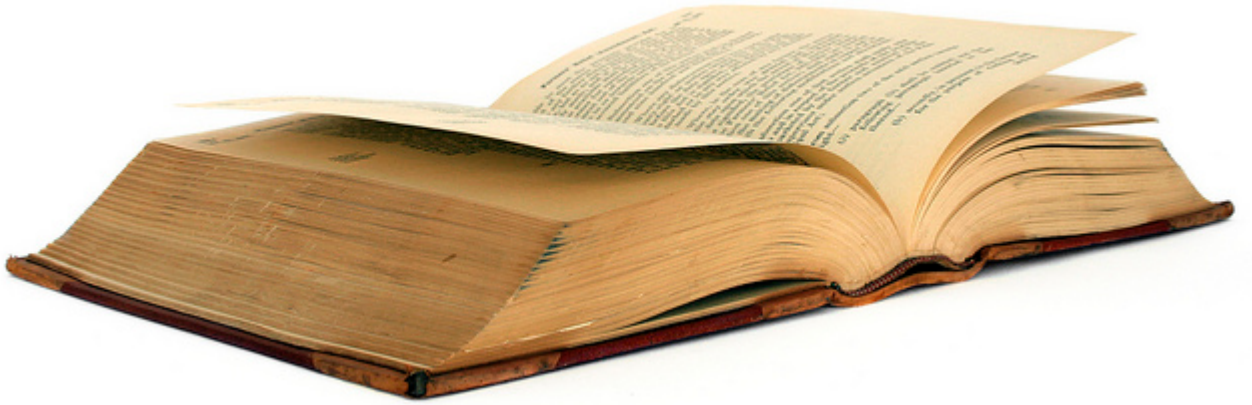
Linux Arcana

Daniel Hanks | Sr. System Administrator

My favorite solution to most problems:

My favorite solution to most problems:

RTFM

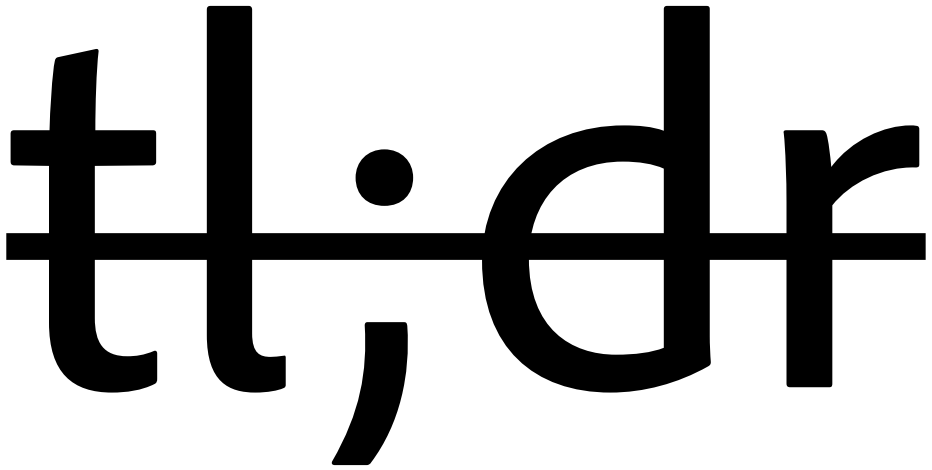


My favorite solution to most problems:

RTFM

(Read the **FULL** manual)

tl;dr



Most of your problems have already been solved for you.

- Most of the time:
  - You don’t need to write a clever script.
  - You don’t need to hack/patch the code.
  - You don’t need the 2000-character one-liner.
  - (Even though it’s awful fun to do all of the above)

Most of your problems have already been solved for you.

- You just need to use the features your tools already have.
- Or use the tool you didn't know you had…

Most of your problems have already been solved for you.

- Wisdom is knowing when you need to hack…

Most of your problems have already been solved for you

- Spend more time **building your product**, and less time writing tools that have already been written for you.
- However, it's a question of:
  - How long will it take me to come up with the hack vs.
  - How long will it take me to find the tool or command-line arg that will solve the problem…
  - Reminds me of the halting problem.

The aims of this talk

- Know Your Tools
- What do all those things in /bin, /usr/bin, /sbin, /usr/sbin do?
- Do you know all of their command-line options?
- (I don't … yet)
- Criteria for inclusion:
  - (Relatively to very) Obscure
  - Useful

Arcana Exploration Strategy

```
### What package installed /bin/ls?
]$ rpm –qf /bin/ls
coreutils-7.2-4.fc11.i586

### What other curious binaries did coreutils install?
]$ rpm –ql coreutils | grep bin
/bin/arch
/bin/basename
/bin/cat
/bin/chgrp
/bin/chmod
/bin/chown
/bin/cp
/bin/cut
/bin/date
...

### What do all of those do?
```

Arcana Exploration Strategy

```
### Where is the ssh binary installed?
]$ which ssh
/usr/bin/ssh

### What package installed that binary?
]$ rpm –qf /usr/bin/ssh
openssh-clients-5.2p1-2.fc11.i586

### What other interesting things did that package install?
]$ rpm –ql openssh-clients | grep bin
...
/usr/bin/ssh-add
/usr/bin/ssh-agent
/usr/bin/ssh-copy-id
/usr/bin/ssh-keyscan

### Ooooooooh, what do those do?
man ssh-copy-id
```

And speaking of openssh…

```
### ssh-copy-id: Distribute public ssh keys easier

### Compare the old way...
]$ ssh remote.host
[remote.host]$ vi .ssh/authorized_keys
[remote.host]$ chmod 600 .ssh/authorized_keys

### With the easier way...
]$ ssh-copy-id remote.host

### If you have a particular key you want to copy over…
]$ ssh-copy-id –i super_duper_key remote.host
```

Speaking of ssh keys…

```
### SSH Keys & Agents – save keystrokes, and stop sending your
### password over the wire

### Create a keypair
]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/<user>/.ssh/id_rsa):
/home/<user>/.ssh/super-duper-key
Enter passphrase (empty for no passphrase): <type passphrase>
Enter same passphrase again: <type passphrase again>
Your identification has been saved in /home/<user>/.ssh/super-
duper-key.
Your public key has been saved in /home/<usre>/.ssh/super-duper-
key.pub.
The key fingerprint is:
ef:a9:67:5a:0b:7e:41:eb:88:9f:3c:ce:49:0e:bb:91 <user>@hyades

### See ssh-keygen(1) for a ton of other options to craft your
### ideal keypair.
```

Speaking of ssh keys…

```
### Now distribute your key
]$ ssh-copy-id –i ~/.ssh/super-duper-key remote.host

]$ ssh remote.host
Enter passphrase for key 'super': <key pasphrase goes here>

### More secure, but we still have to provide a passphrase.
### Let’s hire an agent to do that for us.

### Fortunately ssh agents are pretty easy to come by,
```

SSH Agents

```
### Spin up an agent (Latin-derived word for one who does
### something for us)
]$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-FIeuY25965/agent.25965; export
SSH_AUTH_SOCK;
SSH_AGENT_PID=25966; export SSH_AGENT_PID;
echo Agent pid 25966;

### Either copy and run those shell variables, or invoke with
### `ssh-agent` or 'ssh-agent /bin/bash' (though there are
### subtle differences between these two invocations).

### Now add your key to the agent
]$ ssh-add super-duper-key
Enter passphrase for super-duper-key: <passphrase goes here>
Identity added: super (super-duper-key)

### Now we can ssh without passphrases or passwords
]$ ssh remote.host
Last login: Fri May  3 23:14:22 2013 from some host
```

SSH Agents

```
### What keys have we added to our agent?
]$ ssh-add –l

### Only add your keys for a maximum of 3-week
]$ ssh-add –t 3w

### Lock the agent with a password
]$ ssh-add -x
Enter lock password: <type a lock password>
Again: <and again>
Agent locked.
$ ssh remote.host
user@remote.host's password:

### Now unlock the agent
]$ ssh-add -X
Enter lock password: <lock password>
Agent unlocked.
```

SSH Agents - Forwarding

```
### If your pub keys are spread throughout your machines,
### you only need to fire up an agent once. Just enable
### agent forwarding in your local SSH config:

]$ echo “ForwardAgent yes” >> ~/.ssh/config

### Now you can hop from machine to machine wherever your
### pubkey is available.
```

Speaking of the SSH Client Config

```
### Usually ~/.ssh/config
### See ssh_config(5)

### A favorite for dealing with over-zealous
### connection-closing firewalls
ServerAliveInterval 60

### Host / User Aliases
Host myhost
 User my_other_name
 IdentityFile ~/.ssh/my_other_key
 Hostname some.real.hostname

]$ ssh myhost
my_other_name@some.real.hostname's password:

### We're only scratching the surface here. Tons of useful
### options.
```

SSH: Escape sequences

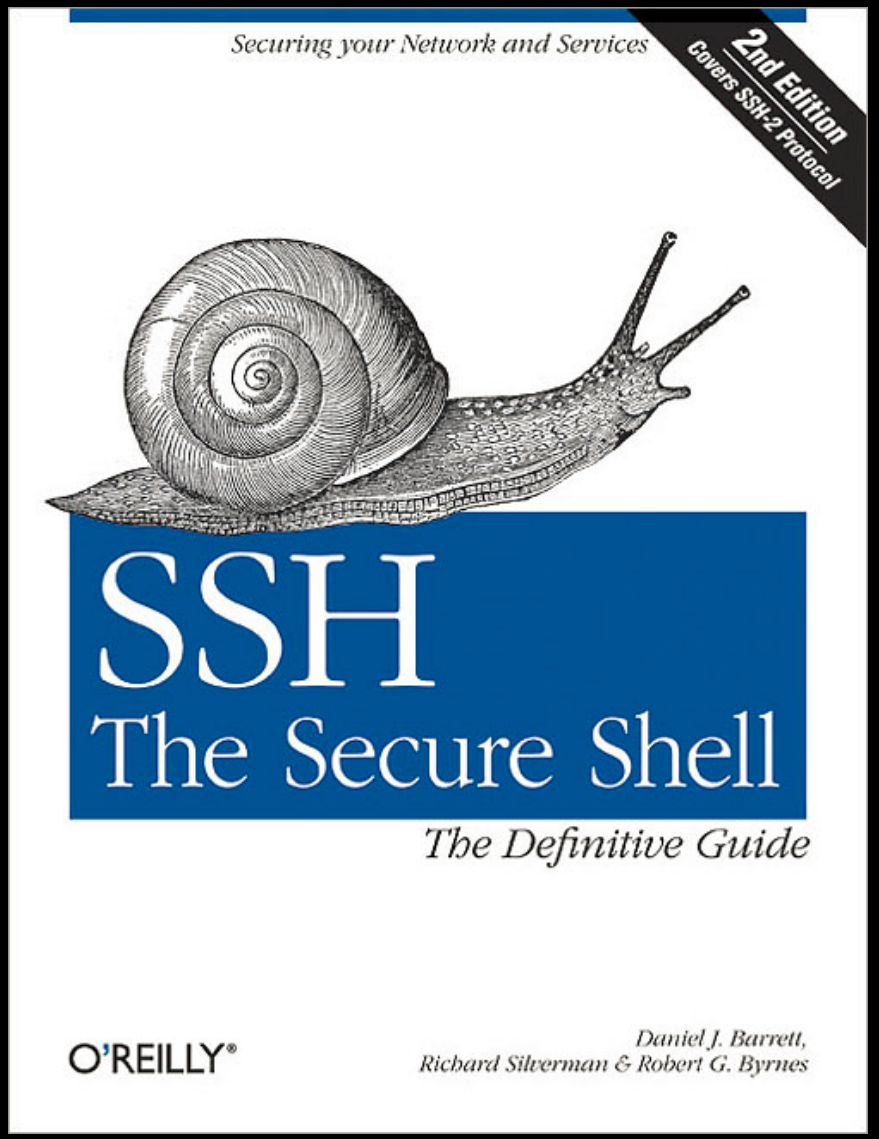
```
### Default escape character is newline/carriage return
### followed by '~', '~' can be overridden with
### ssh –e <char>

### Suspend the current ssh shell
[user@host1] ssh host2
[user@host2] ~<Ctrl-Z>
### Now back on host1
[1]+  Stopped                 ssh host2
### Resume the session on host2
[user@host1]fg
ssh host2
[user@host2]
```

SSH: Escape sequences

```
### Change or add tunnels / forwarding mid-session
[user@host1]$ <Enter>~C
ssh> ?
Commands:
    -L[bind_address:]port:host:hostport    Request local forward
    -R[bind_address:]port:host:hostport    Request remote forward
    -D[bind_address:]port                  Request dynamic forward
    -KR[bind_address:]port                 Cancel remote forward

### List all forwarded connections
[user@host1]$ <Enter>~#
The following connections are open:
  #0 client-session (t4 r0 i0/0 o0/0 fd 4/5 cfd -1)

### Terminate your ssh connection (useful for frozen shells)
[user@host1]$ <Enter>~.

### We could spend another session talking about ssh tunnels
```

SSH: So much more we haven't covered

~ 600 pages of SSH arcana



So much more we haven't covered

Now – go forth and apply this Arcanic Search™ methodology to everything else on your Linux system...

One more tidbit of ssh: ssh-keyscan

```
### ssh-keyscan – Useful for pre-populating known_hosts
### files, scans with non-blocking IO in parallel
ssh-keyscan host1 host2 host3 ... hostN
# host1 SSH-2.0-OpenSSH_5.8
host1 ssh-rsa <ssh public host key>
# host2 SSH-2.0-OpenSSH_5.4
host2 ssh-rsa <ssh public host key>

### Scan hosts from a list, one per line
ssh-keyscan –f <file>

### Scan hosts from STDIN
echo –e “host1\nhost2\nhost3\n” | ssh-keyscan -
```

One more tidbit of ssh: ssh-keyscan

```
### ssh-keyscan – Useful for pre-populating known_hosts
### files, scans with non-blocking IO in parallel
ssh-keyscan host1 host2 host3 ... hostN
# host1 SSH-2.0-OpenSSH_5.8
host1 ssh-rsa <ssh public host key>
# host2 SSH-2.0-OpenSSH_5.4
host2 ssh-rsa <ssh public host key>

### Scan hosts from a list, one per line
ssh-keyscan –f <file>

### Scan hosts from STDIN
echo –e “host1\nhost2\nhost3\n” | ssh-keyscan -
```

The ‘-’ filename

```
### A lot of man pages indicate you can pass in ‘-’ (STDIN)
### for a filename. Why would you want to do that?

### Diff a file on the local machine, with one on a remote
### machine
]$ cat /some/file | ssh remote.host ‘diff –u /some/file –’

### And go the other way
]$ ssh remote.host ‘cat /some/file’ | diff /some/file –’

### For colorized diff inspection you could do
]$ diff <file1> <file2> > mydiff
]$ vim mydiff

### But this is faster (demo)
]$ diff /some/file /other/file | vim –
### Also see vimdiff, vim -d
```

The '-' filename

```
### Poor mans rsync
]$ tar zcvf - /some/local/dir | \
   ssh remote.host tar zxvf – –C /some/remote/dir

### On a related note, many programs take -- (double-dash)
### to tell them to stop processing further command line
### args. Useful in the following scenario
]$ rm -- -some-file-that-starts-with-a-dash
```

The '-' filename

```
### Poor mans rsync
]$ tar zcvf - /some/local/dir | \
   ssh remote.host tar zxvf – –C /some/remote/dir

### On a related note, many programs take -- (double-dash)
### to tell them to stop processing further command line
### args. Useful in the following scenario
]$ rm -- -some-file-that-starts-with-a-dash
```

Bash continuation lines with '\'

```
### Use '\' to break up long command-lines
### Useful to make documentation of long command lines
### more readable, and you can still cut and paste
]$ ./my_command  \
      --arg1=one \
      --arg2=two \
      --other-arg=something-else
```

Moving on to shell loops (BASH)

```
for x in <list>; do <something with $x>; done

# E.g.,
for file in `ls`; do cat $file; done
for dir in `find /home`; do chown root $dir; done
for file in *.txt; do cat $file; done

# Loops can be nested
for dir in `find /home`; do for file in `ls $dir`; do echo
“$dir/$file”; done; done

# But watch out for lists that get too big
# Switch to xargs under such cases
find /home | xargs chown root
```

Moving on to shell loops (BASH)

```
for x in <list>; do <something with $x>; done

# E.g.,
for file in `ls`; do cat $file; done
for dir in `find /home`; do chown root $dir; done
for file in *.txt; do cat $file; done

# Loops can be nested
for dir in `find /home`; do for file in `ls $dir`; do echo
“$dir/$file”; done; done

# But watch out for lists that get too big
# Switch to xargs under such cases
find /home | xargs chown root
```

Some Latin arcana…at no extra cost

E.g. (Latin for *exempli gratia*, use in place of '*for example*'…)
I.e. (Latin for *id est*, use in place of '*that is…*')
Viz. (Latin for *videlicet*, use in place of '*namely…*')
Etc. (Latin for *et cetera*, meaning, literally, '*and others*' (things…))
Et al (Latin for *et* {*alii* (m), *aliae* (f), *alia* (n)}, meaning '*and others*' (people)*
See also *Et alibi* = '*and others*' (places)

* http://www.thefreedictionary.com/et+al.

Some Latin arcana…at no extra cost

E.g. (Latin for *exempli gratia*, use in place of '*for example*'…)
I.e. (Latin for *id est*, use in place of '*that is…*')
Viz. (Latin for *videlicet*, use in place of '*namely…*')
Etc. (Latin for *et cetera*, meaning, literally, '*and others*' (things…))
Et al (Latin for *et* {*alii* (m), *aliae* (f), *alia* (n)}, meaning '*and others*' (people)*
See also *Et alibi* = '*and others*' (places)

And speaking of curly brace expansion…

* http://www.thefreedictionary.com/et+al.

Brace Expansion (BASH)

```
### An elegant way to generate lists
]$ for server in {larry,moe,curly}.example.com; do
  echo $server
done
larry.example.com
moe.example.com
curly.example.com

### Elements in the {} list can be empty
]$ for server in www{,2,3,4,5}.example.com; do
  echo $server
done
www.example.com    # Note the empty list member here
www2.example.com
www3.example.com
www4.example.com
www5.example.com
```

Brace Expansion (BASH)

```
### Useful to save typing
]$ cp /some/path/to/a/file.{old,new}

### Expands to:
]$ cp /some/path/to/a/file.old /some/path/to/a/file.new


### Lists can be nested
]$ mkdir -p
rpm/{SRPMS,BUILD,SOURCES,SPECS,RPMS/{i386,noarch,i686}}

### Creates
### rpm/SRPMS rpm/BUILD rpm/SOURCE rpm/SPECS
### rpm/RPMS/i386 rpm/RPMS/noarch rpm/RPMS/i686
```

seq

```
### seq: A useful tool for generating numeric-based lists

]$ seq 3
1
2
3

]$ seq 4 8
4
5
6
7
8

### Count 1 to 10, incrementing by 2
]$ seq 1 2 10
1
3
5
7
9
```

seq

```
### Specify a separator with -s
]$ seq -s " " 1 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

]$ seq -s "-" 1 20
1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20

### -w: Make all values the same width, pad with zeros
]$ seq –w 1 100
001
002
003
004
005
006
...
097
098
099
100
```

seq

```
### Pass in format strings with -f
### -f <%e, %f or %g> (exp., floating point, integer)
]$ for host in `seq –f “www%g.example.com” 1 100`; do
   echo $host
done
www1.example.com
www2.example.com
...
www10.example.com

### Note the use of backticks here to swap in the output of
### running the seq command in the backticks.
### Alternatively, $(command ...) does similar
```

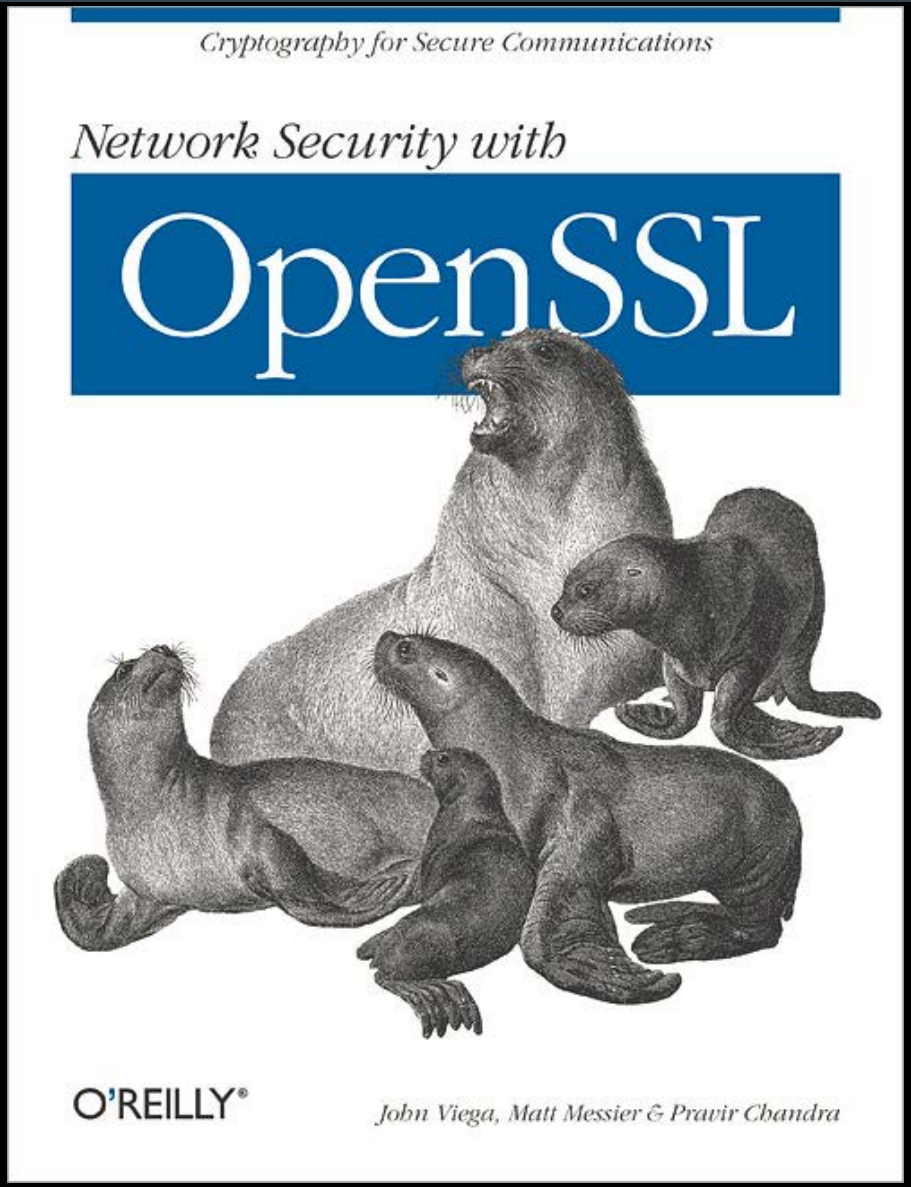
OpenSSL: s_client

```
### Need to interact with a web server using SSL? (Demo)
Use s_client to interact normally (HTTP), over SSL
]$ openssl s_client –connect www.host.com:443
<Lots of SSL Output>
GET / HTTP/1.1
Host: www.host.com
...
```

OpenSSL: General

```
### Much more to explore here

Standard commands
asn1parse      ca             ciphers        crl            crl2pkcs7
dgst           dh             dhparam        dsa            dsaparam
enc            engine         errstr         gendh          gendsa
genrsa         nseq           ocsp           passwd         pkcs12
pkcs7          pkcs8          prime          rand           req
rsa            rsautl         s_client       s_server       s_time
sess_id        smime          speed          spkac          verify
version        x509

Message Digest commands (see the `dgst' command for more details)
md2            md4            md5            rmd160         sha
sha1

Cipher commands (see the `enc' command for more details)
aes-128-cbc    aes-128-ecb    aes-192-cbc    aes-192-ecb    aes-256-cbc
aes-256-ecb    base64         bf             bf-cbc         bf-cfb
bf-ecb         bf-ofb         cast           cast-cbc       cast5-cbc
cast5-cfb      cast5-ecb      cast5-ofb      des            des-cbc
des-cfb        des-ecb        des-ede        des-ede-cbc    des-ede-cfb
des-ede-ofb    des-ede3       des-ede3-cbc   des-ede3-cfb   des-ede3-ofb
des-ofb        des3           desx           rc2            rc2-40-cbc
rc2-64-cbc     rc2-cbc        rc2-cfb        rc2-ecb        rc2-ofb
rc4            rc4-40
```

OpenSSL: General



The humble ls

```
ls --full-time # Show long timestamps
ls –B # Ignore files ending in ~ (backups)
ls –h # Human-readable sizes (G, K, M, etc.)
ls –I # Print the inode number of each file
ls –F # “Append indicator (one of */=>@|) to entries”
ls –Q # Enclose each entry in double-quotes
ls –Q --quoting-style=(literal,locale,shell,shell-
always,c,escape)
ls –r # Reverse order
ls –R # Recursive
ls -1 # Only one entry per line
ls –z # Show selinux context info
ls –m # Separate entries by commas (instead of newlines)
ls –X # Sort by filename extension
la –lart # Long listing, all files, reverse sort, sort by
time
```

stat: when ls isn't enough

```
### stat
]$ stat file
 File: `file'
  Size: 11              Blocks: 8          IO Block: 4096
regular file
Device: fe00h/65024d    Inode: 3152063     Links: 1
Access: (0600/-rw-------)  Uid: (60041/  dhanks)   Gid: (
522/  dhanks)
Access: 2013-05-04 00:12:11.273892983 -0700
Modify: 2013-05-04 00:12:11.273892983 -0700
Change: 2013-05-04 00:12:11.273892983 -0700

### Use with format strings (--format=FORMAT) for flexible
### output. See stat(1) for more details.
### E.g.,
]$ stat –c “%a %A %i %s“ file
600 -rw------- 3152063 11
```

stat: when ls isn't enough

```
### A substitute for ls –l with stat, showing permissions in octal
]$ stat -c "%a %h %U %G %s %y %n" *
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 eight
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 five
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 four
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 nine
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 one
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 seven
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 six
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 ten
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 three
664 1 dhanks dhanks 0 2013-05-04 10:00:00.000000000 -0600 two
```

ps: Customized list of running processes with output formats

```
### Show a process tree, ps -f
]$ ps fauwx
...
root      3059  0.0  0.1  54136  2284 ?        Ss    2012  17:23 /usr/libexec/postfix/master
postfix   3073  0.0  0.1  54460  2592 ?        S     2012   9:05  \_ qmgr -l -t fifo -u
postfix   8901  0.0  0.1  54204  2252 ?        S    08:29   0:00  \_ pickup -l -t fifo -u
postfix   9949  0.0  0.1  56664  2940 ?        S    10:02   0:00  \_ smtpd -n smtp -t inet -u
postfix   9950  0.0  0.1  54184  2236 ?        S    10:02   0:00  \_ proxymap -t unix -u
postfix   9951  0.0  0.1  54200  2252 ?        S    10:02   0:00  \_ anvil -l -t unix -u
postfix   9952  0.0  0.1  54212  2540 ?        S    10:02   0:00  \_ trivial-rewrite -n rewrite -t unix -
u
...

### Use –o to specify format (No spaces in your format)
]$ ps –eo pid,state,user,args
...
 3171 S root     /usr/sbin/atd
 3200 S 68       hald
 3201 S root     hald-runner
 3251 S root     /usr/sbin/smartd -q never
 3254 S root     /sbin/mingetty tty1
 3255 S root     /sbin/mingetty tty2
 3259 S root     /sbin/mingetty tty6
 3291 S root     /usr/bin/python -tt /usr/sbin/yum-updatesd
 3315 R root     /usr/libexec/gam_server
...
```

df – Show available disk space

```
### df –i = Show inode usage
### When you get “Disk full” messages, but have plenty of
### room on the drive
]$ df –i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/mapper/VolGroup00-LogVol00
                     59998208  113573 59884635    1% /
/dev/sda1              26104      45   26059    1% /boot
tmpfs                 256383       1  256382    1% /dev/shm
/dev/sdb1            30539776   21474 30518302    1% /backup
```

lsof – List Open Files

```
### By itself (if you're root), a listing of all open files
### on the system
]# lsof
...
### But not limited to files in the traditional sense
### List all open TCP sockets
]# lsof –iTCP

### List all files opened by pid 1234
]# lsof –p 1234

### List all open files by crond, httpd, or sendmail
]# lsof –c crond –c httpd –c sendmail

### List all files open by procs matching <regex>
]# lsof –c/<regex>/

### List all open files in /var
]# lsof +D /var
```

lsof – List Open Files

```
### List all processes which have this file open
]# lsof /var/mysql/mysql/user.frm

### And so much more...
```

sort

```
### Basic alphabetical sort
echo –e “bravo\nalpha\ndelta\ncharlie” | sort
alpha
bravo
charlie
delta

### Numeric sort
sort –n <file>

### Reverse sort
sort –r <file>

### Sort by the second field in each line, fields separated by ‘|’
sort –k 2 -t ‘|’ <file>

### A sort chain I use often:
cat <file> | sort | uniq –c | sort –n

### Sort standard input (default, or specify ‘-’)
cat file | sort
```

sort

```
### Human-readable numeric sort (can deal with ‘10k’, ‘100G’, etc)
### Only in newer sorts...
]$ sort –h <file>

### Combine with du to find disk hogs
]# cd /
]# du –sh * | sort -h
0       proc
0       sys
4.0K    mnt
4.0K    selinux
4.0K    srv
16K     lost+found
44K     dev
3.4M    tmp
7.5M    bin
13M     boot
14M     sbin
21M     lib64
23M     opt
35M     etc
52M     lib
142M    root
228M    var
1.2G    usr
23G     home
```

vim

```
### Vim as binary editor
vim –b <binary file>

### Start in vimdiff mode
vim –d <file1> <file2>

### Start in read-only mode
vim –R # See also view

### For the paranoid, encrypt the file you edit (with
### caveats, see :help encryption, see also :X
### Not particularly strong encryption.
vim –x <file>

### Start vim in restricted mode (See also rvim, rview)
### Prevents opening shells or suspending vim. Useful for
### handing out vim access via sudo
vim -Z
```

watch

```
### Run a command repeatedly, shows command output over
### time
]$ watch –n 1 ‘cat /proc/meminfo’

### Highlight changes between runs
]$ watch –d –n 1 ‘cat /proc/meminfo’
```

cal

```
### Print out an Ascii calendar
]$ cal 6 2014
     June 2014
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

### Last month, this month, and next month
]$ cal -3
      April 2013            May 2013              June 2013
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6              1  2  3  4                  1
 7  8  9 10 11 12 13   5  6  7  8  9 10 11   2  3  4  5  6  7  8
14 15 16 17 18 19 20  12 13 14 15 16 17 18   9 10 11 12 13 14 15
21 22 23 24 25 26 27  19 20 21 22 23 24 25  16 17 18 19 20 21 22
28 29 30              26 27 28 29 30 31     23 24 25 26 27 28 29
                                            30
```

bc – The incredibly capable command-line calculator

```
### Pipe from the command-line
]$ echo “1 + 2 * 3 / 4 ^ 6” | bc –l
1.00146484375000000000

### Or run interactively
]$ bc –l
3 * 2
6
obase=16
12
C
16
10
obase=2
32
100000
345
101011001
obase=10
ibase=2
111
7
2^10
1024
```

rpm

```
### List all files installed by an rpm
]$ rpm –ql openssh

### What package installed /usr/bin/foo?
]$ rpm –qf /usr/bin/foo

### Is everything installed by openssh intact?
### Size, Mode, MD5 sum, Device, Link, User, Group, mTime
### . = passed, ? = can't read to verify
### Attribute markers: config, doc, ghost, license, readme
]$ rpm –V httpd
]$ rpm -V httpd
.......T    /etc/httpd/conf.d/README
.......T  c /etc/httpd/conf.d/proxy_ajp.conf
.......T  c /etc/httpd/conf.d/welcome.conf
S.5....T  c /etc/httpd/conf/httpd.conf
..?.....    /usr/sbin/suexec

### Is everything install by rpm intact?
]$ rpm -Va
```

rpm

```
### Customize output format with –queryformat (Don’t forget \n)
]$ rpm –qa --queryformat=“%{epoch}:%{name}-%{version}-%{arch}\n”
(none):basesystem-8.0-noarch
(none):zlib-1.2.3-x86_64
(none):popt-1.10.2.3-x86_64
(none):libtermcap-2.0.8-x86_64
(none):libsepol-1.15.2-x86_64
(none):procps-3.2.7-x86_64
(none):libSM-1.0.1-x86_64
(none):libidn-0.6.5-x86_64
(none):libattr-2.4.32-x86_64
(none):binutils-2.17.50.0.6-x86_64
...

### Fix file permissions (per package spec)
]$ rpm --setperms openssh
]$ rpm --setugids openssh

### Fix everything
]$ for pkg in `rpm –qa`; do rpm –setperms $package; done
```

yum

```
### Only one tip to share with you (newer yums)
yum distro-sync
```

“Synchronizes the installed package set with the latest packages available, this is done by either obsoleting, upgrading or downgrading as appropriate. This will "normally" do the same thing as the upgrade command however if you have the package FOO installed at version 4, and the latest available is only version 3, then this command will downgrade FOO to version 3.”

awk

```
### Problem:
### Show me all HTTP 500 errors in my Apache logs.
### Just use grep, right?
]$ grep 500 /var/log/httpd/access.log

### But what if ‘500’ shows up in some of the URLs we serve?
### Well, we know the status code is in field 10 of each line:
]$ awk ‘$10 ~ /^5/’ /var/log/httpd/access.log

### Print only fields 7 and 8, from such lines
]$ awk ‘$10 ~ /^5/ {print $7, $8}’ /var/log/httpd/access.log

### What if our we have (potentially multiple) spaces in some of
### the fields before field 10?
### Let’s say we know the status code is 2 fields from line end…
### Display lines from a file where the last field from the end
### of the line matches ^5 (like HTTP 500 errors)
awk '$(NF - 1) ~ /^5/' /var/log/httpd/access.log

### Awk basic invocation pattern
]$ awk ‘<pattern> { <do stuff> }’
```

A note about man pages…

Many man pages say:

“The full documentation for <command> is maintained as a Texinfo manual. If the info and <command> programs are properly installed at your site, the command

info coreutils 'ls invocation'

should give you access to the complete manual.

- info usually has more details
- But you have to know how to navigate info pages (demo)
  - h for help
- Remember, man pages sometimes have more than one section

]$ man [<section>] <something>

Arcane homework assignments…

- expect and autoexpect
- script
- Generic Colourizer: http://kassiopeia.juls.savba.sk/~garabik/software/grc/README.txt
- man proc (Tons of detail about the /proc filesystem—Lots of goodies hiding in there).
- Ssrace, ltrace
- ip (iproute2)
- swapon / swapoff
- comm (as opposed to diff)
- logger (send stuff to syslog from the command line)
- https://github.com/jkbr/httpie
  - Amazing alternative to wget, curl, et al.
- http://public.wsu.edu/~brians/errors/errors.html
  - Common errors in English Usage (not Linux, but fun)
- http://mmb.pcb.ub.es/~carlesfe/unix/tricks.txt
- .d directories (/etc/httpd/conf.d, /etc/cron.d, etc.)
- nc (netcat)
- /bin/{false,true,yes}
- diff, patch

Other sources of arcana

- http://news.ycombinator.com
- UNIX Power Tools (O’Reilly)
- Linux in a Nutshell (O’Reilly)
- Man pages
- Info pages

Image credits

- Book: http://www.flickr.com/photos/brenda-starr/5076790282/sizes/z/in/photostream/
- Toolbox: http://www.flickr.com/photos/publicresourceorg/493813720/sizes/o/in/photostream/

Info

- Slides: brainshed.com
- @danhanks
- danhanks@gmail.com


Adobe