BASH CHEATSHEET

BASICS

jp@adams:~$	logged in as normal
user	
jp@adams:~#	logged in as root user
jp@adams:/tmp$	away from root directory
pwd	where you are
$PATH	environment variable
type <cmnd>	find cmnd
which <cmnd>	find cmnd in $PATH
which -a <cmnd>	return all results
apropos <term>	search manpages & descripts for term
locate <term>	search DB files for term
ls <path>	list files
ls -a <path>	unhide hidden files
ls -F <path>	
ls -l <path>	long listing
ls -L <path>	show linked file info
ls -Q <path>	quote names
ls -r <path>	reverse sort order
ls -R <path>	recurse thru subdirs
ls -S <path>	sort by file size
ls -1 <path>	short format, 1/line
ls -d .*	list just hidden files
'quotable stuff'	pass stuff w/o interpret
$-	interactivity variable

OUTPUT

echo <string>	output to terminal
echo "<string w/ spacing>"	preserve whitespaces
printf '<format>' <arg>...	formatted output
<cmnd> > file.txt	save output to path
ls -C <arg> > file.txt	save output of ls
<cmnd> 1> msgfile 2> errfile	1> = STDOUT 2> = STDERR
<cmnd> >% outfile	STDOUT & STDERR to same file
<cmnd> >> outfile	append to outfile
head n file	first n lines (default 10)
tail n file	last n lines (default 10)
tail +1 file	entire file
tail +2 file	skips first line
<cmnd> > /dev/null	throw output away
{ cmnd1; cmnd2; ... } > outfile	capture multi cmnd out
cmnd1 | cmnd2	use out1 as in2 (pipe)
cmnd1 | tee out1 | cmnd2	save out1 + use as in2
rm $(find . -name '*.txt')	use subshell to build arg
set -o noclobber	prevent file overwrites
set+o noclobber	allow file overwrites
echo thistext >| outfile	clobber file anyway

INPUT

cmnd < infile	get input from file
grep $1 <<EOF ... EOF	use heredoc to include input data inside bash script

grep $1 <<-'EOF'
    ...
    EOF
	indent here-docs
	for readability

read -p "prompt" ANSWER	capture user input
read -s -p "passwd: " PASSWD	-s: don't echo input

EXECUTING COMMANDS

$ somecmnd	run any executable
$ chmod a+x ./myscript	exec permission
$ ./myscript	now you can run it
$ echo $?	did cmnd work? 0=ok
$ cmd1 ; cmd2 ; cmd3	execute sequence
$ cmd1 && cmd2	do cmd2 if cmd1 OK
$ cmd1 & cmd2 & cmd3	run all 3 in parallel
(($?))	use exit stat as OK flag
$ nohup cmd &	run cmd as background (for long exec times)
$ cmd || printf "%b" "oops"	display error msgs

$ for SCRPT in path
do if [-f $SCR -a -x $SCR]
  then $SCR
  fi
done
	run all scripts in given directory

SHELL VARIABLES

# this is a script comment.	
: <<'END_OF_HELPDOC' ... END_OF_HELPDOC	embed userdoc in script
$ export THISVAR	export variable
$ env	all exported variables
$ set	see all vars in curnt shell
$1 or ${1}	1st cmnd line argument
$*	all cmnd line arguments
${#}	how many args?
VERBOSE=0	suppress echo to term
THISVAR=${1:-"/tmp"}	get default var value if not supplied by user
cd ${HOME:=/tmp}	set default var value

SHELL LOGIC & ARITHMETIC

TOTAL=$((X + 10 + Y *2)	basic arithmetic in script

assignment operators:

BASH CHEATSHEET

=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |=

if [cond] then … fi	branch on conditional
if [flag "$FILE"] …	file feature tests
if [flag1 $FILE -a flag2 $FILE] …	test for flag1 AND flag2
if [flag1 $FILE -o flag2 $FILE] …	test for flag1 OR flag2

file feature test flags:
-b (is file a block special devices)
-c (is file a character special)
-d (is a directory?)
-e (file exists?)
-f (regular file?)
-g (set-group-ID-bit set?)
-h (symbolic link?)
-G (file owned by effective group id?)
-k (sticky bit set?)
-L (symbolic link?)
-O (file owned by effective user id?)
-p (named pipe?)
-r (readable?)
-s (size > 0?)
-S (socket?)
-u (set-user-id bit set?)
-w (writable?)
-x (executable?)

if ["$VAR1" -eq "$VAR2"]	test for equality (num)
if ["$VAR" = "$VAR2"]	test for equality (str)

numeric comparisons:
-lt, -gt (less than, greater than)
-le, -ge (less than or equal, greater than or equal)
-eq, -ne (equal, not equal)
string comparisons:
<, > (less than, greater than)
<=, >= (less than or equal, greater than or equal)
=, != (equal (== also works), not equal)

shopt -s extglob	enable globbing
if [["${VAR}" == *.jpg]]	test for pattern matches
if [["$[VAR]" =~ "(<regexp>)"	test for regexp matches

pattern match types:
@(…) (only one match)
*(…) (zero or more matches)
+(…) (one or more matches)
?(…) (zero or one match)
!(…) (not this match, but anything else)

while ((A < B)) do … done	loop while condition met
for ((i=0 ; i<10 ; i++)) do … done	loop with a count
for n in $(seq 1.0 0.01 1.1) do … done	FP loops
case $VAR in	case statements

```
case $VAR in
        *.gif dogif $FN ;;
        *.png dopng $FN ;;
        *) doallelse $FN ;;
esac
```

TOOLS I

$ grep <str> *.c	find string matches
$ grep -l <str> *.c	return filenames
$ grep -q <str> *.c	return True/False result
$ grep -i <str> *.c	case insensitive
$ grep -i <str> *.c | grep -v <str2>	pare down results
$ zgrep <str> *.zip	grep zipfiles

TOOLS II

$ <cmd> | sort	sort results
$ <data> | sort -n	sort as numerics
$ <stuff> | sort -r	reverse order
$ <stuff> | sort --ignore-case	case-insensitive
$ <ipaddrs> | sort -t. -n +3.0	sort using separator chr
$ <stuff> | cut -12-15	cut out columnar subset
$ <stuff> | sort -u	remove duplicates
$ tr <before> <after>	translate characters
$ tr -d '\r' <file.dos >file.txt	DOS to Linux file format
$ wc datafile	word count
$ wc -l datafile	line count
$ wc -w datafile	words only
$ wc -c datafile	character count

FINDING FILES

$ find . -name '*.mp3'	find your mp3 files
$ find . -iname '*.mp3'	case-insensitive
$ find . -name '*.jpg' -mtime +90 -print	find by date
$ find . -type d -name "*java*' -print	find by type
$ find . -size +3000k -print	find by size
$ find -i thistext *.txt	find by content
$ locate thistext	find by content (fast)

FUNCTIONS, TRAPS, ALIASES

$ nohup mydaemon 0<&- 1>/dev/null 2>&1 &	'daemonize' a script
source $HOME/myprefs.cfg	read global config file
function max () {…}	define function
max 128 256	call function
trap -l	list trap handlers (linux)
trap 'echo "aha!" ' ABRT EXIT	define trap handler
alias ls='ls -a'	redefine common cmnd

PARSING

TODO

SECURITY

TODO

ADVANCED OPTIONS

$ hexdump -C filename	see output in hex mode
TODO	

CONFIGURATION / CUSTOMIZATION

TODO

HOUSEKEEPING

$ unzip '*.zip'	unzip multiple zip files
$	

PRODUCTIVITY

$ pushd /var/log/here	push wd to dirctry stack
$ popd	pop directory stack
$!!	repeat last cmnd
$!!:s/H/A	repeat last cmnd edited
$ ls myfile<tab>	autofind path for myfile
$ echo *	re-display last result

TIPS & GOTCHAS

$ chmod a+x script.sh	set execute permissions
$ bash ./ismyscriptok.sh	using bash explicitly
#!/bin/bash	usual shebang line

REFERENCES