/boot.log.3 (deflated 40%)
  adding: var/log/boot.log.4 (deflated 40%)
```

How to zip a directory and it's files recursively?

```
# zip -r var-log-dir.zip /var/log/
updating: var/log/ (stored 0%)
  adding: var/log/wtmp (deflated 78%)
  adding: var/log/scrollkeeper.log (deflated 94%)
  adding: var/log/rpmpkgs.3 (deflated 68%)
  adding: var/log/spooler (stored 0%)
  adding: var/log/cron.2 (deflated 90%)
  adding: var/log/spooler.1 (stored 0%)
  adding: var/log/spooler.4 (stored 0%)
  adding: var/log/httpd/ (stored 0%)
  adding: var/log/rpmpkgs.1 (deflated 68%)
  adding: var/log/anaconda.log (deflated 79%)
  adding: var/log/secure.2 (deflated 93%)
```

How to unzip a *.zip compressed file?

```
# unzip var-log.zip
Archive:  var-log.zip
  inflating: var/log/acpid
  inflating: var/log/anaconda.log
  inflating: var/log/anaconda.syslog
  inflating: var/log/anaconda.xlog
   creating: var/log/audit/
```

To see a detailed output during unzip pass the -v option as shown below.

```
# unzip -v var-log.zip

Archive:  var-log.zip
 Length   Method    Size  Ratio   Date   Time   CRC-32
Name
--------  ------  ------- -----   ----   ----   ------    ---
-
    1916  Defl:N      369  81%  02-08-08 14:27  e2ffdc0c
var/log/acpid
   13546  Defl:N     2900  79%  02-02-07 14:25  34cc03a1
var/log/anaconda.log

skip..

    7680  Defl:N      411  95%  12-30-08 10:55  fe876ee9
var/log/wtmp.1
   40981  Defl:N     7395  82%  02-08-08 14:28  6386a95e
var/log/Xorg.0.log
--------          -------  ---                            ---
----
41406991          2809229  93%                            56
files
```

How to list a content of zip file with uncompressing it?

```
# unzip -l var-log.zip

Archive:  var-log.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
     1916  02-08-08 14:27   var/log/acpid
```

```
    13546  02-02-07 14:25   var/log/anaconda.log

..skip..

    40981  02-08-08 14:28   var/log/Xorg.0.log
    40981  02-08-07 14:56   var/log/Xorg.0.log.old
 --------                   -------
 41406991                   56 files
```

Hack 45. Advanced compression using zip command.

There are 10 levels of compression provided by zip command.

- Level 0 is the lowest level, where it just archives the file without any compression.
- Level 1 will perform little compression. But, will be very fast.
- Level 6 is the default level of compression.
- Level 9 is the maximum compression. This will be slower when compared to default level. In my opinion, unless you are compressing a huge file, you should always use level 9.

In the example below, I used Level 0, default Level 6, and Level 9 compression on a same directory. See the compressed file size yourself.

```
# zip var-log-files-default.zip /var/log/*

# zip -0 var-log-files-0.zip /var/log/*

# zip -9 var-log-files-9.zip /var/log/*

# ls -ltr
-rw-r--r--   1 root       root        2817248 Jan  1 13:05
var-log-files-default.zip
-rw-r--r--   1 root       root       41415301 Jan  1 13:05
var-log-files-0.zip
-rw-r--r--   1 root       root        2582610 Jan  1 13:06
var-log-files-9.zip
```

Hack 46. Password Protection of Zip files

Pass the option -P to the zip command to assign a password to the zip file.

```
# zip -P mysecurepwd var-log-protected.zip /var/log/*
```

The above option is good if you are using the command inside a shell-script for background jobs. However, when you are performing the compression interactively on the command-line, you don't want the password to be visible in the history. So, use the option -e as shown below to assign the password.

```
# zip -e var-log-protected.zip /var/log/*
Enter password:
Verify password:
updating: var/log/acpid (deflated 81%)
updating: var/log/anaconda.log (deflated 79%)
```

When you are uncompressing a password protected file, it will ask for the password as shown below.

```
# unzip  var-log-protected.zip
Archive:  var-log-protected.zip
[var-log-protected.zip] var/log/acpid password:
```

Hack 47. Validate a zip archive

Sometime you may want to validate a zip archive without extracting it. To test the validity of the zip file, pass option -t as shown below.

```
# unzip -t var-log.zip

     Archive:  var-log.zip
    testing: var/log/acpid            OK
    testing: var/log/anaconda.log     OK
```

```
    testing: var/log/anaconda.syslog   OK

skip...

    testing: var/log/wtmp              OK
    testing: var/log/wtmp.1            OK
    testing: var/log/Xorg.0.log        OK

No errors detected in compressed data of var-log.zip.
```

Hack 48. Tar Command Basics

tar command (tape archive) is used to convert a group of files into an archive.

```
Syntax: tar [options] [tar-archive-name] [other-file-
names]
```

How can I create a single backup file of all files and subdirectories under my home directory?

The following command creates a single archive backup file called my_home_directory.tar under /tmp. This archive will contain all the files and subdirectories under /home/jsmith.

- Option c, stands for create an archive.
- Option v stands for verbose mode, displays additional information while executing the command.
- Option f indicates the archive file name mentioned in the command.

```
# tar cvf /tmp/my_home_directory.tar /home/jsmith
```

How do I view all the files inside the tar archive?

Option t will display all the files from the tar archive.

```
# tar tvf /tmp/my_home_directory.tar
```

How do I extract all the files from a tar archive?

Option x will extract the files from the tar archive as shown below. This will extract the content to the current directory location from where the command is executed.

```
# tar xvf /tmp/my_home_directory.tar
```

How do I extract tar.gz files to a specific directory?

```
# tar xvfz /tmp/my_home_directory.tar.gz –C
/home/ramesh
```

Hack 49. Combine gzip, bzip2 with tar

How to use gzip with tar?

Add *option z* to the tar command when dealing with tar.gz compressed file.

```
# tar cvfz /tmp/my_home_directory.tar.gz /home/jsmith

# tar xvfz /tmp/my_home_directory.tar.gz

# tar tvfz /tmp/my_home_directory.tar.gz
```

Note: Using gzip is faster when compared to bzip2.

How to use bzip2 with tar?

Add *option j* to the tar command when dealing with tar.bz2 compressed file.

```
# tar cvfj /tmp/my_home_directory.tar.bz2 /home/jsmith

# tar xvfj /tmp/my_home_directory.tar.bz2

# tar tvfj /tmp/my_home_directory.tar.bz2
```

Note: Using bizp2 gives higher level of compression when compared to gzip.

Chapter 8: Command Line History

When you are using Linux command line frequently, using the history effectively can be a major productivity boost. In fact, once you have mastered the 15 examples that I've provided here, you'll find using command line more enjoyable and fun.

Hack 50. Display TIMESTAMP in history using HISTTIMEFORMAT

Typically when you type history from command line, it displays the command# and the command. For auditing purpose, it may be beneficial to display the timestamp along with the command as shown below.

```
# export HISTTIMEFORMAT='%F %T '

# history | more
      1  2008-08-05 19:02:39 service network restart
      2  2008-08-05 19:02:39 exit
      3  2008-08-05 19:02:39 id
      4  2008-08-05 19:02:39 cat /etc/redhat-release
```

Note: You can also setup the following alias to view the recent history commands.

```
alias h1='history 10'
alias h2='history 20'
alias h3='history 30'
```

Hack 51. Search the history using Control+R

I strongly believe that this may be your most frequently used feature of history. When you've already executed a very long command, you can simply

search history using a keyword and re-execute the same command without having to type it fully. Press Control+R and type the keyword.

In the following example, I searched for red, which displayed the previous command “cat /etc/redhat-release” in the history that contained the word red.

```
# [Note: Press Ctrl+R from the command prompt, which
will display the reverse-i-search prompt as shown
below]

(reverse-i-search)`red': cat /etc/redhat-release
[Note: Press enter when you see your command, which
will execute the command from the history]

# cat /etc/redhat-release
Fedora release 9 (Sulphur)
```

Sometimes you want to edit a command from history before executing it. For e.g. you can search for httpd, which will display service httpd stop from the command history, select this command and change the stop to start and re-execute it again as shown below.

```
# [Note: Press Ctrl+R from the command prompt, which
will display the reverse-i-search prompt]

(reverse-i-search)`httpd': service httpd stop
[Note: Press either left arrow or right arrow key when
you see your command, which will display the command
for you to edit, before executing it]

# service httpd start
```

Hack 52. Repeat previous command quickly using 4 different methods

Sometime you may end up repeating the previous commands for various reasons. Following are the 4 different ways to repeat the last executed command.

1. Use the **up arrow** to view the previous command and press enter to execute it.
2. **Type !!** and press enter from the command line
3. **Type !-1** and press enter from the command line.
4. **Press Control+P** will display the previous command, press enter to execute it

Hack 53. Execute a specific command from history

In the following example, If you want to repeat the command #4, execute !4 as shown below.

```
# history | more
       1  service network restart
       2  exit
       3  id
       4  cat /etc/redhat-release

# !4
cat /etc/redhat-release
Fedora release 9 (Sulphur)
```

Hack 54. Execute previous command that starts with a specific word

Type ! followed by the starting few letters of the command that you would like to re-execute. In the following example, typing !ps and enter, executed the previous command starting with ps, which is 'ps aux | grep yp'.

```
# !ps
ps aux | grep yp
       root     16947  0.0  0.1  36516  1264 ?
Sl   13:10   0:00 ypbind
       root     17503  0.0  0.0   4124   740 pts/0
S+   19:19   0:00 grep yp
```

Hack 55. Control the total number of lines in the history using HISTSIZE

Append the following two lines to the .bash_profile and relogin to the bash shell again to see the change. In this example, only 450 command will be stored in the bash history.

```
# vi ~/.bash_profile

HISTSIZE=450
HISTFILESIZE=450
```

Hack 56. Change the history file name using HISTFILE

By default, history is stored in ~/.bash_history file. Add the following line to the .bash_profile and relogin to the bash shell, to store the history command in .commandline_warrior file instead of .bash_history file. I'm yet to figure out a practical use for this. I can see this getting used when you want to track commands executed from different terminals using different history file name.

```
# vi ~/.bash_profile
HISTFILE=/root/.commandline_warrior
```

Hack 57. Eliminate the continuous repeated entry from history using HISTCONTROL

In the following example pwd was typed three times, when you do history, you can see all the 3 continuous occurrences of it. To eliminate duplicates, set HISTCONTROL to ignoredups as shown below.

```
# pwd

# pwd

# pwd

# history | tail -4
       44  pwd
       45  pwd
       46  pwd
       47  history | tail -4

[Note: There are three pwd commands in history, after
executing pwd 3 times as shown above]

# export HISTCONTROL=ignoredups

# pwd

# pwd

# pwd

# history | tail -3
       56  export HISTCONTROL=ignoredups
       57  pwd
       58  history | tail -4
```

```
[Note: There is only one pwd command in the history,
even after executing pwd 3 times as shown above]
```

Hack 58. Erase duplicates across the whole history using HISTCONTROL

The ignoredups shown above removes duplicates only if they are consecutive commands. To eliminate duplicates across the whole history, set the HISTCONTROL to erasedups as shown below.

```
# export HISTCONTROL=erasedups

# pwd

# service httpd stop

# history | tail -3
       38  pwd
       39  service httpd stop
       40  history | tail -3

# ls -ltr

# service httpd stop

# history | tail -6
       35  export HISTCONTROL=erasedups
       36  pwd
       37  history | tail -3
       38  ls –ltr
       39  service httpd stop
       40  history | tail -6

[Note: The previous service httpd stop after pwd got
erased]
```

Hack 59. Force history not to remember a particular command using HISTCONTROL

When you execute a command, you can instruct history to ignore the command by setting HISTCONTROL to ignorespace AND typing a space in front of the command as shown below. I can see lot of junior sysadmins getting excited about this, as they can hide a command from the history.

It is good to understand how ignorespace works. But, as a best practice, don't hide purposefully anything from history.

```
# export HISTCONTROL=ignorespace

# ls –ltr

# pwd

#  service httpd stop

[Note:  There is a space at the beginning of service,
to ignore this command from history]

# history | tail -3
       67  ls –ltr
       68  pwd
       69  history | tail -3
```

Hack 60. Clear all the previous history using option -c

Sometime you may want to clear all the previous history. However you may still want to keep the history moving forward.

```
# history -c
```

Hack 61. Substitute words from history commands

When you are searching through history, you may want to execute a different command but use the same parameter from the command that you've just searched.

In the example below, the !!:$ next to the vi command gets the argument from the previous command to the current command.

```
# ls anaconda-ks.cfg
anaconda-ks.cfg

# vi !!:$
vi anaconda-ks.cfg
```

In the example below, the !^ next to the vi command gets the first argument from the previous command (i.e cp command) to the current command (i.e vi command).

```
# cp anaconda-ks.cfg anaconda-ks.cfg.bak
anaconda-ks.cfg

# vi !^
vi anaconda-ks.cfg
```

Hack 62. Substitute a specific argument for a specific command

In the example below, !cp:2 searches for the previous command in history that starts with cp and takes the second argument of cp and substitutes it for the ls -l command as shown below.

```
# cp ~/longname.txt /really/a/very/long/path/long-
filename.txt

# ls -l !cp:2
ls -l /really/a/very/long/path/long-filename.txt
```

In the example below, !cp:$ searches for the previous command in history that starts with cp and takes the last argument (in this case, which is also the second argument as shown above) of cp and substitutes it for the ls -l command as shown below.

```
# ls -l !cp:$
ls -l /really/a/very/long/path/long-filename.txt
```

Hack 63. Disable the usage of history using HISTSIZE

If you want to disable history all together and don't want bash shell to remember the commands you've typed, set the HISTSIZE to 0 as shown below.

```
# export HISTSIZE=0

# history

# [Note:  History did not display anything]
```

Hack 64. Ignore specific commands from the history using HISTIGNORE

Sometimes you may not want to clutter your history with basic commands such as pwd and ls. Use HISTIGNORE to specify all the commands that you want to ignore from the history.

Please note that adding ls to the HISTIGNORE ignores only ls and not ls -l. So, you have to provide the exact command that you would like to ignore from the history.

```
# export HISTIGNORE="pwd:ls:ls -ltr:"

# pwd

# ls

# ls -ltr

# service httpd stop

# history | tail -3
       79  export HISTIGNORE="pwd:ls:ls -ltr:"
       80  service httpd stop
       81  history

[Note: History did not display pwd and ls]
```

Chapter 9: System Administration Tasks

Hack 65. Partition using fdisk

After you've installed brand new disks on your server, you have to use tools like fdisk to partition it accordingly.

Following are the 5 typical actions (commands) that you can execute inside fdisk.

- n - New Partition creation
- d - Delete an existing partition
- p - Print Partition Table
- w - Write the changes to the partition table. i.e save.
- q - Quit the fdisk utility

Create a partition

In the following example, I created a /dev/sda1 primary partition.

```
# fdisk /dev/sda

Device contains neither a valid DOS partition table,
nor Sun, SGI or OSF disklabel Building a new DOS
disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course,
the previous content won't be recoverable.

The number of cylinders for this disk is set to 34893.
There is nothing wrong with that, but this is larger
than 1024, and could in certain setups cause problems
with:
```

```
1) software that runs at boot time (e.g., old versions
of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will
be corrected by w(rite)

Command (m for help): p

Disk /dev/sda: 287.0 GB, 287005343744 bytes
255 heads, 63 sectors/track, 34893 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot  Start   End      Blocks   Id  System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-34893, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-34893,
default 34893):
Using default value 34893

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Verify that the partition got created successfully

```
# fdisk /dev/sda

The number of cylinders for this disk is set to 34893.
There is nothing wrong with that, but this is larger
than 1024, and could in certain setups cause problems
with:
```

```
1) software that runs at boot time (e.g., old versions
of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/sda: 287.0 GB, 287005343744 bytes
255 heads, 63 sectors/track, 34893 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot  Start         End      Blocks   Id  System
/dev/sda1        1       34893   280277991   83  Linux

Command (m for help): q
```

Hack 66. Format a partition using mke2fsk

After partitioning the disks, it is still not ready for usage, as we need to format the disk. At this stage, if you try to view the disk information, it will give the following error message indicating that no valid superblock is present.

```
# tune2fs -l /dev/sda1

tune2fs 1.35 (28-Feb-2004)
tune2fs: Bad magic number in super-block while trying
to open /dev/sda1

Couldn't find valid filesystem superblock.
```

To format the disk, use mke2fs as shown below.

```
# mke2fs /dev/sda1
```

You can also pass the following optional parameter to the mke2fs.

- -m 0 : reserved-blocks-percentage – This indicates the percentage of the filesystem blocks reserved for the root user. Default is 5%. In the following example, it is set to 0.
- -b 4096 : block-size specified in bytes. Valid values are 1024, 2048 and 4096 bytes per block.

```
# mke2fs -m 0 -b 4096 /dev/sda1

mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
205344 inodes, 70069497 blocks
0 blocks (0.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=71303168
2139 block groups
32768 blocks per group, 32768 fragments per group
96 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736,
1605632, 2654208, 4096000, 7962624, 11239424, 20480000,
23887872

Writing inode tables: done
Writing superblocks and filesystem accounting
information: done

This filesystem will be automatically checked every 32
mounts or 180 days, whichever comes first.  Use tune2fs
-c or -i to override.
```

The above command will create an ext2 filesystem. To create an ext3 file system do the following:

```
# mkfs.ext3 /dev/sda1

# mke2fs –j /dev/sda1
```

Hack 67. Mount the partition

After creating a partition and formatting, you can mount it to a mount point.

First create a directory where the partition should be mounted.

```
# mkdir /home/database
```

Mount the file system.

```
# mount /dev/sda1 /home/database
```

To automatically mount the filesystem after the reboot, add the following entry to the /etc/fstab

```
/dev/sdaa /home/database ext3 defaults 0 2
```

Hack 68. Fine tune the partition using tune2fs

Use the tune2fs -l /dev/sda1 to view the filesystem information as shown below.

```
# tune2fs -l /dev/sda1

tune2fs 1.35 (28-Feb-2004)
Filesystem volume name:   /home/database
Last mounted on:          <not available>
Filesystem UUID:          f1234556-e123-1234-abcd-
bbbbaaaaae11
Filesystem magic number:  0xEF44
Filesystem revision #:    1 (dynamic)
Filesystem features:      resize_inode filetype
sparse_super
Default mount options:    (none)
Filesystem state:         not clean
```

```
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              1094912
Block count:              140138994
Reserved block count:     0
Free blocks:              16848481
Free inodes:              1014969
First block:              0
Block size:               2048
Fragment size:            2048
Reserved GDT blocks:      512
Blocks per group:         16384
Fragments per group:      16384
Inodes per group:         128
Inode blocks per group:   8
Filesystem created:       Tue Jul  1 00:06:03 2008
Last mount time:          Thu Aug 21 05:58:25 2008
Last write time:          Fri Jan  2 15:40:36 2009
Mount count:              2
Maximum mount count:      20
Last checked:             Tue Jul  1 00:06:03 2008
Check interval:           15552000 (6 months)
Next check after:         Sat Dec 27 23:06:03 2008
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Default directory hash:   tea
Directory Hash Seed:      12345829-1236-4123-9aaa-
ccccc123292b
```

You can also use the tune2fs to tune the ex2/ext3 filesystem parameter. For example, if you want to change the Filesystem volume name, you can do it as shown below.

```
# tune2fs -l /dev/sda1 | grep volume
Filesystem volume name:   /home/database

# tune2fs -L database-home /dev/emcpowera1
tune2fs 1.35 (28-Feb-2004)
```

```
# tune2fs -l /dev/sda1 | grep volume
Filesystem volume name:   database-home
```

Hack 69. Create a swap file system.

Create a file for swap usage as shown below.

```
# dd if=/dev/zero of=/home/swap-fs bs=1M count=512
512+0 records in
512+0 records out

# ls -l /home/swap-fs
-rw-r--r--  1 root root 536870912 Jan  2 23:13
/home/swap-fs
```

Use mkswap to setup a Linux swap area in the /home/swap-fs file that was created above.

```
# mkswap /home/swap-fs

Setting up swapspace version 1, size = 536866 kB
```

Once the file is created and has been setup for Linux swap area, it is time to enable the swap using swapon as shown below.

```
# swapon /home/swap-fs
```

Add the following line to /etc/fstab and reboot the system for the swap to take into effect.

```
/home/swap-fs swap swap defaults 0 0
```

Hack 70. Create a new user

Add a new user – Basic method

Specify only the user name.

```
# useradd jsmith
```

Add a new user with additional Parameter

You can also specify the following parameter to the useradd

- -c : Description about the user.
- -e : expiry date of the user in mm/dd/yy format

```
# adduser -c "John Smith - Oracle Developer" -e
12/31/09 jsmith
```

Verify that the user got added successfully.

```
# grep jsmith /etc/passwd
jsmith:x:510:510:John Smith - Oracle
Developer:/home/jsmith:/bin/bash
```

Change the user password.

```
# passwd jsmith

Changing password for user jsmith.
New UNIX password:
BAD PASSWORD: it is based on a dictionary word
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Note: Make sure to follow these best practices to create a strong password for the user.

How to identify the default values used by useradd?

Following are the default values that will be used when an user is created.

```
# useradd -D

GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

Hack 71. Create a new group and assign to an user

Create a new developer group.

```
# groupadd developers
```

Validate that the group was created successfully.

```
# grep developer /etc/group
developers:x:511:
```

Add an user to an existing group.

You cannot use useradd to modify an existing user, as you'll get the following error message.

```
# useradd -G developers jsmith
useradd: user jsmith exists

# usermod -g developers jsmith
```

Validate the users group was modified successfully.

```
# grep jsmith /etc/passwd
jsmith:x:510:511:Oracle
Developer:/home/jsmith:/bin/bash

# id jsmith
uid=510(jsmith) gid=511(developers)
groups=511(developers)

# grep jsmith /etc/group
jsmith:x:510:
developers:x:511:jsmith
```

Hack 72. Setup SSH passwordless login in OpenSSH

You can login to a remote Linux server without entering password in 3 simple steps using ssky-keygen and ssh-copy-id as explained in this example.

ssh-keygen creates the public and private keys. ssh-copy-id copies the local-host's public key to the remote-host's authorized_keys file. ssh-copy-id also assigns proper permission to the remote-host's home, ~/.ssh, and ~/.ssh/authorized_keys.

Step 1: Create public and private keys using ssh-key-gen on local-host

```
jsmith@local-host$ ssh-keygen

Generating public/private rsa key pair.
```

```
Enter file in which to save the key
(/home/jsmith/.ssh/id_rsa):[Enter key]
Enter passphrase (empty for no passphrase): [Press
enter key]
Enter same passphrase again: [Pess enter key]
Your identification has been saved in
/home/jsmith/.ssh/id_rsa.
Your public key has been saved in
/home/jsmith/.ssh/id_rsa.pub.
The key fingerprint is:
33:b3:fe:af:95:95:18:11:31:d5:de:96:2f:f2:35:f9
jsmith@local-host
```

Step 2: Copy the public key to remote-host using ssh-copy-id

```
jsmith@local-host$ ssh-copy-id -i ~/.ssh/id_rsa.pub
remote-host

jsmith@remote-host's password:
Now try logging into the machine, with "ssh 'remote-
host'", and check in:
.ssh/authorized_keys to make sure we haven't added
extra keys that you weren't expecting.
```

Note: ssh-copy-id appends the keys to the remote-host's .ssh/authorized_key.

Step 3: Login to remote-host without entering the password

```
jsmith@local-host$ ssh remote-host

Last login: Sun Nov 16 17:22:33 2008 from 192.168.1.2

[Note: SSH did not ask for password.]

jsmith@remote-host$ [Note: You are on remote-host here]
```

Hack 73. Use ssh-copy-id along with ssh-agent

Using ssh-copy-id along with the ssh-add/ssh-agent

When no value is passed for the option -i and If ~/.ssh/identity.pub is not available, ssh-copy-id will display the following error message.

```
jsmith@local-host$ ssh-copy-id -i remote-host

/usr/bin/ssh-copy-id: ERROR: No identities found
```

If you have loaded keys to the ssh-agent using the ssh-add, then ssh-copy-id will get the keys from the ssh-agent to copy to the remote-host. i.e, it copies the keys provided by ssh-add -L command to the remote-host, when you don't pass option -i to the ssh-copy-id.

```
jsmith@local-host$ ssh-agent $SHELL

jsmith@local-host$ ssh-add -L
The agent has no identities.

jsmith@local-host$ ssh-add
Identity added: /home/jsmith/.ssh/id_rsa
(/home/jsmith/.ssh/id_rsa)

jsmith@local-host$ ssh-add -L
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAsJIEILxftj8aSxMa3d8t6JvM79D
aHrtPhTYpq7kIEMUNzApnyxsHpH1tQ/Ow==
/home/jsmith/.ssh/id_rsa

jsmith@local-host$ ssh-copy-id -i remote-host
jsmith@remote-host's password:
Now try logging into the machine, with "ssh 'remote-
host'", and check in: .ssh/authorized_keys to make sure
we haven't added extra keys that you weren't expecting.
[Note: This has added the key displayed by ssh-add -L]
```

Three Minor Annoyances of ssh-copy-id

Following are few minor annoyances of the ssh-copy-id.

1. Default public key: ssh-copy-id uses ~/.ssh/identity.pub as the default public key file (i.e when no value is passed to option -i). Instead, I wish it uses id_dsa.pub, or id_rsa.pub, or identity.pub as default keys. i.e If any one of them exist, it should copy that to the remote-host. If two or three of them exist, it should copy identity.pub as default.
2. The agent has no identities: When the ssh-agent is running and the ssh-add -L returns "The agent has no identities" (i.e no keys are added to the ssh-agent), the ssh-copy-id will still copy the message "The agent has no identities" to the remote-host's authorized_keys entry.
3. Duplicate entry in authorized_keys: I wish ssh-copy-id validates duplicate entry on the remote-host's authorized_keys. If you execute ssh-copy-id multiple times on the local-host, it will keep appending the same key on the remote-host's authorized_keys file without checking for duplicates. Even with duplicate entries everything works as expected. But, I would like to have my authorized_keys file clutter free.

Hack 74. Crontab

Using cron you can execute a shell-script or Linux commands at a specific time and date. For example a sysadmin can schedule a backup job that can run every day.

How to add a job to the cron?

```
# crontab –e
0 5 * * * /root/bin/backup.sh
```

This will execute /root/bin/backup.sh at 5 a.m every day.

Description of Cron fields.

Following is the format of the crontab file.

{minute} {hour} {day-of-month} {month} {day-of-week} {full-path-to-shell-script}

- minute: Allowed range 0 - 59
- hour: Allowed range 0 - 23
- day-of-month: Allowed range 0 - 31
- month: Allowed range 1 - 12. 1 = January. 12 = December.
- Day-of-week: Allowed range 0 - 7. Sunday is either 0 or 7.

Crontab examples

1. Run at 12:01 a.m. 1 minute after midnight everyday. This is a good time to run backup when the system is not under load.

```
1 0 * * * /root/bin/backup.sh
```

2. Run backup every weekday (Mon - Fri) at 11:59 p.m.

```
59 11 * * 1,2,3,4,5 /root/bin/backup.sh
```

Following will also do the same.

```
59 11 * * 1-5 /root/bin/backup.sh
```

3. Execute the command every 5 minutes.

```
*/5 * * * * /root/bin/check-status.sh
```

4. Execute at 1:10 p.m on 1st of every month

```
10 13 1 * * /root/bin/full-backup.sh
```

5. Execute 11 p.m on weekdays.

```
0 23 * * 1-5 /root/bin/incremental-backup.sh
```

Crontab Options

Following are the available options with crontab:

- crontab -e : Edit the crontab file. This will create a crontab, if it doesn't exist
- crontab -l : Display the crontab file.
- crontab -r : Remove the crontab file.
- crontab -ir : This will prompt user before deleting a crontab.

Hack 75. Safe Reboot Of Linux Using Magic SysRq Key

The magic SysRq key is a key combination in the Linux kernel which allows the user to perform various low level commands regardless of the system's state.

It is often used to recover from freezes, or to reboot a computer without corrupting the filesystem. The key combination consists of

Alt+SysRq+commandkey. In many systems the SysRq key is the printscreen key.

First, you need to enable the SysRq key, as shown below.

```
echo "1" > /proc/sys/kernel/sysrq
```

List of SysRq Command Keys

Following are the command keys available for Alt+SysRq+commandkey.

- 'k' – Kills all the process running on the current virtual console.
- 's' – This will attempt to sync all the mounted file system.
- 'b' - Immediately reboot the system, without unmounting partitions or syncing.
- 'e' – Sends SIGTERM to all process except init.
- 'm' – Output current memory information to the console.
- 'i' - Send the SIGKILL signal to all processes except init
- 'r' - Switch the keyboard from raw mode (the mode used by programs such as X11), to XLATE mode.
- 's' – sync all mounted file system.
- 't' - Output a list of current tasks and their information to the console.
- 'u' - Remount all mounted filesystems in readonly mode.
- 'o' – Shutdown the system immediately.
- 'p' – Print the current registers and flags to the console.
- '0-9′ - Sets the console log level, controlling which kernel messages will be printed to your console.

- ‘f’ - Will call oom_kill to kill process which takes more memory.
- ‘h’ – Used to display the help. But any other keys than the above listed will print help.

We can also do this by echoing the keys to the /proc/sysrq-trigger file. For example, to re-boot a system you can perform the following.

```
echo "b" > /proc/sysrq-trigger
```

Perform a Safe reboot of Linux using Magic SysRq Key

To perform a safe reboot of a Linux computer which hangs up, do the following. This will avoid the fsck during the next re-booting. i.e Press Alt+SysRq+letter highlighted below.

- un**R**aw (take control of keyboard back from X11,
- t**E**rminate (send SIGTERM to all processes, allowing them to terminate gracefully),
- **k**Ill (send SIGILL to all processes, forcing them to terminate immediately),
- **S**ync (flush data to disk),
- **U**nmount (remount all filesystems read-only),
- re**B**oot.

Chapter 10: Apachectl and Httpd Examples

After you have installed Apache2, if you want to use apachectl and httpd to it's maximum potential, you should go beyond using start, stop and restart. The 9 practical examples provided in this chapter will help you to use apachectl and httpd very effectively.

Apachectl acts as SysV init script, taking arguments like start, stop, restart and status. It also acts as front-end to httpd command, by simply passing the command line arguments to httpd. So, all the commands you execute using apachectl, can also be executed directly by calling httpd.

If you don't have Apache, refer to the tutorials: install apache from source or install LAMP stack using yum.

Hack 76. Pass different httpd.conf filename to apachectl

Typically you'll modify the original httpd.conf to try out different Apache directives. If something doesn't work out, you'll revert back the changes. Instead of playing around with the original httpd.conf, copy it to a new httpd.conf.debug and use this new httpd.conf.debug file with Apache for testing purpose as shown below using option -f.

```
# apachectl -f conf/httpd.conf.debug

# httpd -k start -f conf/httpd.conf.debug

[Note: you can use either apachectl or httpd as shown
above]

# ps -ef | grep http
root   25080     1  0 23:26 00:00:00 /usr/sbin/httpd -f
```

```
conf/httpd.conf.debug
apache 25099 25080  0 23:28 00:00:00 /usr/sbin/httpd -f
conf/httpd.conf.debug

[Note: ps shows the httpd running with httpd.conf.debug
file]
```

Once you are satisfied with the changes and Apache runs without any problem with httpd.conf.debug, you can copy the changes to httpd.conf and start the Apache normally as shown below.

```
# cp httpd.conf.debug httpd.conf

# apachectl stop

# apachectl start

# ps -ef | grep httpd
root     25114     1  0 23:28 00:00:00 /usr/sbin/httpd
-k start
daemon   25115 25114  0 23:28 00:00:00 /usr/sbin/httpd
-k start

[Note: ps indicates that the httpd is running using the
default config file]
```

Hack 77. Use a temporary DocumentRoot without modifying httpd.conf

This is very helpful, when you are trying out different layout for your website and don't want to modify the original files under the default DocumentRoot.

Take a copy of your original DocumentRoot directory (/var/www/html) to a new temporary DocumentRoot directory (/var/www/html_debug). Make all your changes under this temporary DocumentRoot directory (/var/www/html_debug) and start the Apache with this temporary directory as shown below using option -c.

```
# httpd -k start -c "DocumentRoot /var/www/html_debug/"
```

If you want to go back to original configuration using the default DocumentRoot (/var/www/html), simply restart the Apache as shown below.

```
# httpd -k stop
# apachectl start
```

Hack 78. Increase the Log Level temporarily

While you are debugging an issue, you can change the LogLevel of the Apache temporarily, without modifying the LogLevel directive in the httpd.conf as shown below using option -e. In this example, the LogLevel is set to debug.

```
# httpd -k start -e debug

[Sun Aug 17 13:53:06 2008] [debug] mod_so.c(246):
loaded module auth_basic_module
[Sun Aug 17 13:53:06 2008] [debug] mod_so.c(246):
loaded module auth_digest_module
```

Possible values you can pass to option -e are:

- debug
- info
- notice
- warn
- error
- crit
- alert
- emerg

Hack 79. Display the modules inside Apache

Display the modules compiled inside Apache

```
# httpd -l

Compiled in modules:
core.c
prefork.c
http_core.c
mod_so.c
```

Display both static and dynamic module loaded by Apache

When you pass option -l, to httpd, it will display only the static modules. Passing option -M, will display both static and shared modules as shown below.

```
# httpd -M

Loaded Modules:
core_module (static)
mpm_prefork_module (static)
http_module (static)
so_module (static)
auth_basic_module (shared)
auth_digest_module (shared)
authn_file_module (shared)
authn_alias_module (shared)
Syntax OK
```

Hack 80. Show all accepted directives inside httpd.conf

This is like an extended help for httpd, which will display all the httpd.conf directives and the places where they are valid. For a specific directive, it tells all the possible values and where it can be used inside the httpd.conf. This can be very helpful, when you want to quickly know about a particular Apache directive.

```
# httpd -L

HostnameLookups (core.c)
“on” to enable, “off” to disable reverse DNS lookups,
or “double” to enable double-reverse DNS lookups
Allowed in *.conf anywhere

ServerLimit (prefork.c)
Maximum value of MaxClients for this run of Apache
Allowed in *.conf only outside <Directory>, <Files> or
<Location>

KeepAlive (http_core.c)
Whether persistent connections should be On or Off
Allowed in *.conf only outside <Directory>, <Files> or
<Location>

LoadModule (mod_so.c)
a module name and the name of a shared object file to
load it from
Allowed in *.conf only outside <Directory>, <Files> or
<Location>
```

Hack 81. Validate the httpd.conf after making changes

Use option -t to validate whether there are any issues with a specific Apache configuration file. In the example shown below, it displays that there is a

problem at line 148 in the httpd.conf.debug. mod_auth_basicso is missing a . (period) before the so.

```
# httpd -t -f conf/httpd.conf.debug

httpd: Syntax error on line 148 of
/etc/httpd/conf/httpd.conf.debug:
Cannot load /etc/httpd/modules/mod_auth_basicso into
server:
/etc/httpd/modules/mod_auth_basicso: cannot open shared
object file: No such file or directory
```

Once you fix the issue, it will display Syntax OK.

```
# httpd -t -f conf/httpd.conf.debug

Syntax OK
```

Hack 82. Display the httpd build parameters

Use option -V (upper-case V), to display Apache version number and all the parameters that are used while building the Apache.

```
# httpd –V

Server version: Apache/2.2.9 (Unix)
Server built:   Jul 14 2008 15:36:56
Server's Module Magic Number: 20051115:15
Server loaded:  APR 1.2.12, APR-Util 1.2.12
Compiled using: APR 1.2.12, APR-Util 1.2.12
Architecture:   32-bit
Server MPM:     Prefork
threaded:     no
forked:     yes (variable process count)
Server compiled with….
-D APACHE_MPM_DIR="server/mpm/prefork"
-D APR_HAS_SENDFILE
-D APR_HAS_MMAP
```

```
-D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
-D APR_USE_SYSVSEM_SERIALIZE
-D APR_USE_PTHREAD_SERIALIZE
-D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
-D APR_HAS_OTHER_CHILD
-D AP_HAVE_RELIABLE_PIPED_LOGS
-D DYNAMIC_MODULE_LIMIT=128
-D HTTPD_ROOT="/etc/httpd"
-D SUEXEC_BIN="/usr/sbin/suexec"
-D DEFAULT_PIDLOG="logs/httpd.pid"
-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
-D DEFAULT_LOCKFILE="logs/accept.lock"
-D DEFAULT_ERRORLOG="logs/error_log"
-D AP_TYPES_CONFIG_FILE="conf/mime.types"
-D SERVER_CONFIG_FILE="conf/httpd.conf"
```

If you want display only the Apache version number, use the option -v (lower-case v) as shown below.

```
# httpd -v

Server version: Apache/2.2.9 (Unix)
Server built:   Jul 14 2008 15:36:56
```

Hack 83. Load a specific module only on demand

Sometimes you may not want to load all the modules in the Apache. For e.g. You may want to load ldap related modules to Apache, only when you are testing LDAP. This can be achieved as shown below.

Modify the httpd.conf and add IfDefine directive called load-ldap (you can name this anything you want).

```
<IfDefine load-ldap>
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module
```

```
modules/mod_authnz_ldap.so
</IfDefine>
```

When you are testing ldap and would like to Load the ldap related modules, pass the load-ldap to Option -D, as shown below:

```
# httpd -k start -e debug -Dload-ldap -f
/etc/httpd/conf/httpd.conf.debug

[Sun Aug 17 14:14:58 2008] [debug] mod_so.c(246):
loaded module ldap_module
[Sun Aug 17 14:14:58 2008] [debug] mod_so.c(246):
loaded module authnz_ldap_module
[Note: Pass -Dload-ldap, to load the ldap modules into
Apache]

# apachectl start

[Note: Start the Apache normally, if you don't want to
load the ldap modules.
```

Chapter 11: Bash Scripting

Hack 84. Execution Sequence of .bash_* files

What is the sequence in which the following files are executed?

- /etc/profile
- ~/.bash_profile
- ~/.bashrc
- ~/.bash_login
- ~/.profile
- ~/.bash_logout

Execution sequence for interactive login shell

Following pseudo code explains the sequence of execution of these files.

```
execute /etc/profile
IF ~/.bash_profile exists THEN
    execute ~/.bash_profile
ELSE
    IF ~/.bash_login exist THEN
        execute ~/.bash_login
    ELSE
        IF ~/.profile exist THEN
            execute ~/.profile
        END IF
    END IF
END IF
```

When you logout of the interactive shell, following is the sequence of execution:

```
IF ~/.bash_logout exists THEN
    execute ~/.bash_logout
END IF
```

Please note that /etc/bashrc is executed by ~/.bashrc as shown below:

```
# cat ~/.bashrc
if [ -f /etc/bashrc ]; then
. /etc/bashrc
Fi
```

Execution sequence for interactive non-login shell

While launching a non-login interactive shell, following is the sequence of execution:

```
IF ~/.bashrc exists THEN
    execute ~/.bashrc
END IF
```

Note: When a non-interactive shell starts up, it looks for ENV environment variable, and executes the file-name value mentioned in the ENV variable.

Hack 85. How to generate random number in bash shell

Use the $RANDOM bash built-in function to generate random number between 0 - 32767 as shown below.

```
$ echo $RANDOM
```

```
22543

$ echo $RANDOM
25387

$ echo $RANDOM
647
```

Hack 86. Debug a shell script

To debug a shell script use set -xv inside the shell script at the top.

Shell script with no debug command:

```
$ cat filesize.sh
#!/bin/bash
for filesize in $(ls -l . | grep "^-" | awk '{print
$5}')
do
  let totalsize=$totalsize+$filesize
done
echo "Total file size in current directory: $totalsize"
```

Output of Shell script with no debug command:

```
$ ./filesize.sh
Total file size in current directory: 652
```

Shell script with Debug command inside:

Add set -xv inside the shell script now to debug the output as shown below.

```
$ cat filesize.sh
#!/bin/bash
set -xv
for filesize in $(ls -l . | grep "^-" | awk '{print
```

```
$5}')
do
  let totalsize=$totalsize+$filesize
done
echo "Total file size in current directory: $totalsize"
```

Output of Shell script with Debug command inside:

```
$ ./fs.sh
++ ls -l .
++ grep '^-'
++ awk '{print $5}'
+ for filesize in '$(ls -l . | grep "^-" | awk
'\''{print $5}'\'')'
+ let totalsize=+178
+ for filesize in '$(ls -l . | grep "^-" | awk
'\''{print $5}'\'')'
+ let totalsize=178+285
+ for filesize in '$(ls -l . | grep "^-" | awk
'\''{print $5}'\'')'
+ let totalsize=463+189
+ echo 'Total file size in current directory: 652'
Total file size in current directory: 652
```

Execute Shell script with debug option:

Instead of giving the set -xv inside the shell script, you can also provide that while executing the shell script as shown below.

```
$ bash -xv filesize.sh
```

Hack 87. Quoting

echo statement without any special character.

```
$ echo The Geek Stuff
```

```
The Geek Stuff
```

Echo statement with a special character ; . semi-colon is a command terminator in bash. In the following example, “The Geek” works for the echo and “Stuff” is treated as a separate Linux command and gives command not found.

```
$ echo The Geek; Stuff
The Geek
-bash: Stuff: command not found
```

To avoid this you can add a \ in front of semi-colon, which will remove the special meaning of semi-colon and just print it as shown below.

```
$ echo The Geek\; Stuff
The Geek; Stuff
```

Single Quote

Use single quote when you want to literally print everything inside the single quote. Even the special variables such as $HOSTNAME will be print as $HOSTNAME instead of printing the name of the Linux host.

```
$ echo 'Hostname=$HOSTNAME ;  Current User=`whoami` ;
Message=\$ is USD'

Hostname=$HOSTNAME ;  Current User=`whoami` ;
Message=\$ is USD
```

Double Quote

Use double quotes when you want to display the real meaning of special variables.

```
$ echo "Hostname=$HOSTNAME ;  Current User=`whoami` ;
Message=\$ is USD"
```

```
Hostname=dev-db ;  Current User=ramesh ; Message=$ is
USD
```

Double quotes will remove the special meaning of all characters except the following:

- $ Parameter Substitution.
- ` Backquotes
- \$ Literal Dollar Sign.
- \´ Literal Backquote.
- \" Embedded Doublequote.
- \\ Embedded Backslashes.

Hack 88. Read data file fields inside a shell script

This example shows how to read a particular field from a data-file and manipulate it inside a shell-script. For example, let us assume the employees.txt file is in the format of {employee-name}:{employee-id}:{department-name}, with colon delimited file as shown below.

```
$ cat employees.txt
Emma Thomas:100:Marketing
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

The following shell script explains how to read specific fields from this employee.txt file.

```
$ vi read-employees.sh
#!/bin/bash
IFS=:
echo "Employee Names:"
echo "---------------"
while read name empid dept
```

```
do
  echo "$name is part of $dept department"
done < ~/employees.txt
```

Assign execute privilege to the shell script and execute it.

```
$ chmod u+x read-employees.sh

$ ./read-employees.sh
Employee Names:
---------------
Emma Thomas is part of Marketing department
Alex Jason is part of Sales department
Madison Randy is part of Product Development department
Sanjay Gupta is part of Support department
Nisha Singh is part of Sales department
```

Chapter 12: System Monitoring and Performance

Hack 89. Free command

free command displays all the necessary information about system physical (RAM) and swap memory.

```
Syntax: free [options]
```

What is the total RAM on my system?

In the example below, the total physical memory on this system is 1GB. The values displayed below are in KB.

```
# free
       total   used    free   shared  buffers  cached
Mem: 1034624   1006696 27928  0       174136   615892
-/+ buffers/cache:     216668      817956
Swap:    2031608       0    2031608
```

What is the total memory on my system including RAM and Swap?

In the following command:

- option m displays the values in MB
- option t displays the "Total" line, which is sum of physical and swap memory values
- option o is to hide the buffers/cache line from the above example.

free -mto

```
             total       used       free     shared    buffers     cached
Mem:          1010        983         27          0        170        601
Swap:         1983          0       1983
Total:        2994        983       2011
```

Hack 90. Top Command

top command displays real time information about various performance metrics of the system such as CPU Load, Memory Usage, Processes list etc.

```
Syntax: top [options]
```

How to view my current system status including CPU usage?

Execute top without any option from the command line, which will display the output shown below. The top command output will keep displaying the real-time values, until you press "Control + c" or q to exit from the command output.

top

```
top - 13:10:13 up 171 days, 20:21,  3 users,  load average: 0.01, 0.05, 0.00

Tasks: 194 total,   1 running, 193 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.6% us,  0.7% sy,  0.0% ni, 98.7% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   1034624k total,  1007420k used,    27204k free,   174540k buffers
Swap:  2031608k total,        0k used,  2031608k free,   615904k cached

PID   USER    PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11912 apache  15   0 31828  13m  3916 S    1    0.2   0:46.35  httpd
19299  oracle  19   0  279m  18m   17m S    1    0.2   0:00.03  oracle
11398 jsmith  16   0  107m  28m  6404 S    0    0.4   0:03.07  perl
```

How to read the output of the top command shown above?

- Line 1 “top”, indicates that the system has been up and running for 171 days.
- Line 2 “Tasks”, displays the total number of processes along with a breakdown of running, sleeping, stopped and zombie processes count.
- Line 3 “Cpu(s)” displays the current CPU utilization of the system. In this example, CPU is 98.7% idle
- Line 4 “Mem” and line 5 “Swap” provides the memory information. This is the same information from the free command.
- The rest of the lines display all the active processes on the system, sorted default by CPU usage (%CPU column). i.e the most CPU intensive processes will be displayed on the top by default.

There are several command line options and interactive options available for top commands. Let us review couple of essential options for top command.

How to identify the most memory intensive processes?

While the output of the top command displayed, press F, which will display the following message and show all fields available for sorting, press n (which is for sorting the processes by Memory) and press enter. This will display the processes in the top output sorted by memory usage.

```
Current Sort Field:  K  for window 1:Def
```

Select sort field via field letter, type any other key to return

How to add additional fields (for e.g. CPU Time) to the top output?

While the top command is running, press f, which will display the following message and show all fields available for display, press l, which will add the CPU Time to the display columns in the top output.

```
Current Fields:  AEHIOQTWKNMbcdfgjplrsuvyzX  for window 1:Def

Toggle fields via field letter, type any other key to return
```

How to get the full path name and parameters of the running processes?

While the top command is running, press c, which will display full pathname of running processes as shown below in the command column. i.e Instead of httpd, it displays /usr/local/apache2/bin/httpd.

```
PID   USER    PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND

11912 apache  15   0 31828  13m  3916 S    1    0.2   0:46.35
/usr/local/apache2/bin/httpd
```

How to view the individual CPUs in the top command?

While the top command is running, press 1 (number one), which will display the performance data of the individual CPUs on that machine as shown below.

```
top - 13:10:13 up 171 days, 20:21,  3 users,  load average: 0.01, 0.05, 0.00
Tasks: 194 total,   1 running, 193 sleeping,   0 stopped,   0 zombie
Cpu0  : 10.2% us,  2.6% sy,  0.0% ni, 86.8% id,  0.3% wa,  0.0% hi,  0.0% si
Cpu1  :  9.6% us,  8.0% sy,  0.0% ni, 82.4% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu2  :  1.3% us,  1.3% sy,  0.0% ni, 95.0% id,  2.3% wa,  0.0% hi,  0.0% si
Cpu3  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
```

```
Mem:   1034624k total,  1007420k used,     27204k free,   174540k buffers
Swap:  2031608k total,        0k used,  2031608k free,   615904k cached
```

Hack 91. Ps Command

ps command (process status) will display snapshot information of all active processes.

```
Syntax: ps [options]
```

How to display all the processes running in the system?

Use "ps aux", as shown below.

```
# ps aux | more

USER       PID %CPU %MEM    VSZ   RSS TTY  STAT START  TIME COMMAND
root         1   0.0   0.0   2044   588 ?       Ss       Jun27
0:00  init [5]
apache   31186      0.0       1.6  23736 17556 ?       S        Jul26
0:40  /usr/local/apache2/bin/httpd
apache   31187      0.0       1.3  20640 14444 ?       S        Jul26
0:37  /usr/local/apache2/bin/httpd
```

You can also use "ps -ef | more", to get a similar output

Print the Process Tree

You can use either **ps axuf** or **ps -ejH** to display processes in a tree format. The tree structure will help to visualize the process and it's parent process immediately. For clarity purpose, few columns have been cut-off in the output below.

```
# ps axuf
```

```
root     Oct14   0:00 /opt/VRTSralus/bin/beremote
root     Oct14   0:00  \_ /opt/VRTSralus/bin/beremote
root     Oct14   0:00      \_ /opt/VRTSralus/bin/beremote
root     Oct14   0:00      \_ /opt/VRTSralus/bin/beremote
root     Oct14   0:01      \_ /opt/VRTSralus/bin/beremote
root     Oct 14  0:00      \_ /opt/VRTSralus/bin/beremote
root     Dec03   0:01 /usr/local/sbin/sshd
root     Dec22   1:08 /usr/local/sbin/sshd
root     23:35   0:00  \_ /usr/local/sbin/sshd
511      23:35   0:00      \_ -bash
511                           \_ ps axuf
```

Note: You can also use pstree command to display process in tree structure.

View Processes Owned by a Particular User

The following command displays all the process owned by Linux user-name: oracle.

```
$ ps U oracle

PID TTY      STAT   TIME COMMAND
5014 ?        Ss     0:01 /oracle/bin/tnslsnr
7124 ?        Ss     0:00 ora_q002_med
8206 ?        Ss     0:00 ora_cjq0_med
8852 ?        Ss     0:01 ora_pmon_med
8854 ?        Ss     0:00 ora_psp0_med
8911 ?        Ss     0:02 oraclemed (LOCAL=NO)
```

View Processes Owned by Current User

Following command displays all the process owned by the current user.

```
$ ps U $USER

PID TTY      STAT   TIME COMMAND
10329 ?        S      0:00 sshd: ramesh@pts/1,pts/2
10330 pts/1    Ss     0:00 -bash
```

```
10354 pts/2    Ss+    0:00 -bash
10530 pts/1    R+     0:00 ps U ramesh
```

Hack 92. Df Command

df command (disk free) displays the amount of total and free disk space available on the mounted filesystems.

```
Syntax: df [options] [name]
```

How much GB of disk space is free on my system?

Use df -h as shown below. Option -h displays the values in human readable format (for example: K for Kb, M for Mb and G for Gb). In the sample output below, / filesystem has 17GB of disk space available and /home/user filesystem has 70GB available.

```
# df –h

Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1              64G   44G   17G  73%   /
/dev/sdb1             137G   67G   70G  49%   /home/user
```

What type of filesystem do I have on my system?

Option -T will display the information about the filesystem Type. In this example / and /home/user filesystems are ext2. Option -a will display all the filesystems, including the 0 size special filesystem used by the system.

```
# df -Tha

Filesystem    Type    Size  Used  Avail Use%  Mounted on
```

```
/dev/sda1      ext2   64G   44G   17G  73%   /
/dev/sdb1      ext2  137G   67G   70G  49%   /home/user
none           proc     0     0     0   -     /proc
none           sysfs    0     0     0   -     /sys
none           devpts   0     0     0   -     /dev/pts
none           tmpfs 2.0G     0  2.0G   0%   /dev/shm
```

Hack 93. Kill Command

kill command can be used to terminate a running process. Typically this command is used to kill processes that are hanging and not responding.

```
Syntax: kill [options] [pids|commands]
```

How to kill a hanging process?

First, identify the process id of the particular process that you would like to kill using the ps command. Once you know the process id, pass it as a parameter to the kill command. The example below shows how to kill the hanging apache httpd process. Please note that typically you should use "apachectl stop" to stop apache.

```
# ps aux | grep httpd
USER       PID %CPU %MEM    VSZ   RSS TTY  STAT START  TIME COMMAND
apache   31186  0.0  1.6  23736 17556 ?    S    Jul26
0:40  /usr/local/apache2/bin/httpd
apache   31187  0.0  1.3  20640 14444 ?    S    Jul26
0:37  /usr/local/apache2/bin/httpd

# kill 31186 31187
```

Please note that the above command tries to terminate the process graciously by sending a signal called SIGTERM. If the process does not get terminated, you can forcefully terminate the process by passing a signal called SIGKILL, using the option -9 as shown below. You should either be the owner of the process or a privileged user to kill a process.

```
# kill -9 31186 31187
```

Another way to kill multiple processes easily is by adding the following two functions to the .bash_profile.

```
function psgrep ()
{
 ps aux | grep "$1" | grep -v 'grep'
}

function psterm ()
{
 [ ${#} -eq 0 ] && echo "usage: $FUNCNAME STRING" && return 0
 local pid
 pid=$(ps ax | grep "$1" | grep -v grep | awk '{ print $1 }')
 echo -e "terminating '$1' / process(es):\n$pid"
 kill -SIGTERM $pid
}
```

Now do the following, to identify and kill all httpd processes.

```
# psgrep http

USER       PID %CPU %MEM    VSZ    RSS TTY  STAT START  TIME
COMMAND
apache   31186       0.0        1.6  23736 17556 ?        S
Jul26       0:40  /usr/local/apache2/bin/httpd
apache   31187       0.0        1.3  20640 14444 ?        S
Jul26       0:37  /usr/local/apache2/bin/httpd

# psterm httpd

terminating 'httpd' / process(es):
31186
```

```
31187
```

Hack 94. Du Command

du command (disk usage) will print the file space usage for a particular directory and its subdirectories.

How much space is taken by my home directory and all its subdirectories?

In the following example, option -s stands for summary only. i.e it displays only the total size of /home/jsmith and not the individual sizes of all the subdirectories inside the /home/jsmith. Option -h displays the information in a human readable format. i.e K for KB, M for MB and G for GB. The ~ indicates the user home directory. This command is same as “du -sh /home/jsmith”

```
# du -sh ~
320M    /home/jsmith
```

To get the subdirectories under /home/jsmith listed, execute the above command without the s option.

Hack 95. lsof commands.

Lsof stands for ls open files, which will list all the open files in the system. The open files include network connection, devices and directories. The output of the lsof command will have the following columns:

- COMMAND process name.
- PID process ID
- USER Username

- FD file descriptor
- TYPE node type of the file
- DEVICE device number
- SIZE file size
- NODE node number
- NAME full path of the file name.

View all open files of the system

Execute the lsof command without any parameter as shown below.

```
# lsof | more
COMMAND     PID      USER   FD     TYPE     DEVICE     SIZE      NODE
NAME
init          1      root  cwd      DIR       8,1     4096        2 /
init          1      root  rtd      DIR       8,1     4096        2 /
init          1      root  txt      REG       8,1    32684   983101 /sbin/init
init          1      root  mem      REG       8,1   106397   166798 /lib/ld-
2.3.4.so
init          1      root  mem      REG       8,1  1454802   166799
/lib/tls/libc-2.3.4.so
init          1      root  mem      REG       8,1    53736   163964
/lib/libsepol.so.1
init          1      root  mem      REG       8,1    56328   166811
/lib/libselinux.so.1
init          1      root   10u     FIFO      0,13             972 /dev/initctl
migration     2      root  cwd      DIR       8,1     4096        2 /
skipped...
```

The lsof command by itself without may return lot of records as output, which may not be very meaningful except to give you a rough idea about how

many files are open in the system at any given point of view as shown below.

```
# lsof | wc -l

3093
```

View open files by a specific user

Use lsof -u option to display all the files opened by a specific user.

```
# lsof -u ramesh

vi      7190 ramesh  txt    REG        8,1   474608
475196 /bin/vi

sshd    7163 ramesh    3u  IPv6   15088263
TCP dev-db:ssh->abc-12-12-12-12.socal.res.rr.com:2631
(ESTABLISHED)
```

A system administrator can use this command to get some idea on what users are executing on the system.

List Users of a particular file

If you like to view all the users who are using a particular file, use lsof as shown below. In this example, it displays all users who are currently using vi.

```
# lsof /bin/vi

COMMAND  PID  USER    FD   TYPE DEVICE   SIZE   NODE NAME
vi      7258  root   txt    REG    8,1 474608 475196 /bin/vi
vi      7300  ramesh txt    REG    8,1 474608 475196 /bin/vi
```

Hack 96. Sar Command

Sar commands comes with the sysstat package. Make sure sysstat is installed. If you don't have sar installed on your system, get it from Sysstat project.

Sar is an excellent monitoring tool that displays performance data of pretty much every resource of the system including CPU, memory, IO, paging, networking, interrupts etc.,

Sar Collects, Reports (displays) and Saves the performance data. Let us look at all the three aspects separately

Sadc - System activity data collector

/usr/lib/sadc (System activity data collector) command collects the system data at a specified time interval. This uses the daily activity data file that is located under /va/log/sa/sa[dd], where dd is the current day.

Sa1 shell-script

/usr/lib/sa1 in-turn calls the /usr/lib/sadcs. sa1 is invoked from the crontab as shown below. Run this every 5 minutes or 15 minutes depending on your need. I prefer to schedule it for every 5 minutes in the cron tab as shown below.

```
*/5 * * * * root /usr/lib/sa/sa1 1 1
```

Sa2 shell-script

/usr/lib/sa2 is a shell script that will write a daily report in the /var/log/sa/sa[dd] file, where dd is the current day. Invoke the sa2 from the crontab once a day at midnight.

```
# 59 23 * * * root /usr/lib/sa/sa2 -A
```

Note: /etc/cron.d/sysstat files comes with the sysstat package that includes some default value for the sa1 and sa2, which you can change accordingly.

Display CPU Statistics using Sar Command

```
# sar -u

Linux 2.6.9-42.ELsmp (dev-db)        01/01/2009
12:00:01 AM  CPU  %user  %nice   %system   %iowait     %idle
12:05:01 AM  all  3.70    0.00      0.85      0.00     95.45
12:10:01 AM  all  4.59    0.00      1.19      0.06     94.16
12:15:01 AM  all  3.90    0.00      0.95      0.04     95.11
12:20:01 AM  all  4.06    0.00      1.00      0.01     94.93
12:25:01 AM  all  3.89    0.00      0.87      0.00     95.23
12:30:01 AM  all  3.89    0.00      0.87      0.00     95.23

Skipped..

Average: all     4.56     0.00      1.00      0.15     94.29
```

Note: If you need a break down of the performance data for the individual CPU's, execute the following command.

```
# sar -u -P ALL
```

Display Disk IO Statistics using sar command

```
# sar -d

Linux 2.6.9-42.ELsmp (dev-db)        01/01/2009
12:00:01 AM    DEV              tps    rd_sec/s  wr_sec/s
12:05:01 AM    dev2-0          1.65       1.28     45.43
12:10:01 AM    dev8-1          4.08       8.11     21.81

Skipped..

Average:       dev2-0           4.66    120.77     69.45
Average:       dev8-1          1.89       3.17      8.02
```

Display networking Statistics using sar command

```
# sar -n DEV | more

Linux 2.6.9-42.ELsmp (dev-db)        01/01/2009
12:00:01 AM     IFACE   rxpck/s   txpck/s   rxbyt/s   txbyt/s
rxcmp/s   txcmp/
s  rxmcst/s
12:05:01 AM        lo      0.17      0.16     25.31     23.33
0.00      0.0
0      0.00
12:10:01 AM      eth0     52.92     53.64  10169.74  12178.57
0.00      0.0
0      0.00


# sar -n SOCK |more

Linux 2.6.9-42.ELsmp (dev-db)        01/01/2009
12:00:01 AM    totsck    tcpsck    udpsck    rawsck   ip-frag
12:05:01 AM        50        13         3         0         0
12:10:01 AM        50        13         4         0         0
12:15:01 AM        53        13         5         0         0
```

Hack 97. vmstat Command

For a typical performance monitoring all you need is only vmstat command. This display memory, swap, IO, system and cpu performance information.

The following command executes vmstat every 1 second for 100 times.

```
# vmstat 1 100

procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 0  0      0 282120 134108 5797012    0    0     0     2    0     0  0  0 100  0
 0  0      0 282120 134108 5797012    0    0     0     0 1007   359  0  0 100  0
```

```
0  0      0 282120 134108 5797012    0    0     0     0 1117   577  0  0 100  0
0  0      0 282120 134108 5797012    0    0     0     0 1007   366  0  0 100  0
```

Vmstat procs Section

- r field: Total number of runnable process
- b field: Total number of blocked process

Memory section

- Swpd field: Used swap space
- Free field: Available free RAM
- Buff field: RAM used for buffers
- Cache field: RAM used for filesystem cache

Swap Section

- Si field: Amount of memory swapped from disk per second
- So field: Amount of memory swapped to disk per second

IO Section

- Bi field: Blocks received from disk
- Bo field: Blocks sent to disk.

System Section

- In field: Number of interrupts per second.
- Cs field: Number of context switches per second.

CPU Section

- Us field: Time spend running user code. (non-kernel code)
- Sy field: Time spent running kernel code.
- Id field: Idle time.
- Wa field: Time spent waiting for the IO

Hack 98. Netstat Command

Netstat command displays the network related information such as network connections, routing tables, interface statistics. Following are few examples on how to use netstat command.

Display Active Internet Connections and domain sockets using netstat

```
# netstat -an

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign
Address             State
tcp        0      0 0.0.0.0:5666                0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:4086                0.0.0.0:*
LISTEN
skipped..
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     7894
/tmp/.font-unix/fs7100
unix  2      [ ACC ]     STREAM     LISTENING     9662
/tmp/.gdm_socket
unix  2      [ ACC ]     STREAM     LISTENING     10897
@/tmp/fam-root-
```

Display Active Connections with Process ID and Program Name

This could be very helpful to identify which program has initiated a specific network connection.

```
# netstat -tap

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign
Address             State       PID/Program name
tcp        0      0 *:nrpe                      *:*
LISTEN      16277/xinetd
tcp        0      0 localhost.localdomain:smtp  *:*
LISTEN      7263/sendmail: acce
tcp       34      0 localhost.localdomain:54221
localhost.localdomain:4089  CLOSE_WAIT  29881/httpd
tcp        0   3216 dev-db:ssh                 cpe-76-
94-215-154.soca:4682 ESTABLISHED 11717/sshd: ramesh
```

Display Routing Table

```
# netstat --route

Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS
Window  irtt Iface
192.168.1.0   *               255.255.255.0   U         0 0
0 eth0
162.244.0.0     *               255.255.0.0     U         0 0
0 eth0
default         192.168.1.1   0.0.0.0         UG        0 0
0 eth0
```

Display RAW network statistics

```
# netstat --statistics --raw

Ip:
```

```
    11080343 total packets received
    0 forwarded
    1 with unknown protocol
    0 incoming packets discarded
    11037744 incoming packets delivered
    11199763 requests sent out
Icmp:
    577135 ICMP messages received
    64 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 537
        timeout in transit: 65
        source quenches: 2
        echo requests: 576476
        echo replies: 12
        timestamp request: 3
        address mask request: 3
    581558 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 5079
        echo replies: 576476
        timestamp replies: 3
```

Misc Netstat Commands

- # **netstat --tcp --numeric** List of TCP connection to and from the machine.
- # **netstat --tcp --listening --programs** Display TCP port that the server is listening on along with the program that is listening on that particular port.
- # **netstat -rnC** Display the routing cache

Hack 99. Sysctl Command

Linux kernel parameter can be changed on the fly using sysctl command.

Sysctl helps to configure the Linux kernel parameters during runtime.

```
# sysctl -a

dev.cdrom.autoclose = 1
fs.quota.writes = 0
kernel.ctrl-alt-del = 0
kernel.domainname = (none)
kernel.exec-shield = 1
net.core.somaxconn = 128
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_wmem = 4096        16384   131072
net.ipv6.route.mtu_expires = 600
sunrpc.udp_slot_table_entries = 16
vm.block_dump = 0
```

Modify Kernel parameter in /etc/sysctl.conf for permanent change

After modifying the kernel parameter in the /etc/sysctl.conf, execute sysctl -p to commit the changes. The changes will still be there after the reboot.

```
# vi /etc/sysctl.conf

# sysctl -p
```

Modify kernel parameter temporarily

To temporarily modify a kernel parameter, execute the following command. Please note that after reboot these changes will be lost.

```
# sysctl -w {variable-name=value}
```

Hack 100. Nice Command

Kernel decides how much processor time is required for a process based on the nice value. Possible nice value range is: -20 to 20. A process that has a nice value of -20 is very high priority. The process that has a nice value of 20 is very low priority.

Use ps axl to display the nice value of all running process as shown below.

```
# ps axl

F   UID   PID  PPID PRI  NI   VSZ  RSS WCHAN  STAT TTY
TIME COMMAND
4     0     1     0  16   0  2172  552 -      S    ?
0:17 init [5]
1     0     3     1  34  19     0    0 ksofti SN   ?
3:18 [ksoftirqd/0]
1     0    10     1   5 -10     0    0 worker S<   ?
0:01 [events/0]
4     0  5145     1  25  10 32124 18592 -      SNs  ?
0:08 /usr/bin/python /usr/bin/rhn-applet-gui --sm-client-id
default4
4     0  5147  5142  16   0  3528  604 -      S    ?
0:00 /sbin/pam_timestamp_check -d root
1   503 17552  4180  16   0 14208 3920 -      S    ?
0:01 /home/www/apache2/bin/httpd -f
/home/www/apache2/conf/httpd.conf -k start
```

How to assign a low priority to a shell-script? (higher nice value)

In the example below, when I started the nice-test.sh script in the background, it took the nice value of 0.

```
$ ./nice-test.sh &
[3] 13009

$ ps axl | grep nice-test
0   509 13009 12863  17   0  4652  972 wait   S
```

```
pts/1      0:00 /bin/bash ./nice-test.sh

[Note: 6th column with value 0 is the nice.]
```

Now, let us execute the same shell script with a different nice value as shown below.

```
$ nice -10 ./nice-test.sh &
[1] 13016

$ ps axl | grep nice-test
0   509 13016 12863  30  10  4236  968 wait   SN
pts/1      0:00 /bin/bash ./nice-test.sh

[Note: 6th column with value 10 is the nice value for
the shell-script.]
```

How to assign a high priority to a shell-script? (Lower nice value)

In the following example, let us assign a nice value of -10 (minus 10) to the nice-test.sh shellscript.

```
$ nice --10 ./nice-test.sh &
[1] 13021
$ nice: cannot set priority: Permission denied
```

Note: Only root user can set a negative nice value. Login as root and try the same. Please note that there is a double dash before the 10 in the nice command below.

```
# nice --10 ./nice-test.sh &
[1] 13060

# ps axl | grep nice-test
4     0 13060 13024  10 -10  5388  964 wait   S<
pts/1      0:00 /bin/bash ./nice-test.sh
```

```
[Note: 6th column with value -10 is the nice value of
the shell-script.]
```

Hack 101. Renice Command

Renice alters the scheduling priority of a running process.

How to decrease the priority of a running process? (Increase nice)

In the example below, an existing shell-script is running at nice value of 10. (6th column in the ps output)

```
$ ps axl | grep nice-test
0   509 13245 13216  30  10  5244  968 wait   SN
pts/1      0:00 /bin/bash ./nice-test.sh
```

To increase the nice value (thus reducing the priority), execute the renice command as shown below.

```
$ renice 16 -p 13245
13245: old priority 10, new priority 16

$ ps axl | grep nice-test
0   509 13245 13216  36  16  5244  968 wait   SN
pts/1      0:00 /bin/bash ./nice-test.sh

[Note: Now, the 6th column of the nice-test.sh (PID
13245) shows the new nice value of 16.]
```

How to increase the priority of a running process? (Decrease nice)

In the example below, an existing shell-script is running at a nice value of 10.

(6th column in the ps output)

```
$ ps axl | grep nice-test
0   509 13254 13216  30  10  4412  968 wait   SN
pts/1      0:00 /bin/bash ./nice-test.sh
```

In increase the priority, give a lower nice value as shown below. However, only root can increase the priority of a running process, else you'll get the following error message.

```
$ renice 5 -p 13254
renice: 13254: setpriority: Permission denied
Login as root to increase the priority of a running
process

$ su -

# renice 5 -p 13254
13254: old priority 10, new priority 5

# ps axl | grep nice-test
0   509 13254 13216  25   5  4412  968 wait   SN
pts/1      0:00 /bin/bash ./nice-test.sh

[Note: The 6th column now shows a lower nice value of 5
(increased priority)]
```

12 Amazing and Essential Linux Books

For further reading on Linux, I recommend the following books. The 12 Linux books mentioned here by no means are comprehensive or authoritative list. But, these 12 Books are few of my favorites that I enjoyed reading over the years and I strongly believe will enhance your technical abilities on Linux, if you have not read them yet.

1. **Sed and Awk, by Dale Dougherty and Arnold Robbins**. Sed and Awk have transformed the way I worked on Linux command line. This book is the only material you would ever need on Sed and Awk. Once you've mastered even the basics of Sed and Awk, you'll be amazed with the amount of complex tasks you can perform very quickly and elegently. For my day-to-day quick reference of sed and awk examples, I use the Sed and Awk Pocket Reference, written by the same author.

2. **Learning the Vi and Vim Editors, by Arnold Robbins**. I'm a command-line junkie. So, naturally I'm a huge fan of Vi and Vim editors. Several years back, when I wrote lot of C code on Linux, I used to carry the Vi editor pocket reference with me all the times. Even if you've been using Vi and Vim Editors for several years and have not read this book, please do yourself a favor and read this book. You'll be amazed with the capabilities of Vim editor.

3. **Bash Cookbook, by Carl Albing, JP Vossen and Cameron Newham**. Whether you are a sysadmin, DBA or a developer, you have to write shell script at some point. A wise sysadmin knows that once you've mastered the shell-scripting techniques, you can put your servers on auto-pilot mode by letting the shell-scripts do the grunt work. To get to the auto-pilot mode of sysadmin, you definitely need to master the examples provided in this cookbook. There are quiet few Bash shell books out there. But, this books tops them all by giving lot of detailed examples.

4. **SSH, The Secure Shell, by Daniel J. Barrett, Richard E. Silverman and Robert G. Byrnes**. This is hands-down the best book on SSH. This book explains both theoretical and practical aspects of SSH. Using

SSH as an end-user is fairly straight forward . But, configuring SSH as an administrator is complex and involves a detailed understanding of SSH. This is a must read for any system administrator. The examples in this book show exactly what needs to be done differently for the different flavors of SSH such as SSH1, SSH2 and OpenSSH.

5. **Essential System Administration, by Æleen Frisch.** This is an excellent book for those who like to become a Unix System Administrator. This book covers all the typical system administration tasks. This is a perfect companion when you are dealing with multiple flavors of Unix, as it has examples for AIX, FreeBSD, HP-UX, Linux, Solaris and Tru64. I've used the pocket version of this book — Essential System Administration Pocket Reference, when I was managing multiple flavors of Unix systems at the same time.

6. **Linux Server Hacks, Volume One, by Rob Flickenger**. 100 awesome practical hacks packed in one book. Setup a Linux test bed and try out all these hacks. These hacks are neatly grouped into different sections — Server Basics, Revision Control, Backups, Networking, Monitoring, SSH, Scripting, and Information Servers. Once you've mastered these hacks, you should absolutely read Linux Server Hacks, Volume Two, by William von Hagen and Brian Jones, which has 100 Linux hacks focussed on authentication, monitoring, security, performance and connectivity.

7. **DNS and BIND, by Cricket Liu and Paul Albitz.** Several years ago, I configured my first DNS by reading online documentation. I brought this book to understand how DNS and BIND works. I've already upgraded this book twice when a newer edition was released. This should definitely be in your library, if you are a serious system administrator.

8. **Understanding the Linux Kernel, by Daniel Bovet and Marco Cesati.** If you are a serious developer on Linux environment or a sysadmin, this is a must read. This books explains the inner workings of the Linux Kernel 2.6 in a structured and logical way. This talks about how Kenel handles the Memory Management, Process scheduling, I/O architecture and Block devices. Overall this book is a treat for geeks who are curious to explore what is under the hood of Linux.

9. **Linux Cookbook, by Carla Schroder**. This book covers Linux features from both users and system administrators point of view. There are

two chapters dedicated for installing and managing software on RPM-based system and Debian. If you use RedHat, the Linux Pocket Guide, by Daniel J. Barrett is an excellent addition to your library, which covers all the essential Linux command with a sample usage.

10. **Linux Firewalls, by Michael Rash**. To build a secure Linux system, you must read this book. There are quiet few books out there for iptables. But, this one talks specifically about the fundamentals of how to configure an Intrusion Detection System using iptables, psad and fwsnort. If you want a comprehensive handy reference of all the things iptables can do with specific examples, Linux Iptables Pocket Reference, by Gregor N. Purdy is the best.

11. **Linux Administration Handbook, by Evi Nemeth, Garth Snyder and Trent R. Hein**. During my early days of system administration, I've referred this book frequently. This is pretty detailed book with close to 1000 pages and 30 chapters that are nicely grouped together in three high level sections — Basic Administration, Networking and Bunch O' Stuff.

12. **Beginning Ubuntu Linux, by Keir Thomas and Jaime Sicam**. For those who like to transition from Windows to Linux, install Ubuntu Linux on one of your old laptop or desktop and get this book. I strongly believe in spreading the news about Linux to those who don't use it. If you want any of your loved ones or friends to learn Linux, install Ubuntu on an old laptop and give this book as a gift to them. They'll definitely be very thankful to you.

Extended Reading

Following are few articles from The Geek Stuff blog for your extended reading. Check out Best Of The Blog section for more articles.

- Turbocharge PuTTY with 12 Powerful Add-Ons
- Nagios - Enterprise Monitoring Solution
  - Nagios Jumpstart Guide
  - Monitor Window Server
  - Monitor Linux Server
  - Monitor Network Switch
  - Monitor VPN Device
- Perform SSH and SCP without entering password:
  - From openSSH to openSSH
  - From openSSH to SSH2
  - From SSH2 to SSH2
- Vi / Vim Tips and Tricks
  - Vim Macro Tutorial: How To Record and Play
  - How To Use Vim as Perl IDE and C/C++ IDE
  - Automatic Word Completion in Vim
  - 3 Steps to Add Custom Header to a File Using Vim
- The Ultimate Guide for Creating Strong Passwords
- Firefox Add-On: Hire 7 Personal Bodyguards to Browse Internet Securely
- Tripwire Tutorial: Linux Host Based Intrusion Detection System
- Midnight Commander (mc) Guide: Powerful Text based File Manager for Unix

Your Feedback and Support

I hope you found **Linux 101 Hacks** eBook helpful. Thanks for reading. I sincerely appreciate all the support given by the regular readers of my blog. Without your tremendous support, it would've been difficult to find the motivation to write this eBook.

Subscribe to TGS

To get Linux Tips, HowTos, Guides and Tutorials on an on-going basis, please subscribe to The Geek Stuff blog. If you subscribe, you will get new articles posted on TGS website directly to your inbox or to your RSS reader.

Contact TGS

Please use this contact form to send me your feedback, question, or clarification on any of the 101 hacks mentioned in this eBook.