

A Mac/Unix Computing Cheatsheet, 2003–06

Mac OS X Specific Hacks

Mac OS X actually has a hibernate mode (the contents of the memory get saved to the disk when you put the computer to sleep). To disable:

```
pmset -g | grep hibernatemode && sudo pmset -a hibernatemode 0  
cd /var/vm; sudo rm sleepimage
```

Screen sharing – the VNC server and client are actually built into OS X

You can either type in the browser `vnc://yourmacname.local`, or open a "Screen Sharing" application

Some customizations:

```
defaults write com.apple.ScreenSharing ShowBonjourBrowser_Debug 1  
defaults write com.apple.ScreenSharing 'NSToolbar Configuration ControlToolbar'\  
-dict-add 'TB Item Identifiers'\  
'(Scale,Control,Share,Curtain,Capture,FullScreen,GetClipboard,SendClipboard,Quality)'
```

Make hidden apps translucent

```
defaults write com.apple.Dock showhidden -bool YES
```

Put the screensaver on the Mac's background

```
/System/Library/Frameworks/ScreenSaver.framework/Resources/\  
ScreenSaverEngine.app/Contents/MacOS/ScreenSaverEngine -background &
```

Rebuild launch services database

```
/System/Library/Frameworks/ApplicationServices.framework/\  
Frameworks/LaunchServices.framework/Support/lregister\  
-kill -r -domain local -domain system -domain user
```

Create links to AFP locations

type `afp://computername.local/VolumeName` in TextEdit, select it and drag&drop to desktop

Show hidden files

```
defaults write com.apple.finder AppleShowAllFiles TRUE
```

Run Expose from command line (other key codes are 109, 103)

```
osascript -e 'tell application "System Events" to key code 101'
```

Blue expose blob

```
defaults write com.apple.dock wvous-floater -bool true
```

Interesting "Show Desktop" option

```
defaults write com.apple.dock wvous-olddesktop -bool false && killall Dock
```

Silent volume changes (no beeps): hold Shift while turning volume up/down or unmuting

Genie freeze: shift+orange button and at the same time, `killall Dock`

Expose keyboard control: use arrows to move around windows (in All Windows view), Tab moves across applications

Safari bookmarks using Javascript

make `javascript:window.open("http://www.google.com")` into a bookmark

Startup items

<http://www.osxfaq.com/Tutorials/LearningCenter/HowTo/Startup/index.ws>

Kernel hacking (list of OS X command line instructions)

<http://www.kernelthread.com/mac/osx/tools.html>

Mac OS X Shortcuts

force quit app	cmd + option + esc
copy item	option + drag
switch to next window	cmd + ~
eject (selected media)	cmd + Y
close all Finder windows	cmd + option + W
æ	option + apostrophe
fi	shift + option + 5
fl	shift + option + 6 (fl)
screenshots (area)	cmd + shift + 4
screenshots (window)	cmd + shift + 4 + caps lock
screenshots (screen)	cmd + shift + 3
boot from CD	C (on startup)
boot from another volume	cmd + option + shift + del (on startup)
list of bootable options	option (on startup)
single user mode	cmd + S (on startup)
scrolling text boot	cmd + V
target mode	T (on startup)
eject CD	hold mouse button (on startup)
safe boot mode	shift (on startup)

Useful tcsh aliases

```
alias bc bc -l
alias rm ~/saferm.tcsh
alias mv mv -i
alias cp cp -i
alias c clear
alias ll '/bin/ls -la'
alias ls ls-F
alias grep grep -ns
alias d 'pushd \!*'
alias . 'popd'
alias t 'echo "^[]0;\!*^G"'
alias ip "ifconfig | grep 'broadcast' | awk '{print \"'$3'\"}'"
alias dict 'curl dict://dict.org/d:\!*'
alias vim 'echo "^[]0;\!*^G"; \vim \!*'

```

Useful tcshrc commands

autolist setting	set autolist = ambiguous
enable color	set color
complete case sensitive	set complete = enhance
correct commands	set correct = cmd
set history size	set history = 2000
set history (+ merge)	set savehist = (2000 merge)
beep when match ambiguous	set matchbeep=never
beep when no match	set matchbeep=nomatch
no beep	set nobeep
beep command	set beepcmd
periodic command	alias periodic cmd-to-invoke-periodically
set periodic command period	set tperiod=mins
pre-command alias	alias precmd cmd-to-invoke-before-each-other-command
post-command alias	alias postcmd cmd-to-invoke-after-each-other-command
go back in history	Esc, P
expand globbing	Type a glob, and then Ctrl + X, * or Ctrl + X, G
current, old working directory	\$cwd, \$owd
enter dirs without 'cd'	set implicitcd
cd path	set cdpath= (~ ; ~/Sites)
silent pushd	set pushdsilent
check if variable defined	\$?name (becomes 1 if defined)
escape from aliases	\ls
rehash \$PATH	rehash
show all bound keys	bindkey, bindkey -l
bind to string	bindkey -s ^xa "Me" (literal '^')
bind to command	bindkey -c ^xa "ls"
check bindkey	bindkey -l
unset variable \$autologout	unset autologout

if flags

read, write, execute access	-r -w -x
existence	-e
directory	-d
size	-Z

tcsh prompt

```
set prompt = "%{\033[1;31m%}[air:\!] %c9>%{\033[0m%} "
```

Options:

• %B %c4 %b	bold on, display current directory up to 4 levels, bold off.
• %d %D %w %t	day, date, month, time
• %n %M \n %#	user name, host name, new line, standard prompt
• %/ %~	cwd (cwd with ~'s)
• %M %m	hostname (short)
• %S (%s) %B (%b) %U (%u)	standout, bold, underline
• %t %@ %T %p %P	time
• %n	username
• %d %D %w %W %y %Y	weekday, day, month, year
• %\$	expand variable
• %c4	cur dir up to 4 levels

ls autocompletion

```
setenv LS_COLORS 'no=00:fi=00:di=01;34:ln=04:ex=01;32:*.jpg=33:*.gif=33:*.php↵  
=31:*.c=31:*.cpp=31:*.h=31:*.pl=31:*.pdf=33:*.ps=33'
```

File types:

- no Normal (non-filename) text
- fi Regular file
- di Directory
- ln Symbolic link
- pi Named pipe (FIFO)
- so Socket
- bd Block device
- cd Character device
- ex Executable file
- mi Missing file
- or Orphaned symbolic link

Color numbers:

- 0 default Terminal colors
- 1 brighter
- 4 underline
- 5 flashing
- 30..37 foreground: black, red, green, dark yellow, blue, magenta, cyan, grey
- 40..47 background

Command History

previous command	!!
n-th command	!n
n-th-to-last command	!-n
most recent starting with str	!str
most recent containing str	!?str?
replace old by new in last cmd	^old^new^
argument history	{history prefix}:{argument suffix}, e.g. !!:*
all arguments	*
first argument	~
last argument	\$
n-th argument	n (0=command name)
args n thru m	n-m (-m for 0 through m)
n thru last	n*
n thru penultimate	n-

Shell redirection

shell with advanced redirect	sh -c 'ls 2> err'
force redirect	cat > output
take input till 'EOF'	cat <<EOF
take till 'EOF' + ignore tabs	cat <<-EOF
redirect stderr	cat 2> errput
redirect stdout+stderr to file	cat > file 2>&1
pipe stdout+stderr to command	cat 2>&1 grep
redirect output, error to 2 files	(ls tom > list) >& list.err

tcsh **operations on Variables**

setting	<code>set fn="/Users/tommy/Documents/blah.doc"</code>
accessing	<code>\$fn (-> /Users/tommy/Documents/blah.doc)</code>
path	<code>\$fn:h (-> /Users/tommy/Documents)</code>
file	<code>\$fn:t (-> blah.doc)</code>
extension	<code>\$fn:e (-> doc)</code>
all but extension	<code>\$fn:r (-> /User/tommy/Documents/blah)</code>
can combine	<code>\$fn:t:r (-> blah)</code>
substitution	<code>\$fn:t:s/.doc/.txt (-> blah.txt)</code>
display all variables	<code>set</code>
interpolation	<code>{\$fn}</code>
PID of current shell process	<code>\$\$</code>
multi-valued variable	<code>set tom=`ls`</code>
number of elements	<code> \$#tom</code>
create array	<code>set words=(tom bob sam)</code>
1-based element of the array	<code>\$tom[1]</code>
multi-valued subarray	<code>\$tom[2-4]</code>
check if variable set	<code>\$?tom</code>
increment variable	<code>@ tom++</code>

bash **basics**

setting variables	<code>n1=3</code>
evaluating arithmetic	<code>n=\$((n1+n2+n3))</code>
declaring	<code>declare -i n; n=5*5</code>
declaring with defaults	<code>declare -i i=\${3:-4} (3rd cli or 4=default)</code>
test and set	<code>echo \${test2:=4} (set to 4 if not set)</code>
substrings	<code>test='hello'; echo \${test:3:2} (->lo)</code>

grep **basics**

Each character matches itself except for the special characters (which can be escaped with `\`):

`+?.*^$()[]{}|\`

Syntax:

<code>.</code>	Matches an arbitrary character
<code>(...)</code>	Groups a series of pattern elements to a single element
<code>^</code>	Matches the beginning of the target
<code>\$</code>	Matches the end of the line
<code>[...]</code>	Denotes a class of characters to match.
<code>[^...]</code>	negates the class
<code>(... ...)</code>	Matches one of the alternatives.
<code>(?: regex)</code>	Grouping without back-references. (Does not store results in variables <code>\$1..\$9</code>)
<code>+</code>	Matches the preceding pattern element one or more times
<code>*</code>	Matches the preceding pattern element zero or more times
<code>?</code>	Matches the preceding pattern element zero or one times
<code>{n,m}</code>	Denotes the minimum n and maximum m match count. {n} means exactly n times; {n,} means at least n times; {n,m} means between n and m times.
<code>\w</code>	Matches word characters, i.e. alphanumeric including underscore
<code>\W</code>	matches non-alphanumerics.

<code>\s</code>	Matches whitespace.
<code>\S</code>	matches non-whitespace.
<code>\d</code>	Matches digits.
<code>\D</code>	matches non-digits.
<code>\b</code>	Matches word boundaries.
<code>\t</code>	tab
<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\f</code>	formfeed
<code>\0XX</code>	octal
<code>\xXX</code>	hex
<code>\$1..\$9</code>	Refer to matched subexpressions grouped with (...)

Regular expression modifiers:

<code>g</code>	Matches as many times as possible
<code>i</code>	Case-insensitive matching
<code>m</code>	Treats the string as multiple lines
<code>s</code>	Treats the string as a single line
<code>x</code>	Comments and whitespace (for readability)

Matching, Searching and Replacing, Transliterating in perl

`[expr=~][m]/pattern/modifiers`

Returns true or false depending on whether or not the pattern matched.

Searches `expr` (default: `$_`) for the pattern.

If you prepend the `m` you can use almost any pair of delimiters instead of the slashes.

Useful if you are going to have lots of slashes in your pattern—avoids having to escape them all.

Most common alternative delimiters are `{}`, `[]`, and `##`.

`[$var=~]s/pattern/newtext/modifiers`

Searches the string `var` (default: `$_`) for a pattern, and if found, replaces that part with the replacement text. It returns the number of substitutions made. Almost any delimiter may replace the slashes.

If bracketing delimiters are used pattern and newtext may have their own delimiters, e.g., `s(foo)[bar]`

`$var=~]tr/searchlist/replacementlist/modifiers`

Transliterates all occurrences of the characters found in the search list with corresponding characters in the replacement list. The `d` modifier deletes all characters found in the search list that do not have a corresponding character in the replacement list.

How to... / Useful Commands

(Mac OS X) If you ever happen to want to link against installed libraries in a given directory, LIBDIR, you must either use libtool, and specify the full pathname of the library, or use the `-LLIBDIR` flag during linking and add LIBDIR to the DYLD_LIBRARY_PATH environment variable during execution

Tunneling through an https proxy

Download corkscrew: <http://freshmeat.net/projects/corkscrew/>

Download ntlm authorization proxy 0.9.9 (ntlmmaps): <http://ntlmmaps.sourceforge.net/>

In .ssh/config, type:

```
ProxyCommand /usr/local/bin/corkscrew 127.0.0.1 5865 %h %p
Port 443
```

SSH Tunneling

```
ssh -C -N -L 5900:localhost:5900 mydomain.com
```

```
ssh -nNT -R 1100:local.mydomain.com:1100 remote.mydomain.com
```

Spoof mac address

```
sudo ifconfig en0 lladdr 02:01:02:03:04:05
```

(Mac OS X) Create encrypted disk images

```
hdiutil create -encryption -type SPARSE -stdinpass -srcfolder secrets foo.dmg
```

xargs – a superuseful command to convert text to arguments. For example, to add all files recursively in the "images" folder, type:

```
find images | grep -v -E '\.svn' | xargs svn add
```

enscript -- format printing of text files. For example, for landscape, one column:

```
enscript -1 --pretty-print -G --output=cs141-mips.ps -r --color
```

```
enscript -G2rE -U2 -o foo.ps foo.c
```

Link code

```
gcc -c bluephonekeygen.c && ld /usr/lib/libSystem.B.dylib /usr/lib/dyld↵
  /usr/lib/libcrypto.0.9.7.dylib /usr/lib/crt1.o bluephonekeygen.o -lc
```

Find files with a word in them

```
find ~/ -type f -exec grep -l <word> {} \;
```

Prompt for a password

```
stty -echo; echo -n "Password: "; set tom = $<; stty echo; echo $tom
```

(Mac OS X) attach and install dmg files programmatically

```
hdiutil attach ActivePerl-5.8.x.x.x.dmg -verbose
```

```
cd /Volumes/ActivePerl-5.8.x
```

```
sudo installer -pkg ActivePerl-5.8.x.pkg -target /
```

set bootability/startup disk options	bless
mount device (/dev/disk2, disk1s9)	diskutil mountDisk <device>
locate a file (all with extension)	locate myfirst.c, locate .txt
update locate	/usr/libexec/locate.updatedb
copy, paste to Clipboard (OS X)	pbcopy, pbpaste
open applications (OS X)	open -a application
copy all files + hidden etc. (OS X)	ditto
kill by name (OS X)	killall psname
screen capture (OS X)	screencapture
read configuration (OS X)	defaults read com.apple.terminal
set configuration (OS X)	defaults write com.apple.blah Rows 44
cron jobs	/etc/crontab
Sort by field after \$, ignore blanks	sort -t\$ -k2 -b
reset terminal	tput reset
pids of daemons	/private/var/run/*.pid
property list utility to convert (OS X)	plutil
system profiler (OS X)	AppleSystemProfiler
software update (OS X)	softwareupdate
energy saver (OS X)	pmset -a dim 5 spin 5 sleep 20 womp 1
eject, mount, unmount (OS X)	disktool
sort used to merge files	sort filea fileb
paste files, line by line	paste file1 file2
translate (Mac2Unix)	tr \\r \\n <in.txt >out.txt
pids	/var/run/
xargs using ASCII 0 as separator	xargs -0
image processing	sips --help
wait for a job	wait %n
evaluate expression	expr
parse input options	getopt
make temporary files	mktemp
lock file	lockfile
interact with burners	drutil
notify when job complete (asynch)	notify %job
match any one character	ls b[aei][0-9]g (be6g, ba2g...)
autoupdating tail	tail -f output.txt
calculator	bc
change to root/wheel	chown -R 0:0 file
substitute with sed	sed 's/blah/new/g' file.txt
delete lines sed	sed '/blah/d' file.txt
strip ^H, compress tab, expand	col -b, col -h, col -x
diff, side by side	diff -y
using printf	printf "[%s]\n" `ls`
jump to old working directory	cd - (interchangeable)
pushing / popping directory	pushd, popd
xargs – one at a time	xargs -n1
get file types	file myfirst.c (-> ASCII C text)
execute command on a remote is	rsh is02 command
execute commands in future	at 17:30 < scriptfile
simple mathematical interpreter	expr
execute cmds simultaneously	echo hello & echo bye
execute 2nd only if 1st fails	true echo bye (won't echo bye)

execute 2nd iff 1st successful
get file modification time
find files without listing all
terminal info (including size)
mounting floppy drives in UNIX
setting cross-user permissions
same as which, but unaliases
setting environment variables
user, group id
groups to which user belongs
currently logged on users
all users, tty, when logged
machine name
return hostname (is??)
return tty name
piping
iterate over elements
run after logging off
output sequences
wait
kill all processes
writing to console
copy from input to output
record everything in a script

```
true && echo bye (will echo bye)
ls -l $file | awk '{print $6, $7, $8}'
find ~/ -type f -name <filename> -print
stty -a
mount /dev/fd0 /mnt/floppy msdos
setacl -u user:<username>:<rxw>
where
setenv PATH = xxx:${PATH} (appends xxx)
id
groups
users
who
uname
hostname
tty
ps aux | grep strozek
echo {A,B}{1,2,3} : A1 A2 A3 B1 B2 B3
nohup ./mynos.pl &
seq 1 3 10 = 1\n4\n7\n10 (-f: format)
sleep 30s
kill -KILL -1
echo "Hello" > /dev/tty??
tee
script (type exit to terminate record)
```

Fun Unix Stuff

forkbomb – don't try this at home!
dialing codes
crossword cheater
display login window (OS X)
intercepting console input

```
:(){:|:&}::
/usr/share/misc/inter.phone
/usr/share/dict/words | grep -wi 't..m'
/usr/libexec/WaitingForLoginWindow
cat /dev/tty?? (tty same user's)
```

Terminal control codes

generate control code (octal)	<code>puts("\033")</code>
produce escape character (ASCII 27) <code>^[]</code>	Ctrl + V, ESC
Cursor move up	<code>^[[A</code> (one line), <code>^[[3A</code> (3 lines)
Cursor move down	<code>^[[B</code>
Cursor move left, right	<code>^[[C</code> , <code>^[[D</code>
Newline	<code>^[[E</code>
Bell	Ctrl + V, Ctrl + G
Place cursor at (3,4)	<code>^[[4;3H</code>
Backspace	<code>^[[H</code>
cr	<code>^[[J</code>
Delete to EOL	<code>^[[K</code>
Clear screen	<code>^[[L</code>
lf	<code>^[[M</code>
Down	<code>^[[N</code>
Up	<code>^[[P</code>
Swap cur and prev char	<code>^[[T</code>
Report on cursor position	<code>^[[6n</code> (returns <code>^[[y;xR</code>)
Set title bar	<code>^[]0;title^G</code>

Terminal Colors

UK console	<code>^[[CA</code>
US console	<code>^[[CB</code>
graphics console	<code>^[[C0</code>
color mode	<code>^[[37m</code> or <code>^[[3,33m</code>
0 (normal)	
1 (bold)	
2 (low intensity)	
4 (underline)	
5 (blink)	
7 (reverse background and foreground)	
8 (text off)	
30-37 text colors	
40-47 background colors	

Mouse support in xterm

enable	<code>^[[?1000h</code>
disable	<code>^[[?1000l</code>
on each press	<code>^[[Mxy</code> where (33,33) is top-left
on each depress	<code>^[[M#xy</code>

PHP Cheatsheet

File Uploads

```
<FORM ENCTYPE="multipart/form-data" (important!)>
<INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="4096">
<INPUT NAME="uploaded" TYPE="file">
</FORM>
```

`$uploaded` contains the path to the file, also `$uploaded_name`, `$uploaded_size`, `$uploaded_type`.

Cookies

HTTP_COOKIE_VARS

```
setcookie(name, value, expire, path, domain, secure)
```

Use before output sent to browser

if `expire` omitted, expires on browser close

```
setcookie(name) - delete cookie
```

open pipe	<code>\$file = popen(\$command, \$mode)</code> (can use <code>fputc</code> etc.)
open file	<code>\$file = fopen(\$command, \$mode)</code>
read/write	<code>fputs(\$file, "Hello"); \$str = fgets(\$file, 255);</code>
more read/write	<code>\$char = fgetc(\$file), fputc(\$file,\$char)</code>
read file	<code>\$array = file(\$filename)</code>
end of file?	<code>feof(\$file)</code>
file exists?	<code>file_exists(\$filename), filesize(\$filename)</code>
file time	<code>fileatime(\$file), filectime(\$file), filemtime(\$file)</code>
seeking into file	<code>fseek(\$file, offset); ftell(\$file);</code>
dir listing	<code>\$hnd = opendir(\$directory); readdir(\$hnd);</code>
env variables	<code>getenv, putenv("PATH=/local/bin;");</code>
send mail	<code>mail(recipient, subject, body, extraheaders)</code>
walk array	<code>array_walk(\$array, \$fnstring)</code> – function must be user-defined
assign many vars	<code>list(\$key, \$value) = each(\$colors) (\$colors = array)</code>
image size	<code>getimagesize(\$filename)</code> – returns array (xs, ys, type, string)

MySQL Cheatsheet

```
CREATE DATABASE name
```

```
USE name
```

```
CREATE TABLE name (col attr, ..., PRIMARY KEY (name), INDEX (name))
```

```
    index  integer not null auto_increment
```

```
    blah  integer default 0
```

```
    text  varchar (255)
```

```
    text2 text null
```

```
ALTER TABLE name ADD COLUMN name attr
```

```
ALTER TABLE name DROP COLUMN name
```

```
ALTER TABLE name CHANGE oldcol newcol attr
```

```
INSERT INTO table (col1, col2, ...) VALUES (val1, val2, ...)
```

```
UPDATE table SET col1=value1, col2=value2... WHERE ind=value
```

```
SHOW TABLES
```

```
SHOW FIELDS FROM table
```

SELECT col1, col2, ... FROM table WHERE col=value

can use OR AND, value IS NULL for null values

SELECT DISTINCT col1

WHERE col1 BETWEEN 203 AND 304

WHERE col1 IN ('a', 'b', ...), col NOT IN

ORDER BY col1 ASC, col2 DESC

LIMIT 0,5 – start from record 0, return 5 records

SELECT col1, count(*) FROM table GROUP BY col1 – counts number of col1 entries

SELECT count(*) FROM table – count number of rows

SELECT state, SUM(contribution) AS 'Total', AVG(contribution) As 'Average',

MIN(contribution) AS 'Minimum' FROM contributions GROUP BY state

SELECT * FROM companies, contacts WHERE companies.id = contacts.id