Linux Cheat Sheet

BASH Internals

syntax, use 'echo' + keyword

```
$$ - PID of current shell
$0 - show shell name
$! - PID of last background cmd
$? - exist status of last cmd
$_ - previously created dir (mkdir foo && cd $_)
$@ - show all command's parameters
$# - show # of arguments passed to command
$*  - All arguments passed to command
$1  - first argument passed to command
!! - run previous command
-eq - math equal (int)
-ne - math not equal (int)
-lt - math less than
-le - math less than or equal
-gt - math greater than
-ge - math greater than or equal
-z - string is 0 length (null)

-n - string is not 0 length (not null)

        if [ -n "{var}" ] # if not null then..

-nt - newer than (file or object time)
(-r,-w,-x) - if object is readable,writable,exec
```

set default value if parameter is null or empty

```
var1=""
echo "${var1:-abc}"
>> abc
```

error out if parameter is null or empty

```
var1=""
echo "${var1:?}"
>>   bash: var1: parameter null or not set
```

check if character is inside a string

```
str="1,2,3,4-5"
[[ "$str" == *-* ]] && echo "has dash"
```

source file in same directory as caller script

```
source "$(readlink -f $0 | xargs dirname)/shared.txt"
```

Arrays & Dictionaries

Simple Array

```
Fruits=('Apple' 'Banana' 'Cherry')
echo ${Fruits[1]}  ## Banana
echo ${Fruits[@]:1:2}  ## range from 2nd to 3rd element,  banana cherry
```

add an element to array

```
Fruits+=('Watermelon')
echo ${Fruits[@]}  ## "apple, banana, cherry, watermelon",   @ = show all
elements in array
```

Loop over an array

```
arr=(apples oranges tomatoes)
# Just elements.
for element in "${arr[@]}"; do
    printf '%s\n' "$element"
done
```

create array from string with delimeter

```
str="1,2,3,4,5"
IFS="," read -a ARR <<< $str
echo "${ARR[@]}"
>>> 1 2 3 4 5
```

Key/Value pairs (associative Array, aka Hash, Dictionary)

read in config file, check array of key/val pairs to make sure parameters are set

```
$config = "/etc/file.conf"
declare -A myList=( [first]=
                    [last]=
                    [age]=
                  )

for param in "${!myList[@]}"; do
    value=$(grep ^$param $config)
    var[$param]=$value

     if [ -z ${myList[$param]} ];
     then
        echo "param $param is not set"; exit 1
     fi
done
```

Dictionary/Hash in bash (parse IP + Port hash, netcat to each IP and port)

```
readonly connections=(

'A, 205.209.202.37, 7755'
'B, 205.209.202.1, 8899'
'C, 205.209.202.21, 4578'

)

function nctest(){
  local name ip port
  for fields in ${connections[@]}
  do

    # strip whitespace
    fields="$(echo -e "${fields}" | tr -d '[:space:]')"

    IFS=$',' read -r typ name ip port <<< $fields

    conn=$(nc -zv -w 2 $ip $port 2>&1 | grep 'Connection refused' )

    if [[ -z "${conn}" ]]
    then
      echo "[$name] nc $ip $port OK"
    else
      echo "[$name] nc $ip $port REFUSED"
    fi

    echo "--------------"
  done
}

nctest
```

basic dict

```
declare -A sounds
sounds[dog]="bark"
sounds[cow]="moo"
sounds[cat]="meow"
echo ${sounds[dog]}  # bark
```

Iterate keys and values

```
for val in "${sounds[@]}"; do
  echo $val
done
```

```
for key in "${!sounds[@]}"; do
  echo $key
done
```

Math

calculate value

```
echo $((35+15))
50
```

generate random number 0 to 500

```
$((RANDOM % 500))
```

use calculator

```
echo "12+3" | bc   # 15
echo "10^2" | bc  # 100
echo "10/2" | bc  # 5
```

add 2 variables

```
sum=$(($var1+$var2))
```

Loops and Conditionals

for loop with Range

```
for loop in {1..50}; do echo "processing $loop"; sleep 2; done
> processing 1
> processing 2
etc
```

Range with step size, count every 5

```
for i in {1..50..5}
```

or use a 'seq' operand

```
for i in $(seq 1 5); do echo $i; done
1
2
3..etc
```

sequence with step size,

```
for i in $(seq 1 5 30); do echo $i; done
1
6
11
16
21
26
```

Counter loop

```
for ((i=0; i<100; i++)); do echo $i; done
```

loop thru directories and grep something from config files

```
for i in $(ls); do grep 'something' $i/*.conf ; done
```

Case statement (check input params)

```
case $key in
-u| -username | --username)
   UNAME="$2"
 shift
 ;;
-pw| -password | --password)
   PASSWORD="$2"
 shift
 ;;
-p| -profile | --profile)
   PROFILE="$2"
 shift
 ;;
*)
 echo "Unknown Option"
exit 1
```

```
case $env in
     "test")
          username="testuser"; password="testPW";;
     "prod")
          username="produser"; password="prodPW";;
esac
```

If statement

```
if [[ -z "$string" ]];
then
  echo "String is empty"
elif [[ -n "$string" ]]; then
  echo "String is not empty"
fi
```

Variable Conditionals

```
[[ -z STRING ]] Empty string
[[ -n STRING ]] Not empty string
[[ STRING == STRING ]] Equal
[[ STRING != STRING ]] Not Equal
[[ NUM -eq NUM ]] Equal
[[ NUM -ne NUM ]] Not equal
[[ NUM -lt NUM ]] Less than
[[ NUM -le NUM ]] Less than or equal
[[ NUM -gt NUM ]] Greater than
[[ NUM -ge NUM ]] Greater than or equal
[[ STRING =~ STRING ]] Regexp
(( NUM < NUM )) Numeric conditions
[[ -o noclobber ]] If OPTIONNAME is enabled
[[ ! EXPR ]] ## "Not something"
[[ X ]] && [[ Y ]] ## X And Y
[[ X ]] || [[ Y ]]  ## X Or Y
```

File conditionals

```
[[ -e FILE ]] Exists
[[ -r FILE ]] Readable
[[ -h FILE ]] Symlink
[[ -d FILE ]] Directory
[[ -w FILE ]] Writable
[[ -s FILE ]] Size is > 0 bytes
[[ -f FILE ]] File is type "file"
[[ -x FILE ]] Executable
[[ FILE1 -nt FILE2 ]] 1 is more recent than 2
[[ FILE1 -ot FILE2 ]] 2 is more recent than 1
[[ FILE1 -ef FILE2 ]] Same files
```

Case / Switch

```
case "$1" in
  start | begin)
     service start
  ;;
  stop | kill)
    service stop
  ;;
  *)
  echo "usage: $0 {start|stop}
  ;;
esac
```

GREP

search for joe in names.txt

```
grep 'joe' /names.txt
```

search for 'joe' in directory dir1

```
grep 'joe' /dir1 -r
```

search for 'joe' in dir1 and follow symlinks

```
grep 'joe' /dir1 -R
```

search only files that match

```
grep 'joe' /dir1 -l
```

search only files that dont match

```
grep 'joe' /dir1 -L
```

case insensitive

```
grep 'JOE' /dir1 -i
```

add colo

Debugging & Test

enable line by line processing output in script
```
set -x
```

verbose output only if debug flag is set
```
debug=1
test $debug -gt 0 && echo "var is $var"
```

test if file exists, if not, exit with error
```
test -f "${config}" || { echo "${config} not present, exiting.."; exit 1; }
```

check if parameters are set and not empty, exits out w error if not set
```
err_msg="[ERROR] parameter is not set or empty value:"
myParam=${1:?"$err_msg You have an error, missing myParam"}
```

check segfault core output
```
gdb $(which python) core.python.$date
gdb <my binary> <path to core>
```

Shell Cmds

run python inside Bash with arguments
```
function print_hello {
  NAME="${1}" python - <<END
  import os
  print("Hello there %s" % os.environ['NAME'])
  END
}

print_hello Joe
```

get variable from a json dump using python
```
URL=$(echo ${URL} | python -c 'import sys,json; print json.load(sys.stdin)["url"]')
```

colorize Bash prompt (insert into ~/.bashrc)
```
export PS1="[\[\e[31m\]\u\[\e[m\]\[\e[33m\]@\h\[\e[m\]:\W]$ "
```

Root PS1
```
export PS1="[\[\e[30;41m\]\u\[\e[m\]\[\e[33m\]@\h\[\e[m\]:\W]$ "
```

generate a random password
```
date +%s | sha256sum | base64 | head -c 8; echo
```

remove all empty directories
```
find . -type d -empty -delete
```

copy permissions on file1 to file2
```
chmod --reference file2 file1
```

remove all but specific file
```
rm -f !(theFile.txt)
```

remove files that dont match a specific extension
```
rm !(*.xls|*.slsx|*.csv)
```

find duplicate files (check file hash)
```
find -not -empty -type f -printf "%s
" |  sort -rn |  uniq -d |  xargs -I{} -n1  find -type f -size {}c -print0 |
xargs -0  md5sum |  sort |  uniq -w32 --all-repeated=separate
```

run a command as another user
```
runuser -l joe -c 'whoami'
```

Regex & String Manipulation

convert uppercase files to lowercase
```
rename 'y/A-Z/a-z/' *
```

capitalize 1st letter
```
var="wunderbar"
echo  ${var^}
```

capitalize entire word
```
echo ${var^^}
```

get # of characters in variable
```
var="milkshake"
echo ${#var}
9
```

Check if Word is in a String
```
[[ "$string" == *"$word"* ]] || echo "word not in string"
```

strip off the last character from a string,
```
var="Banana"
echo ${var%?}  // Banana
```

Get value between 2 delimiters,
```
grep ExecStart bitbucket.service | awk -v FS="(bitbucket/|/bin)" '{print $2}')
```

extract filename from a path
```
echo /somedir/blah/postgresql96-9.6.5.x86_64.rpm | awk '{match($1, "[^/]*$",
a)}END{print a[0]}'

postgresql96-9.6.5.x86_64.rpm
```

search for a pattern in all files
```
grep -RnisI "My cat*" /var/log/*
```

Insert string after a delimeter, save in place (insert "dog" after "cat")
```
tmpfile=$(mktemp)
awk '/cat/ { print; print "dog"; next}1' pets.txt > $tmpfile && mv -f $tmpfile
pets.txt
```

String manipulation
```
STR="HELLO WORLD!"
echo ${STR,} #=> "hELLO WORLD!" (lowercase 1st letter)
echo ${STR,,} #=> "hello world!" (all lowercase) STR="hello world!"
echo ${STR^} #=> "Hello world!" (uppercase 1st letter)
echo ${STR^^} #=> "HELLO WORLD!" (all uppercase)

# Substitution

VAR="beachball"

${VAR%suffix}   # Remove suffix
${VAR#prefix}   # Remove prefix
${VAR%%suffix}  # Remove long suffix
${VAR##prefix}  # Remove long prefix
${VAR/beach/basket}  # Replace first match >> basketball

## Length of string
echo ${#VAR}  # 9
```

grep for multiple strings
```
ls -la /home | grep -v "joe\|fred\|bob"
```

another way,
```
cat /etc/passwd | egrep -v '^(root|halt|sync|shutdown|adm|bin|daemon)'
```

get last character of a string
```
permissions="775"
echo "{permissions: -1}"
```

get 2nd and 3rd character from a string
```
str="boris"
second=$(echo $str | head -c 2 | tail -c 1)
third=$(echo $str | head -c 3 | tail -c 1)
```

check if word is in a string
```
str="sun is shining"
[ -z "${str##*'shining'*}" ] && echo "contains word!!!"
```

break down string using delimeter
```
str="deny=5"
key=${str%%'='*}  # key is 'deny'
val=${str##*'='}  # val is 5
```

Cut command

breakdown string by fields
```
string="/mnt/hc/home/user"
echo $string | cut -d'/' -f2   ## mnt
echo $string | cut -d'/' -f2,3,4  ## mnt hc home

# from 2nd field to end of string
echo $string | cut -d'/' -f2-  ## mnt/hc/home/user
```

JSON & YAML

JQ - json parser

show all values in PP format
```
jq . file.json
```

PP api output
```
curl example.org/api/v1/users | jq .
```

show specific key
```
json={\"name\":\"bob\", \"age\":23}
echo $json | jq '.name'
```

parse array key for specific value
```
echo $json | jq '.values[].title'
```

select multiple properties of 1st element
```
jq '.[0] | { _id, email }' file.json
```

delete key
```
cat file.json
{ "name": "joe", "age": 23, "user-name": "j123" }

delete Name key
jq 'del(.name)' file.json > file2.json

delete key with dashes
jq 'del(."user-name")' file.json > file2.json
```

Convert YAML to JSON - 1 liner
```
python -c 'import sys, yaml, json; json.dump(yaml.load(sys.stdin), sys.stdout,
indent=4)' < file.yaml > file.json
```

Functions

basic function
```
myFunc() {
   echo "hello $1"
}

myFunc "bob"
```

Kill

```
kill -1 PID # SIGHUP, shutdown proc + restart
kill -2 PID # TERM, same as control+c
kill -3 PID # CORE, stop proc, create a core dump
kill -9 PID # SIGKILL, kill unresponsive proc, dirty kill
kill -11 PID # SIGSEGV, create core dump on segmentation fault, useful for
misbehaving procs
kill -15 PID # TERM, default kill flag, same as "kill PID"
```

Various

add commands alias

```
alias ls='ls -lta --color=auto
```

Run a specific cmd from history

```
history
120 cat /var/log/messages
121 vi /etc/hosts
!120
## will show /var/log/messages
```

redirect std output to both file and screen

```
program [arguments...] 2>&1 | tee outfile
```

Get Date in specific format

```
echo $(date +%Y%m%d_%H%M%S)
```

Parameter Expansion

```
name="John"
echo ${name}
echo ${name/J/j} #=> "john" (substitution)
echo ${name:0:2} #=> "Jo" (slicing)
echo ${name::2} #=> "Jo" (slicing)
echo ${name::-1} #=> "Joh" (slicing)
echo ${name:(-1)} #=> "n" (slicing from right)
echo ${name:(-2):1} #=> "h" (slicing from right)
echo ${food:-Cake} #=> $food or "Cake"

length=2
echo ${name:0:length}  ## Jo

file="/mnt/dir/abc.cpp"
echo $(basename $file)  # abc.cpp  (basepath)
echo ${file##*/} # foo.cpp (basepath)
echo ${file%.*}  ## /mnt/dir/abc
echo ${file##*.} # cpp (extension)

echo ${STR#*/} # path/to/foo.cpp
echo ${STR##*/} # foo.cpp
echo ${STR/foo/bar} # /path/to/bar.cpp


## set default values

${FOO:-val} ## $FOO, or val if not set
example: port=${1:-22} # if port not set via argument, make it 22

${FOO:=val} ## Set $FOO to val if not set
${FOO:+val} ## val if $FOO is set
${FOO:?message} ## Show error message and exit if $FOO is not set
```

File operations

create 25 new files from one command, use: {1..X}

```
touch myfile{1..25}
```

get file extension

```
file=superman.jpg
name=${file%.*}   # superman
ext=${file#.*}  # jpg
```

delete all files that dont match an extension

```
rm !(*.foo|*.bar|*.baz)
```

delete files from search

```
find . -name '*.pyc' -delete
```

grep 5 lines above and below a certain value

```
cat employees.txt | grep -A 5 -B 5 'Mr. Jones'
```

remove all blank lines from a file

```
grep . file1 > file2
```

read in a file

```
< file.txt | while read line; do
  echo $line
done
```

generate 1GB empty file

```
dd if=/dev/zero of=testfile count=1024000 bs=1024
```

or

```
fallocate -l 1GB testfile
```

create a random large 200MB file,

```
dd if=/dev/urandom of=file.txt bs=2075200 count=100
```

generate 10mb file with random text

```
base64 /dev/urandom | head -c 10000000 > testfile
```

VIM

delete all lines from file

```
:1,$d
```

search for all instances of string 'horse'

```
escape key
/horse
press 'n' to move to next occurence
```

vi a file on remote server

```
vi scp://user@<hostname>//etc/hosts
```

SSH

file permissions

```
/home/user = 700
/home/user/.ssh = 700
/home/user/.ssh/id_rsa = 600
/home/user/.ssh/id_rsa.pub = 644
/home/user/.ssh/authorized_keys = 600
/home/user/.ssh/known_hosts = 644
```

troubleshoot auth errors

on target (where youre trying to ssh into), start SSH on different port, debug mode

```
/usr/sbin/sshd -d -p 2222
```

on client, connect to target

```
ssh user@target -p 2222 -vvv
```

SSH Shuttle

```
pip3 install sshuttle
```

route all connections to 172.31.23.156 via "server B"

```
sshuttle -r user@<server B IP> 172.31.23.156
```

all connections will now be going via remote IP, encrypted
to route ALL connections, use 0/0

```
sshuttle -r user@serverB 0/0
```

proxy connections for a specific website,via jump host serverB, send to background

```
serverA> nohup sshuttle -r serverB `dig +short www.somesite.com | sed "/[^0-9\.]/d"
| xargs -n1 -I '$' echo -n '$/32 '` 2>&1 &
```

pass a custom SSH key to sshuttle

```
sshuttle --dns user@host <IP range> --ssh-cmd 'ssh -i /home/user/priv_key'
```

use SSH as a web proxy

```
ssh -D 8080 username@proxyHost
```

set browsers proxy option to 127.0.0.1:8080, all browsing requests will go via proxyHost

TMUX

start tmux session

```
tmux
```

reattach to session after broken connection

```
tmux ls
  0: 1 windows (created Tue Aug 23 12:39:52 2011) [103x30]
tmux attach -t 0
```

delete session

```
tmux kill-session -t 0
```

check what pub key matches the priv key

```
ssh-keygen -y -e -f ~/.ssh/id_rsa
```

add a new SSH key and copy the public key to remote known_hosts file

```
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub | ssh user@hostname 'cat >> .ssh/authorized_keys'
```

run a command on remote host

```
ssh servername cmdname
```

connect to an unreachable server B (port 2345) via SSH hop over reachable server A

```
ssh user@serverA -L 6789:serverB:2345 -f -N  (localhost:6789 = serverB:2345)
```

Port Tunneling via SSH

(port 1200 is unreachable from server A, connect to it via localhost:1300 via SSH to server B

```
user@serverA> ssh -L 1300:localhost:1200 serverB -fN
```

or via SSH Jumping
A > B > C (B has to have AllowTCPForwarding=yes in *sshd_config*)

```
A> ssh -J user@B user@C
```

Proxy to an unreachable server via reachable
A can talk to B
B can talk to C
A cant talk to C (but needs to)

```
user@serverB> ssh -L 0.0.0.0:9222:serverC:22 (will SSH into C)
user@serverA> ssh serverB -p 9222 (will ssh you into C)
```

setup SSH Sockets

```
mkdir ~/.ssh/sockets
```

```
vim ~/.ssh/config
UseRoaming no
TCPKeepAlive yes
ServerAliveInterval 15
ServerAliveCountMax 6
Host *
Compression yes
ControlMaster auto
ControlPath ~/.ssh/sockets/%r@%h:%p
ControlPersist yes
ControlPersist 600

Host nycweb1
Hostname 192.168.10.2
User root
IdentityFile ~/.ssh/id_rsa
```

show fingerprint of a public key file, useful to track down /var/log/secure messages to see who logged in

```
ssh-keygen -lf /home/user/.ssh/authorized_keys | grep <fingerprint> (looks like
SHA256:zZUd2W)
```

Use a hop server to access a unreachable host, add to ~/.ssh/config

```
Host <target-host>
  ProxyCommand ssh -q -W %h:%p <hop-host>
```

Memory / Diagnostics

Debian - CPU and Mem
```
lshw -html > /tmp/specs.html
```

Show Hardware information
```
inxi -Fxzd
```

Fedora - CPU and Mem
```
cat /proc/cpuinfo
cat /proc/meminfo
lspci -v
```

Show memory usage
```
free -m
```

Show processes by memory usage
```
ps aux | awk '{print $6/1024 " MB\t\t" $11}' | sort -n
```

show actual Memory information (RSS, memory address ,etc)
```
pmap -p <PID>
```

show USB info
```
lsusb -v
```

show size of folder
```
du -sh
```

drop all memory caches
```
echo 1 | sudo tee /proc/sys/vm/drop_caches > /dev/null
```

Swap

clear swap space
```
swapoff -a  (wait till clears)
swapon -a
```

SMEM (Memory usage profiler)

check which user/proc is using swap by %
```
yum install smem
smem -t -p -s swap
```

show memory usage just for my user
```
smem -u
```

show memory usage by user
```
smem -u joe
```

show memory usage by proc
```
smem -p firefox
```

show memory by RSS, PSS, order by Columns

(RSS, resident set size=portion of memory in RAM, rest in swap)

(PSS, proportional set size=portion of main memory, RAM, occupied by proc)
```
smem -c "name user pss rss"
```

DSTAT

show Out of Memory oom procs that are high on list to be killed
```
dstat --top-oom  (yum install dstat)
```

check process OOM score
```
cat /proc/PID/oom_adj  (-10 is lower priority to get killed than 10)
```

Journalctl

tail a log for a process
```
journalctl -u httpd -f
```

show last 100 lines for a process
```
journalctl -u httpd --no-pager -n100
```

tail a process log
```
journalctl -f -u <process-name>
```

see journal disk usage
```
journalctl --disk-usage
```

clear journal log space anything older than 5 days
```
journalctl --vacuum-time=5d
```

keep only last 500mb
```
journalctl --vacuum-size=500M
```

see previous Kernel boot messages
```
journalctl --list-boots (higher numbers are older boots)
-3 89bb7913b7f84948a1dc4e05baa5c606 Tue 2023-02-14 12:58:01 CST—Tue 2023-02-14
15:16:58 CST
-2 e3605a8c134b4c1a86e4576365dddc0a Tue 2023-02-14 15:19:16 CST—Tue 2023-02-14
15:28:37 CST
-1 9d174c4a278241db85df5e38b9d17b19 Tue 2023-02-14 15:30:55 CST—Tue 2023-02-14
15:42:51 CST
```

show bootlog from boot # 3
```
journalctl -b-3
```

show only journal logs after a certain date/time
```
journalctl -S "2020-91-12 07:00:00"
```

DMIDecode

show bios
```
dmidecode -t bios
```

system info
```
dmidecode -t system
```

chassis
```
dmidecode -t chassis
```

memory, processor, slot
```
dmidecode -t memory
dmidecode -t processor
dmidecode -t slot
```

serial #
```
dmidecode -s system-serial-number
```

Show all current users logged in
```
who
w
```

Send msg to all logged-in users
```
wall -n "hello"
```

Show all loaded modules
```
lsmod
```

insert, remove mod
```
insmod fat
rmmod fat
```

Show current runlevel
```
runlevel
```

show IRQ drivers being used
```
cat /proc/interrupts
```

show DMA channels being used (comms between I/O ports)
```
cat /proc/dma
```

show I/O ports being used
```
cat /proc/ioports
```

Stress Testing
```
yum install stress-ng
```

run stress on 2 CPUs
```
stress-ng --cpu 2 --timeout 10s --metrics-brief
```

force Out of memory kill
```
stress-ng --vm 5 --vm-bytes 95% --vm-method all --verify -t 1m -v
```

Stress I/O load, run 5 workers that will continually R/W to temp file
```
stress-ng -d 5
```

Run application with memory limit
systemd-run --user -p MemoryLimit=3G google-chrome

Kill frozen process
```
Alt + PrintScreen + f
```

Find procs using most SWAP space
```
find /proc -maxdepth 2 -path "/proc/[0-9]*/status" -readable -exec awk -v FS=":"
'{process[$1]=$2;sub(/^[ \t]+/,"",process[$1]);} END {if(process["VmSwap"] &&
process["VmSwap"] != "0 kB") printf "%10s %-30s
%20s\n",process["Pid"],process["Name"],process["VmSwap"]}' '{}' \; | awk '{print $(NF-
1),$0}' | sort -hr | head | cut -d " " -f2-
```

Get top 25 Memory hogs
```
ps -eo pid,user,ni,rss,vsz,cputime,lstart,etimes,time,%cpu,%mem,args --sort=-rss |
head -n 25
```

DMESG

check kernel actions during bootup
```
dmesg -T
```

Top, Htop

show by memory
```
top -o %MEM  (hit 'c' to show full command)
```

Network / Ifaces configuration

IP command

show all interfaces
```
ip a
```

show specific interface
```
ip addr show dev em1
```

assign address to interface
```
ip addr add 192.168.5.2 dev em1
```

show only active interfaces
```
ip link ls up
```

bring up an interface
```
ip link set dev em1 up
```

disable an interface
```
ip link set dev em1 down
```

rename inteface w/o network restart
```
ip link set dev em1 down
```

ip link set em1 name eth1
```
ip link set eth1 up
```

delete interface
```
ip link delete em4
```

bring up an interface
```
ip link set em1 up
```

change MTU on interface
```
ip link set em1 mtu 9000
```

see all routes
```
ip route  or  route -n
```

get route for an IP
```
ip route get 192.168.1.2
```

delete route
```
ip route del 192.168.1.2
```

add a new route via gateway
```
ip route add 192.168.1.2 via 192.168.1.1 dev em1
```

add default route
```
ip route add default ia 192.168.1.1 dev em1
```

update route with congestion window and receive window sizing
```
ip route change default via 192.168.38.1 dev em1 proto static initcwnd 10 initrwnd
```

show all tunnels
```
ip tunnel
```

NetworkManager

show all devices
```
nmcli device
```

start a device
```
nmcli device connect em1
```

get UUID
```
nmcli connection show
```

generate UID
```
uuidgen eth0
```

add new connection
```
nmcli c add connection.interface-name enp1s0 type ethernet
```

stop managing iface with NetworkManager
```
nmcli d set eth1 managed no
```

delete interface
```
nmcli dev disconnect eth1
```

show device information (Mac address, etc)
```
nmcli d show
nmcli d show eth0
```

connect, disconnect, status for device
```
nmcli d connect em1
```

bring up / down iface
```
nmcli c up eth1
nmcli c down eth1
```

start/stop network stack
```
nmcli networking off (on)
nmcli n off
nmcli n on (bounce network)
```

reload connections
```
nmcli c reload
```

migrate legacy ifcfg connections from /etc/sysconfig/network-scripts
```
nmcli c migrate em1
```

VLAN config

add new vlan
```
nmcli c add type vlan con-name bond0.252 ifname bond0.252 vlan.parent bond0 vlan.id
252
```

add a nested vlan
```
nmcli c add type vlan con-name bond0.252:5 ifname bond0.252.5 dev bond0.252 id 5
```

Network Teaming/Bonding

create network team from em1, em2 ifaces

can create teaming modes based on following:
```
1) broadcast - transmits data over all ports
2) roundrobin - transmits data over all ports in turn
3) activebackup - transmits data over one port while the other are kept as backup
4) loadbalance - transmits data overa ll ports with active Tx load balancing
5) random - random selected port
6) lacp - 802.3ad link aggreggation protocol
```

1. create team iface
   ```
   nmcli c a type team con-name bond0 ifname bond0 team.runner loadbalance
   ```
2. add ifaces to team
   ```
   nmcli c a type team-slave con-name em1 ifname em1 master bond0
   nmcli c a type team-slave con-name em2 ifname em2 master bond0
   ```
3. configure bond0 IP details
   ```
   nmcli c m bond0 ipv4.addresses 192.168.40.20/24
   nmcli c m bond0 ipv4.gateway 192.168.40.1
   nmcli c m bond0 ipv4.dns 8.8.8.8
   nmcli c m bond0 ipv4.method manual
   ```
   all configs are saved into /etc/NetworkManager/system-connections
4. restart NM
   ```
   systemctl restart NetworkManager
   ```
5.

Network Diagnostics

Tracepath # of hops for HTTP request (better than traceroute)
```
tracepath 123.123.21.2
tracepath nycweb1
```

check link speed of iface
```
ethtool em1 (speed: x)
```

check TCP statistics
```
netstat -s -t
```

check congestion and other info
```
netstat -s
```

show drop packet statistics for iface
```
ip -s link show em1
```

query statistics for iface
```
column -t /proc/net/dev
```

Check if port 120 is open and listening
```
netstat -an | grep 120
```

SS (like netstat)

check user, PID listening on port 8080
```
ss -ap4 | grep 8080
```

show all TCP connections
```
ss -t
```

show all Listening TCP conns
```
ss -lt
```

show all UDP connections (for Listening, add -lu)
```
ss -u
```

display PIDs of sockets
```
ss -p
```

display PIDs of sockets
```
ss -p
```

filter by port number
```
ss -at '( dport = :22 or sport = :22 )'
```

show conns from specific source or dest address
```
ss src <IP address>
ss dst <IP address>
```

TCP Dump

show all interfaces tcpdump can listen on
```
tcpdump -D
```

listen on specific interface
```
tcpdump -i eth0
```

listen on all ifaces
```
tcpdump -i any
```

listen on specific port or portrange
```
tcpdump portrange 3334-3380
tcpdump -i any port 12345
```

listen on multiple ports
```
tcpdump -i any port '(80 or 443)'
```

search for specific src IP and port over an iface
```
tcpdump port 1234 and src 1.1.1.1 -i em1
```

search for specific subnet
```
tcpdump port 1234 and net 1.2.3.4/24
```

ord packet capture into a .cap file
```
pdump -w capture.cap
```

d contents of a .cap file
```
pdump -r capture.cap
```

play only IP address and ports instead of hostnames
```
pdump -n
```

play only where destination IP is 192.168.5.1 (for source use -n src)
```
pdump -n dst host 192.168.5.1
```

capture TCP packets where port is between 1 and 1023

Check Traceroute and Ping at same time, live stream
```
mtr www.google.com
```

Check Port communication

find process thats holding a certain port #
```
netstat -tulpn | grep 5000
```

Netcat

Chat client
```
On Server - start NC session
hostA: nc -l 9933

on Client, connect to NC session
hostB: nc hostA 9933

can type messages between servers like chat client
```

netcat from specific interface
```
nc  hostname  22  -s  192.168.30.23  -v
```

start a Netcat Bash session (ghetto SSH)
```
serverA> nc -l 5000 -e /bin/bash
serverB> nc serverA 5000
```

Netcat Ghetto web server
```
while true ; do nc -l -p 1500 -c 'echo -e "HTTP/1.1 200 OK\n\n $(date)"' ; done
```

Scan a range of IPs for an open port,
```
for i in {1..25};do nc -zv 208.224.251.$i 8003 -w 2 ;done
```

Scan an IP for open ports (Ghetto Nmap)
```
nc myhost 1-100 -zv
(will scan ports 1-100 and report if open or not)
```

Spin up a webserver with custom port, check that you can connect to port
```
python2 serverA> python -m SimpleHTTPServer 8331
python3 serverA> python3 -m http.server 8331
serverB> nc serverA 8331
```

connect on a UDP port
```
nc -u <hostname> <port> -vv
```

transfer files between 2 hosts
```
hostA> netcat -l 4444 > /tmp/file1
hostB> echo "cats suck dogs rule" > myfile
hostB> nc hostA 4444 < myfile
hostA> cat /tmp/file1
cats suck dogs rule
```

proxy a port via another host (similar to Socat and Redir)

A needs to connect to C:8333, but doesnt have direct access,
A will use B as a hop to C:8333
```
hostB> nc -k -l 0.0.0.0 8333 --sh-exec "nc hostC 8333"
hostA> nc hostB 8333 -v (will conn  A > B:8333 > C:8333)
```

NPING (part of nmap pkg)
```
send TCP packets over port 22, 80 and 443, send 500 packets at rate of
60 packets / sec with sleeptime of 3 seconds between attempts

(for UDP use --udp [hostname/IP]
nping --tcp nycweb01 -p 80,443,22 -c 500 -rate 60 --delay 3

send ARP request to all hosts on subnet
nping -arp 192.168.30.0/24

send ICMP echo
nping nycweb01 -icmp -icmp-type echo

send packets to ports 20-35
nping -tcp nycweb01 -p20-35

send UDP packets
nping -udp -c 2 -p 23000 <target>
```

To spin up webserver on specific network interface,
```
python -c 'import BaseHTTPServer as bhs, SimpleHTTPServer as shs;
```

```
tcpdump -n tcp dst portrange 1-1023
```

capture packets where destination host is 192.168.5.1 and port is 5049

```
tcpdump -n "dst host 192.168.5.1 and dst port 5049"
```

print packets in ASCII or HEX

```
tcpdump -i any -A (ascii)  or -X (hex)
```

run in background and record to file

```
nohup tcpdump -i any port 27025 -w myfile.cap &
```

rotate pcap file similar to logrotate, this will create 10 pcap files of 200MB each

```
tcpdump -n -W 10 -C 200 -w /tmp/file.pcap
```

check multicast data

```
tcpdump -i p1p2 -s0 -vvv host 233.143.214.1
```

Flags

```
[.] - ACK (Acknowledgment)
[S] - SYN (Start Connection)
[P] - PSH (Push Data)
[F] - FIN (Finish Connection)
[R] - RST (Reset Connection)
[S.] - SYN-ACK (SynAcK Packet)
```

Wireshark

show only bad packets

```
add to filter: tcp.analysis.flags
```

```
show only Resets
tcp.flags.reset == 1
```

show packets between dates

```
(frame.time >= "July 27, 2020 08:40:00" && frame.time <= "July 30, 2020 08:42:42")
```

```
show only problematic packets
_ws.expert.severity == error
```

```
show only UDP / TCP/ RTCP
type into filter: udp (or tcp, rtcp)
```

Check MAC address mapping to IP

```
arp
```

clear ARP cache

```
ip -s -s neigh flush all
```

IP Routing Table

```
route -n
```

add a new route,

```
ip route add 118.100.1.173 via 192.168.38.17 dev p1p2 metric 200
```

add new route permanently

```
vim /etc/sysconfig/network-scripts/route-p1p2
201.224.250.40 via 192.168.38.33 metric 200
```

delete a route

```
ip route del 118.100.1.173
```

modify existing route

```
ip route del 40.2.2.0/24 via 30.1.2.2
ip route add 40.2.2.0/24 via 30.1.2.2 metric 1234
# kill all connections on port 21
tcpkill -i eth0 port 21
```

add TCP permissions to TCP analyzer tools so non-root users can create sockets and access network interfaces

```
setcap cap_net_raw,cap_net_admin=eip /usr/bin/tcpreplay
setcap cap_net_raw,cap_net_admin=eip /usr/bin/tcpdump
```

IPERF

check bandwidth usage

```
yum install iperf3
```

on server:

```
iperf3 -s
```

on client:

```
iperf3 -c <IP of server> -p 5001 <port> -P 20 <# of parallel TCP conns> -t 20 <run for x seconds>

Client connecting to 208.224.251.3, TCP port 5001
TCP window size: 90.0 KByte (default)
------------------------------------------------------------
[  3] local 172.31.23.96 port 48908 connected with 208.224.251.3 port 5001
write failed: Connection reset by peer
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 0.0 sec   130 KBytes  68.9 Mbits/sec
```

test bw using 10 parallel sessions, each session sending maximum of 2mb

```
iperf3 -P 10 -b 2M -c <hostname>
```

use UDP iperf, will flood the connection with datagrams, without session limits of TCP, more accurate bandwidth read than TCP

```
iperf3 -u -b 2M -c <hostname>
```

show iperf3 details (congestion, window sizes, etc)

```
iperf3 -c nycweb1 -t 15 --debug
```

Nuttcp - advanced iperf

check bandwidth usage on a host directly

```
iftop -PN
```

SAR

show live statistics of traffic over all interfaces
sar -n DEV 1 2

```
DEV = network iface info
EDEV = network errors
NFS = active NFS clients
NFSD = NFS server info
SOCK = socket info
ALL = all above
```

Parameter Description:

```
IFACE: LAN interface
rxpck/s: packets received per second
txpck/s: packets sent every second
rxbyt/s: number of bytes received per second
txbyt/s: number of bytes sent per second
rxcmp/s: compressed packets received per second
txcmp/s: compressed packets sent every second
rxmcst/s: multicast packets received per second
rxerr/s: bad packets received per second
txerr/s: bad packets sent every second
coll/s: conflicts per second
rxdrop/s: the number of received packets dropped per second because the
buffer is full
txdrop/s: the number of sent packets dropped per second because the
buffer is full
txcarr/s: number of carrier errors per second when sending packets
rxfram/s: the number of frame alignment errors received per second
rxfifo/s: the number of FIFO over speed errors per second of received
packets
txfifo/s: the number of FIFO over speed errors per second in packets
sent
```

show statistics for all ifaces

```
cat /proc/net/dev
```

```
bhs.HTTPServer(("192.168.200.99", 8331),
shs.SimpleHTTPRequestHandler).serve_forever()'
```

check ports using nmap

```
nmap localhost
PORT    STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
89/tcp   open  su-mit-tg
```

NMAP
check subnet for open ports

```
nmap -sP -PS22,3389 192.168.30.1/24
```

DNS

Check DNS routing

```
host github.com
github.com has address 192.30.253.113
github.com has address 192.30.253.112
github.com mail is handled by 10 ALT3.ASPMX.L.GOOGLE.com.
```

Dig into DNS query

```
dig www.domain.com
```

check all DNS name servers

```
cat /etc/resolv.conf
```

get your public IP from google

```
dig +short myip.opendns.com @resolver1.opendns.com
124.245.66.135
```

or this

```
curl -4 icanhazip.com
```

Check all open network connections

```
lsof -i
```

Check which procs are holding up deleted files

```
lsof  +L1
```

check output of df vs du

```
df shows total usage including file descriptors, du shows actual usage
kill any procs holding up "deleted" file descriptors, will show reduction of used space
```

Get true Timezone

```
curl https://ipapi.co/timezone
```

Multicast

see what MC groups are present

```
ip maddr
```

Network Utilities

hping3 - like ping but can connect to ports and use TCP
iftop - iface network activity top
ss - better version of netstat (ss -ap4)
iptraf - interface and network cmd line gui tool (very good)

Kill a TCP session w/o killing process (will only kill **new** connections, not Established)

```
yum install dsniff
```

```
tcpkill -i eth0 port 28394
```

kill Established TCP connection via port (doesnt kill parent process)

lsof -np <PID of Parent> | grep <IP of remote host> (get the FD number, 4th column)

```
gdb -p <PID of parent> --batch  -ex 'call shutdown(FD #)'
```

ie, need to kill this specific TCP session but not kill MyApp (this app has other TCP established connections)

```
tcp        0      0 192.168.38.21:25959     108.124.250.173:50443 ESTABLISHED 221955/MyAPP

lsof -np 221955 | grep 108.124.250.173
risk_gate 221955 qbsim   17u     IPv4 2857516568         0t0        TCP 192.168.38.21:25959->108.124.250.173:51212 (ESTABLISHED)
```

FD id = 17u (update), now free up this file descriptor

```
gdb -p 211955 --batch -ex 'call shutdown(17u, 2)'
```

Close File descriptor without killing the process (ie proc is up but file is deleted)
check for deleted files

```
lsof +L | grep deleted | grep <filename> (get PID of this proc)
```

get ID of FD (4th column of lsof output), ie 43w

detach file descriptor from proc

```
gdb -p <PID> --batch -ex 'p close(43)'
```

IPTables

show all rules

iptables -L -n -v

show all FORWARD rules

iptables -L FORWARD --line-numbers

delete a rule

iptables -D FORWARD <line number>

check existing NAT rules

iptables -t nat -v -L POSTROUTING --line-number

iptables -t nat -v -L PREROUTING --line-number

forward any request from ServerA port 80 to ServerB port 80 on server A

(make sure to add **sysctl -w net.ipv4.ip_forward=1**)

iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination <IP of serverB>:80

iptables -t nat -A POSTROUTING -p tcp -j MASQUERADE

change outgoing packets IP header

iptables -t nat -A POSTROUTING -d <destination IP> --dport <PORT> -j SNAT --to-source <IP you want to change to>

forward an OUTGOING packet for a specific port (going from host A), to another host (host B)

host A>

iptables -t nat -A OUTPUT -p tcp --dport 8331 -j DNAT --to-destination 10.182.26.8:8331

allow all connections from an IP

iptables -A INPUT -s 59.50.131.179 -j ACCEPT

forward a packet going to a specific hostname and port to another hostname:port

iptables -t nat -A PREROUTING -p tcp -d 18.224.251.4 --dport 22 -j DNAT --to-destination 192.168.10.22:22

completely flush all chains, rules, filter, raw, mangle, etc

allow all incoming connections to avoid being locked out

iptables -P INPUT ACCEPT

flush custom chains, nat, raw, security, mangle, filter rules

iptables -X

iptables -t nat -F

iptables -t raw -F

iptables -t security -F

iptables -t mangle -F

iptables -F

save all IPTABLES rules permanently

iptables-save > /etc/sysconfig/iptables

restore from file

iptables-restore < /tmp/backup.iptables

add Debug log to prerouting rule #3 (tail syslog)

iptables -t nat -I PREROUTING 3 -j LOG

allow SSH port 22 only from address 190.120.30.3, block all others

iptables -I INPUT -p tcp '!' -s 190.120.30.3 --dport 22 -j REJECT

allow SSH port for specific address

iptables -A INPUT -p tcp -s 190.120.30.3 --dport 22 -j ACCEPT

block port

iptables -A OUTPUT -p tcp --dport 2500 -j DROP

allow a port

iptables -A INPUT -p tcp --dport 2500 -j ACCEPT

allow an IP address

iptables -A INPUT -p tcp -s 192.168.3.5 -j ACCEPT

iptables -A OUTPUT -p tcp -d 192.168.3.5 -j ACCEPT

block an IP address

iptables -A INPUT -s 192.130.2.4 -j DROP

block range of IPs

iptables -A INPUT -s 192.168.2.0/24 -j DROP

allow range of ports (1200 and 5000-6000)

iptables -A INPUT -p tcp --match multiport --dports 1200,5000:6000 -m conntrack -j ACCEPT

redirect port to another port on same host

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j REDIRECT --to-port 2525

create a custom CHAIN

iptables -N My-Custom-Rules

Packages / Libs / Modules

show installed software

Debian distro

dpkg -l

apt-cache search [pkg name]

Fedora distro

yum list installed

rpm -qa | grep [pkg name]

yum search [pkg name]

RPM install package

rpm -i pkg.rpm

rpm -i mypkg.rpm --force (force install)

rpm -i mypkg.rpm --nodeps (ignore dependencies)

what RPM does a file belong to?

rpm -qf /usr/bin/mysqlaccess

show files inside installed RPM package

rpm -ql package-name

show files inside local uninstalled RPM package

rpm -qpl local-file.rpm

Show libraries for a program

ldd /bin/ls

refresh YUM cache

yum clean expire-cache|

yum clean all

show dependency for a package

yum -q deplist $pkg

see install/upgrade history

yum history

get info on specific yum transaction

yum history info <# of transaction>

rollback yum patch

yum history undo <# of transaction>

rebuild cached library list or add new libs

vi /etc/ld.so.conf

ldconfig

Install package including its dependencies, example 'mysql'

yum deplist mysql | awk '/provider:/ {print $2}' | sort -u | xargs yum -y install

show installed packages by disk space usage (Centos)

rpm -qa --queryformat '%10{size} - %-25{name} \t %{version}\n' | sort -n

remove old kernels, keep only current and 1 older

```
package-cleanup --oldkernels --count=2
```

Modules

show custom modules

```
dkms status (yum install dkms)
```

show loaded modules

```
lsmod | grep <modname>
```

load module (insert)

```
insmod /lib/modules/<kernel version>/kernel/drivers/<etc>
```

unload module

```
rmmod /lib/modules/<kernel version>/kernel/drivers/etc
```

same but using modprobe w/o needing path to modules

```
insert/load:   modprobe <modname>
remove:  modprobe -r <modname>
```

Process / Init / CPU

get uptime of a process
```
ps -p $$ -o etime=
```
where $$ is PID, result is in format dd-hh:mm:ss

find PID of a process (add to .bashrc)
```
function pid() { ps -fU $USER | grep $1 | grep -v "grep" | grep -v "ps -fU"  ;}
```

Run process in background (use & to push to background)
```
./run_script.sh &
```

Get current PID
```
$$
```

kill all processes by name (with confirmation)
```
pkill -f $name
```

kill process by Port
```
fuser -k 5100/tcp
```

kill process by owner name
```
killall -u username
```

find process by name, kill all
```
ps -ef | grep "vault server" | grep -v grep | awk '{print $2}' | xargs kill -9
```

show all procs and their children
```
pstree -ap
```

show 4 way scrollable process tree
```
ps awwfux | less -S
```

show all processes and children
```
ps -ef --forest
```

show # of processes per user
```
ps hax -o user | sort | uniq -c | sort -r
```

kill a process running on Port 8331
```
kill -9 $(lsof -i :8331 | awk '{l=$2} END {print l}')
```

get amount of open file descriptors by user
```
lsof -u <username> | wc -l
```

get home directory of a process ID
```
pwdx <PID>
```

NTP

check offset of time between 2 servers,
[23:38 root@web1:~]# ntpdate -q web2
server 10.112.42.8, stratum 2, offset 0.005212, delay 0.02580
13 Aug 23:39:01 ntpdate[17325]: adjust time server 10.182.48.8 offset 0.005212 sec

offset of less than 5/1000s of a second

check offset against timeserver
ntpq -p

run in debug
ntpdate -dv <name of timeserver>

Synchronize time w another host over SSH (server2 has correct date)
date --set="$(ssh user@server2 date)"

Hide processes and PIDs for non-root users
edit /etc/fstab
proc /proc proc defaults,hidepid=2 0 0

remount
mount -o remount,rw,hidepid=2 /proc

to add an exception for a group/user (let this group see other PIDs), add 'gid' & remount
proc /proc proc defaults,hidepid=2,gid=joe 0 0

Isolate CPUs for specific processeses

Isolate CPUs for specific processeses
```
grubby --default-kernel
/boot/vmlinuz-3.10.0-862.14.4.el7.x86_64

grubby --info=/boot/vmlinuz-3.10.0-862.14.4.el7.x86_64
args="ro no_timer_check console=tty0 console=ttyS0,115200n8
net.ifnames=0 biosdevname=0 elevator=noop crashkernel=auto
LANG=en_US.UTF-8"

## get current islated cores
cat /sys/devices/system/cpu/isolated

## add cpu isolation
grubby --update-kernel=/boot/vmlinuz-3.10.0-862.14.4.el7.x86_64 --
args=isolcpus=2,3

reboot host to pickup changes

## remove isolation
grubby --remove-args="isolcpus=2,3" --update-kernel=<kernel name>
```

Run command with a process "niceness" or priority (-20 highest priority, 19 lowest)
```
nice -18 cat /etc/hosts
```

Check new incoming connections on port, live
```
ss -nap | grep 4433
```

Change a running program's priority (change to priority 7, PID 168390 for all processes running by users 'root' and 'joe'
```
renice 10 168390 -u root joe
```

Systemctl

show all enabled services
```
systemctl list-unit-files | grep enabled
```

show all running services
```
systemctl list-units --type=service --state=running
```

start / stop / status / refresh / reload / enable / disable / show
```
systemctl start httpd.service
```

analyze bad startup script
```
systemd-analyze verify monit.service
```

refresh sysctl
```
systemctl daemon-reload
```

I/O
monitor high disk IO
```
* * * * * root /usr/sbin/iotop -botqqqk --iter=60 | grep -P "\d\d\.\d\d K/s" >>
/var/log/iotop
```

Limit CPU usage for a process #2240 to 50% of CPU and also its child procs
```
cpulimit -pid 2240 -l 50 -i
```

Taskset and NUMACTL
start a process on only 1st CPU core
```
taskset -c 0 /bin/nginx
```

for multiple CPU affinity
```
nohup taskset -c 0,1,2,5 /bin/program
```

get range of CPUs on which process can run on (affinity)
```
taskset -cp <PID>
```

get CPU on which a PID is running on
```
ps -mo pid,tid,fname,user,psr -p <PID>
```

pin processes to specific CPUs that are isolated (by default, numa does not allow pin to isolated CPUs, must use ALL option)
```
cat /proc/cmdline
isolcpus=2,3
nohup numactl --all -C 2,3 /bin/myprogram
```

find User and Parent PID of a zombie process thats holding up a port
#1 get the iNODE
```
root@min1# netstat -ltpnae | awk 'NR==2 || /:18100/'
Proto Recv-Q Send-Q Local Address           Foreign Address     State       User  Inode  PID/Program name
tcp        1      0 0.0.0.0:18100           0.0.0.0:*           LISTEN      1000  24444060  -
tcp        1      0 192.168.37.5:18100      208.224.250.11:1046 CLOSE_WAIT  0     0         -
```

#2 search by iNODE
```
root@min1# lsof | awk 'NR==1 || /24444060/'
COMMAND     PID  TID        USER   FD      TYPE             DEVICE   SIZE/OFF       NODE NAME
trading_engine 138517 138529      joe   50u  IPv4         24444060       0t0          TCP *:18100 (LISTEN)
```

run program in background, no output
nohup programName 2>&1 &

clear abrtd messages
if getting abrtd-cli timed out
check /var/spool/abrt or /usr/local/spool/abrt
remove old abrt files, restart abrtd service

Generate Core Dump file into specific location
sysctl -w kernel.core_pattern = /mnt/core.%e.%p.%h.%t

add core limits, set limit of core size to 5mb - 4096 bytes per block
5(MB) * 1024 * 1024 / 4096
1mb = 256 blk
1gb = 262,144 blk

vim /etc/security/limits.conf
joe soft   core   1280 (4096 bytes per block, 5MB core = 1280 blocks)
joe hard   core   1280

check core limits (start new session as Joe)
joe> ulimit -a
create new background proc
joe> nohup python -m SimpleHTTPServer &
kill proc to generate core file
joe> kill -s SIGTRAP $(pgrep python)

User / Group / Sudo

USERS

create new user
```
adduser eric
```

add user to Group
```
usermod -aG mygroup eric
```

remove user from Group
```
gpasswd -d <user> <group>
```

add user to multiple groups
```
usermod -aG group1,group2,group3 eric
```

change UID for user
```
usermod -u 2550 eric
```

change GID for user
```
groupmod -g 2550 eric
```

lock a user account
```
passwd -l eric
```

unlock user account
```
passwd -uf eric
```

delete a user's password
```
passwd --delete eric
```

change user's shell
```
usermod --shell /bin/bash eric
```

remove expired password requirements for user
```
chage -m 0 -M 99999 -I -1 -E -1 jsmith
```

remove expired password for all users
```
for user in $(cut -d: -f1 /etc/passwd); do sudo chage -M 99999 -I -1 -m 0 -E -1 $user; done
```

delete /home folders of users that are deleted on system
(check for user GroupID, if 'UKNOWN', then delete that home folder)
```
for i in $(ls /home) ; do stat --format='%G' $i |  xargs echo $i | grep UNKNOWN |
awk {'print "/home/"$1'} | xargs rm -rf;done
```

remove user from group
```
gpasswd -d joe wheel
```

create a nologin user (no home dir)
```
useradd -r joe
or
adduser -r -s /bin/nologin jsmith
```

create user Joe with custom home dir, custom ID 999, custome group ID 555, add to 2 groups (corp, webadmins)
```
useradd -d /var/home/joe -u 999 -g 555 -G corp,webadmin joe
```

via Perl
```
adduser --home /var/home/joe -u 999 -g 555 -G corp,web joe
```

GROUPS

add new group
```
groupadd mygroup
```

remove group
```
groupdel mygroup
```

modify group ID
```
groupmod -g 999 mygroup
```

change group name
```
groupmod -n newgroup oldgroup
```

show what cores a process is running on
```
for i in $(pgrep <name of process>); do ps -mo pid,tid,fname,user,psr -p $i;done
```

allow user to run command as another user (joe can run htop command as fred)
```
vi /etc/sudoers.d/htop
joe ALL=(fred) NOPASSWD: /bin/htop

joe>  sudo -u fred /bin/htop
```

or another way
```
joe> su -c 'bash myprogram.sh' fred
```

Check sudoers syntax
visudo -cf /etc/sudoers.d/mysudo

File / Dir

Rsync
sync files from one Dir1 to Dir2
rsync -azP dir1/ dir2 ## -z flag is compression

-azP flag is used to compress file (z), and P for partial, it will only rsync deltas instead of starting all over from scratch

RSYNC file to a remote system's /tmp dir
rsync -azP file1 root@remotesystem:/tmp

rsync and exclude logs, png
rsync -azP --exclude={*.log,*.png} server1:/tmp/dir /tmp

Pull file from a remote system to a local /tmp dir
rsync -azP root@remotesystem:/opt/file1 /tmp

If Rsync not found, use path

```
--rsync-path=/usr/bin/rsync
```

Rsync using a hop server (A > B > C)
assuming you can ssh joe@A > joe@B
and can ssh from joe@B > joe@C

rsync -azP -e "ssh -A joe@B ssh" file1 joe@C:/tmp

will rsync local file1 via B, into C

if Rsync versions dont match up, can also do this, (rsyncs file on C to localhost via B)
rsync -azP -e 'ssh -o "ProxyCommand ssh -A joe@B nc %h %p"' joe@C:/tmp/xferfile .

rsync - set mod and ownership on incoming files/dirs,

```
hostA> ls -la /home/joe
drwxrwsr-x. 3 joe   groupA  21 Sep 16  2018 tmp/

hostB> rsync -azP --chmod 644 --chown=mary:accounting hostA:/home/joe/tmp .
hostB> ls -la
drw-r--r--  mary  accounting  /tmp
```

Rsync using specific SSH keys

Sort file
sort -d filename ## alphabetically
sort -r filename ## reverse order
sort -n filename ## numeric sort
sort -M filename ## sort by month date

SSHFS
```
sudo sshfs -o allow_other,defer_permissions root@xxx.xxx.xxx.xxx:/
/mnt/droplet
```

copy all files to destination except for whatever is in .gitignore
```
cp -r !($(cat .gitignore)) /tmp/dest
```

mount NFS share
yum install nfs-utils nfs-utils-lib
service nfs start
mount -t nfs <serverIP>:/path/of/mount /mnt/point

remove first 500 lines of a file, in place (shrink a log file)
```
sed -i -e 1,500d file.log
```

reduce log file to 200b
```
truncate -s 200 file.log
```

User & Group Permissions

give 'sysadmin' Group 777 permission to a dir /opt/test
```
chmod g+rwx /opt/test
```

change group ownership for symlink (recurse down)
```
chgrp -Rh mygroup /home/user/dir
```

add execute bit for group on all folders
```
find . type -d | xargs chmod g+x
```

change group ownership of a dir
```
chgrp sysadmins /opt/test
```

Get ACL on a directory
```
getfacl /opt/test
```

give Sysadmins group 777 to /opt/test
```
setfacl -m group:sysadmins:rwx /opt/test
```

to set recursively down,
```
setfacl -Rm u:joe:rwx /home/mary
```

remove ACL
```
setfacl -x user:antony /opt/test
```

give r/w access to /home/user1 and preserve SSH security
```
chmod 750 /home/user1
setfacl -m user:user2:rw /home/user1
```

remove all ACLs from file or dir
```
setfacl -b /home/user1
```

set a default ACL for a directory (all new files or dirs created in this directory will inherit ACL permissions)
```
setfacl -d -m u::rwx,g::rwx,o::r- /opt/testdir
setfacl -Rdm u:joe:rwx /opt/somedir
```

Backup and restore all permissions
make a backup of all permissions in a directory,
```
getfacl -R /home/user > /tmp/permissions_backup
```

restore all perms recursively
```
setfacl --restore=/tmp/permissions_backup
```

ensure all files and dirs created by user, inherit the Group permission of parent directory (SUID bit) - this example gives Joe rwx, gives group "employees" only Read (directories get set with X in order for group members to 'ls' to them), all others have no access to this folder or subfolders

```
1. chgrp -Rh employees /home/joe
2. setfacl -d -Rm u::rwX,g::rX,o::-  /home/joe
3. chmod -R g+s /home/joe  (set S bit to inherit parent permissions for all new
subfolders)
4. chmod -R g-w /home/joe  (removes write perms for group inside joe's home folder)
```

add timestamp to a tail of log file
```
tail -f /var/log/messages | while read ; do echo "$(date +%T.%N) $REPLY" ; done
```

copy all ssh keys for every user from 1 host to another
```
host1> for i in $(ls /home);do rsync -azP /home/$i/.ssh/id_rsa*
host2:/home/$i/.ssh/ ;done
```

get directory permissions of a user's directory in numeric form
```
stat -c '%a' /home/user     >> 700
```

get owner of directory
```
stat -c '%u' /home/user   >> user  (use %g for group)
```

Logrotate

place all logrotate confs in /etc/logrotate.d
```
/var/log/httpd/*log {
    rotate 3  # how many rotated files to keep left over
    size 10MB  # rotate if log exceeds this
    daily  # rotate on daily basis unless size max criteria is met first
    maxage 20 # delete old rotate files over 20 days
    compress  # gzip compress rotated files
    missingok
    notifempty
    sharedscripts
    postrotate
        /sbin/service httpd graceful 1>/dev/null 2>&1 || true
    endscript
}
```

Searching

find all files larger than 100M
find /home -xdev -type f -size +100M | xargs du -sh | sort -hr

Find 10 largest files
find . -type f -print0 | xargs -0 du | sort -n | tail -10 | cut -f2 | xargs -I{} du -sh {}

another way
find /home -type f -exec du -Sh {} + | sort -rh | head -n 5

find all files created in last 120 minutes
find / -cmin 120

Find 10 largest dirs
find . -type d -print0 | xargs -0 du | sort -n | tail -10 | cut -f2 | xargs -I{} du -sh {}

find 25 largest files in current dir and its subdirs
find . -type f -exec ls -al {} \; | sort -nr -k5 | head -n 25

find duplicate files, (based on MD5 hash)
find -type f -exec md5sum '{}' ';' | sort | uniq --all-repeated=separate -w 33

find specific user's files
find . -user <username> -print

find total size of files matching a patter
```
du -ch *.jpg | grep total
```

recursively remove all empty subdirs
find . -depth -type d -empty -exec rmdir {} \;

find all hard links to a file
find /path/to/dir -xdev -samefile <name of file>

find the latest modified files (recursively)
find . -type f -exec stat --format '%Y :%y %n' "{}" \; | sort -nr | cut -d: -f2- | head

find files modified or created in last 2 days
find /dir -newermt "2 days ago" -ls

Show top 10 largest open files
lsof / | awk '{ if($7 > 1048576) print $7/1048576 "MB" " " $9 " " $1 }' | sort -n -u | tail

show 10 largest files in a directory
du -a /opt/blah | sort -n -r | head -n 10

list by size(-S), human readable(-h), all(-a), reverse date order (-r), list (-l), date (-t)

find files older than 300 days, display them
find /tmp -type f -mtime +300 -print | xargs ls -lha

now delete them
find /tmp -type f -mtime +300 -print | xargs rm

Find and Search
find -name filename ## any file

Find recursively any hidden file
find /dirname -name ".*" -print

show only hidden files and directories
ls -l -d .[!.]?*

Find in specific dir
find /tmp -name myfile

Find file in specific location larger than 20MB
find /tmp -size +20M

Find files larger than 20MB and older than 360 days, delete them
find /tmp -type f -size +20M -mtime +300 -print | xargs rm

get last element
echo /my/dir/name/backups/someFile.tar | awk -F"/" {'print $(NF)'}
someFile.tar

get filename from a base path,
basename /my/dir/name/backups/someFile.tar   // someFile.tar

compare contents of 2 directories
diff <(cd </path/to/dir1> && find | sort) <(cd </path/to/dir2> && find | sort)

Freeze (lock) a directory or file from being modified (ACL, permissions, ownership,etc) - only root can unlock this. NOTE - this also prevents creating new files, this "freezes" the dir completely.
chattr +i <dir name> (locks dir)
chattr -i <dir name> (unlocks)

filesystem shows 100% usage, but actual usage is much less (FS has too many inodes open)
1. check amount of free inodes on mount (ie, /home shows 100% usage)
df -i (check INode column)

2. see which files are from dead procs

lsof +L1 | grep /home | awk '{$7=$7/1048576 " MB"}1'

3. unmount directory

umount /home

4. repair FS (check which block device with lsblk)

xfs_repair /dev/sda1

5. remount (mount -a)

Compression / Cron / Mount / Encryption

mount ISO
```
mount -t iso9660 -o loop /home/tecmint/Fedora-18-i386-DVD.iso /mnt/iso/
```

unmount ISO
```
umount /mnt/iso
```

Compression

compress using bz2
```
tar cvfj  mydir.tar.bz2  /home/mydir
```

untar tar.bz2 file
```
tar -xvf file.tar.bz2
```

uncompress bz2 file
```
bzip2 -dk file.bz2
```

compress a file XZ format (best compression)
```
tar -cvpJf mydir.tar.xz /home/user/mydir
```

compress a folder into XZ format, perserve permissions, dont include parent folders, just include the target folder (folder is located in /mnt/hc/myFolder)
```
tar -cJf myFolder.tar.xz -C /mnt/hc myFolder --preserve-permissions
```

untar a XZ tarball
```
tar -xf myFolder.tar.xz
```

untar XZ to a specific directory, preserve permissions inside tarball
```
tar -xf myFolder.tar.xz -C /home/joe --preserve-permissions
```

uncompress XZ file
```
unxz file.xz
```

tar a file or dir into tar.gz
```
tar zcvf name.tar.gz file1 dir1 dir2
```

untar and unzip
```
tar -xvzf file.tar.gz
```

untar .tgz
```
tar xzvf file.tgz
```

compress using LZMA
```
tar -cavvf file.tar.lzma file
```

uncompress LZMA to a directory
```
tar -xavvf file.tar.lzma -C <dir>
```

see whats inside a tar
```
tar -tvf mydir.tar
```

see inside tar.gz
```
tar -tf file.tar.gz
```

see inside tar.xz
```
tar tvfa file.tar.xz
```

untar single file from tar.gz (for bz2, replace tar.gz with tar.bz2)
```
tar --extract --file=mydir.tar.gz file1
```

untar multiple files using wildcard
```
tar -zxvf mydirs.tar.gz --wildcards '*.php'
```

create a symlink
```
ln -s <path to actual binary> <target location>
ln -s /usr/local/bin/myapp /usr/bin/myapp
```

download entire website down to local level (and convert links to local) Wget Mirror
```
wget -mk www.google.com
```

download a file using curl
```
curl -O -u<USERNAME>:<API_KEY> -X GET
https://api.bintray.com/packages/orgname/repo_name/pkg_name/logs/downloads-03-12-2016.csv.gz
```

Make disk backups/images with dd

create a disk backup of disk /dev/sda, save to backup.gz
```
dd if=/dev/sda conv=sync,noerror bs=128K | gzip -c > /mnt/hc/backup.gz
```

restore image onto disk
```
gunzip -c /mnt/hc/backup.gz | dd of=/dev/sda
```

create backup image of host1, and store img on remote host2,
```
host1> dd if=/dev/sda conv=sync,noerror bs=128K status=progress | gzip -c | ssh
user@host2 'dd of=/opt/backup.gz'
```

restore host1 by reading backup from host2
```
host1> ssh user@host2 'dd if=/opt/backup.gz status=progress' | gunzip -c | dd
of=/dev/sda
```

Audit

show status of audit system
```
auditctl -s
```

show all audit rules
```
auditctl -l
```

clear all rules
```
auditctl -D
```

monitor file for any changes
```
auditctl -w /etc/filename -p wa -k myfile_changes
```

see any changes done to file
```
ausearch -k myfile_changes
```

save audit rules permanently
```
add to /etc/audit/rules.d/audit.rules
-w /etc/filename -p wa -k myfile_changs
```

check user actions by user name, from yesterday to now,
```
ausearch -ua joe -ts yesterday -te now -i
```

```
search by specific command and specific directory that was deleted
ausearch -f /tmp/testdir -x /bin/rm
```

```
search by type of event
ausearch -ua joe -m SYSCALL (or EXECVE)
```

search by time range
```
ausearch -ua joe --start 09/09/2019 '12:04:00' --end 09/12/2019 '12:22:00'
```

search by parsing a specific log file
```
ausearch -ua joe --input /tmp/audit.log
```

search raw text grep by port number
```
ausearch -r | grep -E "a[1-5]=\"9999\""
```
```
>> type=EXECVE msg=audit(1677000183.185:1366): argc=4 a0="nc" a1="titan" a2="9999"
a3="-v"
```

get list of failed login attempts by user and IP where theyre coming from
```
last -f /var/log/btmp
```

FTP / LFTP

ftp a file providing username + password
```
lftp sftp://"user:password@host" -e "put -O path/on/target /tmp/file.txt
```

LFTP using SSH options, run command
```
lftp sftp://$user:$ftpcred@$host:$port -e "set sftp:connect-program 'ssh -
oHostKeyAlgorithms=ssh-rsa'; cd $rem_logdir; put $logname; bye"
```

check JSON formats for multiple files
install jsonlint and check format
```
npm install jsonlint -g
$ for i in $(ls | grep *.json); do jsonlint $i; done
```

Cron

show all crons for a user
```
crontab -l -u <username>
```

edit crons for your user
```
crontab -e
```

execute cron manually
```
run-parts /var/spool/cron
```

Test Email
```
yum install mailx
mail -s "test email" user@company.com < /dev/null
```

Centos Xauthority (graphical gui)
```
yum install -y xorg-x11-server-Xorg xorg-x11-xauth xorg-x11-apps
grep -i X11Forwarding /etc/ssh/sshd_config (should be set to Yes)
```

ssh to box
```
ssh -X name@box
xclock (test)
```

SYSCTL

show all current values
```
sysctl -a
```

write new value
```
sysctl -w vm.swappiness=2
```

load values from file
```
sysctl -p /etc/sysctl.conf
```

monitor a command (run command repeatedly)
```
watch -n 5 free -h (runs free -h every 5 sec)
```

ENCRYPTION

OpenSSL

check cert expiration with OpenSSL
```
cat mycert.crt | openssl x509 -noout -enddate
```

Encrypt a file
```
openssl enc -aes-256-cbc -salt -in myFileUnencrypted.txt -out
myFileEncrypted.txt.enc -k myPASSWORD
```

Unencrypt File
```
openssl enc -aes-256-cbc -in MyFileEncrypted.txt.enc -out myFileUnencrypted.txt
<type in password>
```

7ZIP

install 7zip on centos
```
wget
https://www.mirrorservice.org/sites/dl.fedoraproject.org/pub/epel/7/x86_64/Packag
es/p/p7zip-16.02-10.el7.x86_64.rpm
wget
https://www.mirrorservice.org/sites/dl.fedoraproject.org/pub/epel/7/x86_64/Packag
es/p/p7zip-plugins-16.02-10.el7.x86_64.rpm
sudo rpm -U --quiet p7zip-16.02-10.el7.x86_64.rpm
sudo rpm -U --quiet p7zip-plugins-16.02-10.el7.x86_64.rpm
```

Encrypt a file
```
7za a -tzip -p -mem=AES256 testfile.zip testfile (enter password)
```

Decrypt a file
```
7za e testfile.zip  (enter password)
```

GPG

send an encrypted file to a recipient

1. generate new gpg key pair
```
sender> gpg --gen-key
```

enter information including password and email address, *password=S3nD3R*
fred.sender@sender.com

if process is hanging on entropy, run the following to speed it up,
```
haveged -n 50g -f - | dd of=/dev/null
```

```
if password part is failing (may be due to TTY bug), run like this
script /dev/null
gpg --gen-key
```

2. Export the pub key to the recipient
```
sender> gpg --armor --output mypubkey.gpg --export fred.sender@sender.com
```

email this pub key to recipient

3.The reciever needs to generate their own public key w their own password and email

Import the recipient's pubkey into your GPG chain (have the recipient send you their pub key and provide the email associated to the pub key)
```
sender> gpg --import recipient-pubkey.gpg
```

have the recipient import sender's pub key the same way

4. Encrypt the file (has to be 1 single file, not multiple files or directories)
```
encrypt using senders private key
sender> gpg --output myFile.txt.gpg --encrypt --recipient
joe.recipient@recipient.com myFile.txt
```

this will generate a binary gpg file

5. create a file signature (Checksum verification)
```
sender> shasum -a 256 myFile.txt | awk '{print $1}' > myFile.txt.sha256sum
sender> gpg --output myFile.txt.sha256sum.sig --sign myFile.txt.sha256sum  (enter
password: S3nD3R)
(if shasum isnt installed, install with yum install -y perl-Digest-SHA)
```

email both binary **gpg** and **.sig** files to the recipient

6. Receiver unlocks the GPG file using the Reciever's password
```
receiver> gpg --output myFile.txt --decrypt myFile.txt.gpg (enter receiver
password)
```

7. Verify the signature
```
receiver> gpg --output myFile.txt.sha256sum --decrypt myFile.txt.sha256sum.sig
```

additional

check pub GPG keys on host
```
gpg --list-keys --keyid-format LONG --fingerprint
```

check priv keys
```
gpg --list-secret-keys
```

delete pub key from keyring
```
gpg --delete-key D7B5FB7A  (should be something like 2048R/D7BF5B7A)
```

delete priv key from keyring
```
gpg --delete-secret-key "Key name"
```

check which GPG key was used to encrypt a file
```
gpg --list-packets file.gpg
```

Disk / Partitioning / FileSystem

unmount volume
```
umount /mnt/nas1
```

ernel is holding up NFS mount on a bad connection
```
ount -l /mnt/nas1
```

eck what proc is holding up unmounting
```
ser -mv /mnt/nas1
of | grep /mnt/nas1
```

all partitions
```
isk -l
```
```
Disk /dev/sda: 11.3 GB, 11286446080 bytes, 22043840 sectors
```

Resize a logical partition

Expand partition

1. add space to Hard Disk on VM in vCenter or VirtualBox
2. check all partitions, need to resize /opt its 30% full,
```
[root@mrxsplunkidx02 joe]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg0-root  7.8G  1.1G  6.3G  15% /
devtmpfs              1.9G     0  1.9G   0% /dev
tmpfs                 1.9G     0  1.9G   0% /dev/shm
tmpfs                 1.9G  8.6M  1.9G   1% /run
tmpfs                 1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/sda1             976M  110M  799M  13% /boot
/dev/mapper/vg0-home  2.0G  7.0M  1.8G   1% /home
```

```
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000591a7

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     1026047      512000   83  Linux
/dev/sda2         1026048    22042623    10508288   8e  Linux LVM
```

Get ID and type of disk
blkid

Additional Disk checks

http://www.foxhop.net/local-or-san-device-in-linux

check what type of file system is on a partition
```
lsblk -f
```

check disk for errors
```
fsck  (only works for certain filesystems)
```

check what kind of filesystem type
```
df -T
```

show volume groups
```
vgdisplay
```

extend volume group
```
lvextend -r -L+25GB /dev/lvol/name
```

Mount a NetApp device as a local filesystem
```
mount -t nfs -o _netdev,rw,hard,intr,nosuid,dev,bg,nfsvers=3 netappNas01:/netbackup
/netbackup
```

add to /etc/fstab,
```
netappNas01:/netbackup     /netbackup      nfs
_netdev,rw,hard,intr,nosuid,dev,bg,nfsvers=3    0    0
```

check if Disks are local or mounted SAN
```
ls /dev/disk/by-path/   (SANs will have an IP next to path)
```

Increase partition space via vCenter GUI

Problem: current /opt only has 75G of available space, need to add another 20G
```
df -h
/dev/mapper/vg0-opt    80G  1.9G   75G   3% /opt
```

add disk space in vCenter console, increasing disk from 100GB to 120GB

on Centos box check name of scsi device,
```
ls /sys/class/scsi_device/
0:0:0:0
```

rescan scsci bus
```
echo 1 > /sys/class/scsi_device/0\:0\:0\:0/device/rescan
```

check to see if extra space is visible,
```
fdisk -l
Disk /dev/sda: 128.8 GB
```

fdisk /dev/sda
type 'p' - prints out all partitions
type 'n' - create new partition
type 'p' - to make new partition

select the next available sector (default), select default Last Sector

type 'w' to save changes
reboot the VM
once rebooted, type 'fdisk -l', a new partition is added

```
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200     2508799      204800    6  FAT16
/dev/sda3         2508800   209715199   103603200   8e  Linux LVM
/dev/sda4       209715200   251658239    20971520   83  Linux
```

now extend your /dev/mapper/vg0-opt

```
> vgs
VG  #PV #LV #SN Attr   VSize  VFree
  vg0   1   6   0 wz--n- 98.78g    0

> vgextend vg0 /dev/sda4
Volume group "vg0" successfully extended
```

Check to see available PE space (shows 20G of available space)
```
> vgdisplay
Free  PE / Size       639 / <19.97 GiB
```

now resize to full available space, will show 94G of available space

```
> lvextend -l +100%FREE /dev/mapper/vg0-opt
> resize2fs /dev/mapper/vg0-opt
> df -h
/dev/mapper/vg0-opt   100G  1.9G   94G   2% /opt
```

MDADM - Software RAID

Create RAID1 with btrfs on 2 physical disks on Centos 7

check disks
```
lsblk

NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda      8:0    0  40G  0 disk
└─sda1   8:1    0  40G  0 part /
sdb      8:16   0   8G  0 disk
sdc      8:32   0   8G  0 disk
```

remove any existing partitions
```
dd if=/dev/zero of=/dev/sdb bs=512 count=1
```

partition each disk (if using entire disk, can skip this entire Partition section)
```
fdisk /dev/sdb  (do same with /dev/sdc)
n (new partition)
p (primary), select 1, enter, enter
t (select for RAID type), enter "fd"
w (write)
```

RAID examine:
```
mdadm --examine /dev/sd[b-c]
```

Create RAID1
```
mdadm --create /dev/md1 --level=mirror --raid-devices=2 /dev/sd[b-c]1
```
(if no partitions present, remove 1 at end)
```
mdadm: Note: this array has metadata at the start and
    may not be suitable as a boot device.  If you plan to
    store '/boot' on this device please ensure that
    your boot-loader understands md/v1.x metadata, or use
    --metadata=0.90
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md1 started.
```

check RAID
```
mdadm --detail /dev/md1
/dev/md1:
           Version : 1.2
     Creation Time : Fri Aug 28 17:23:24 2020
        Raid Level : raid1
        Array Size : 8382464 (7.99 GiB 8.58 GB)
     Used Dev Size : 8382464 (7.99 GiB 8.58 GB)
      Raid Devices : 2
     Total Devices : 2
       Persistence : Superblock is persistent

       Update Time : Fri Aug 28 17:24:06 2020
             State : clean
    Active Devices : 2
   Working Devices : 2
    Failed Devices : 0
     Spare Devices : 0
```

```
/dev/mapper/vg0-opt   4.8G  1.4G  3.3G  30% /opt
```

3. check logical space
```
lvs
appl vg0 -wi-ao---- 10.00g
home vg0 -wi-ao----  2.00g
opt  vg0 -wi-ao----  5.00g
root vg0 -wi-ao----  8.00g
swap vg0 -wi-ao----  2.00g

check available HD space,
vgdisplay
Free  PE / Size       191 / 5.97 GiB
```

4. Need to add another 5 Gigs to /opt
```
lvextend -r -L +5G /dev/mapper/vg0-opt

to extend ALL remaining free space,
lvextend -l +100%FREE /dev/mapper/vg0-opt
```

5. check the File System type of /opt
```
mount | grep opt
/dev/mapper/vg0-opt on /opt type ext4 (rw,relatime,data=ordered)
```

6. extend physical space
```
resize2fs /dev/mapper/vg0-opt
```

7. check space again, its now 15% full
```
df -h
/dev/mapper/vg0-opt   9.8G  1.4G  8.0G  15% /opt

check logical volume again,
lvs
opt  vg0 -wi-ao---- 10.00g
```

Shrink Partition

need to shrink partition /appl from 2GB to 1GB

```
lvs

appl vg0 -wi-ao---- 2.00g
docker vg0 -wi-ao---- 10.00g
home vg0 -wi-ao---- 60.00g
opt vg0 -wi-ao---- 5.00g
root vg0 -wi-ao---- 8.00g
swap vg0 -wi-ao---- 2.00g
tmp vg0 -wi-ao---- 3.00g
var vg0 -wi-ao---- 3.00g
```

unmount it
```
umount -v /appl
```

get filesystem name
```
df -h
/dev/mapper/vg0-appl   /appl
```

check for file system error
```
e2fsck -ff /dev/mapper/vg0-appl
```
(must pass all 5 stages)

reduce FS by 1GB
```
resize2fs /dev/mapper/vg0-appl 1G
```

reduce the logical volume
```
lvreduce -L -1G /dev/mapper/vg0-appl
```

mount /appl back on
```
mount /dev/mapper/vg0-appl /appl
```

LVM

check size of partition
```
lvdisplay /appl
```

```
--- Logical volume ---
LV Path                /dev/vg0/appl
LV Name appl
VG Name vg0
LV UUID Aim8Q2-gxp2-jnT0-OcS2-d3To-n5Nd-IJmvxo
LV Write Access read/write
LV Creation host, time xxxx, 2018-02-23 11:52:48 -0500
LV Status available # open 1
LV Size 1.00 GiB Current LE 32 Segments 1
Allocation inherit
Read ahead sectors auto - currently set to 8192
Block device 253:6
```

Remove Swap LV and merge it into Root LV

want to remove 4GB swap LV and merge it into root, to give root more space,
```
/dev/mapper/centos-root   50G  1.2G   49G   3% /
```

1. unmount and deactivate Swap LV
```
lvchange -a n /dev/mapper/centos-swap
```

2. remove it
```
lvremove /dev/mapper/centos-swap
```

3. extend Root volume
```
lvextend -l +100%FREE /dev/mapper/centos-root
```

4. Grow the Root volume
```
resize2fs /dev/mapper/centos-root
(if XFS filesystem, use xfs_growfs /dev/mapper/centos-root)
```

rename logical volume group
```
vgdisplay (show all groups)
```

rename group centos to 'hc'
```
vgrename /dev/centos /dev/hc
```

rename logical volume
```
lvrename /dev/hc/disk1 /dev/hc/disk2
```

Remove Swapfile from /home and create new Swap LVM

```
swapoff -a
```

Mount an EC2 volume as /home

attach volume to instance
```
on ec2:
lsblk
```

should be listed as nvme1n1 or similar name

check if filesystem has data
```
file -s /dev/nvme1n1
```

if shows 'data', means volume is empty

create new volume
```
mkfs -t xfs /dev/nvme1n1
```

create new mount point
```
mkdir /home2
```

mount filesystem
```
mount /dev/nvme1n1 /home2
mv /home to /home_old, move /home2 to /home
```

permanent mount, get ID of volume
```
blkid  (get UUID)
```

vi /etc/fstab
```
UUID=<insert ID>    /home  xfs  defaults  0  0
```

remount
```
mount -a
```

Mount volume on EC2 as swap

create volume, gp3 max iops (swap has to have fast read/write speed)
attach volume to instance
on instance,

```
lsblk (get new device name)
fdisk /dev/nvme3n1
n # new partition
t # partition type
82  # swap hex code
```

```
Consistency Policy : resync

              Name : min1:1  (local to host min1)
              UUID : fae3d35a:862d0521:eb39400c:b9a794f0
            Events : 17

    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       1       8       33        1      active sync   /dev/sdc1
```

create BTRFS filesystem
```
mkfs.btrfs /dev/md1
```

mount the filesystem
```
mount /dev/md1 /home (or mountpoint)
```

add to /etc/fstab
```
/dev/md1                /home             btrfs    defaults         0 0
```

create RAID config file
```
mdadm --detail --scan -v > /etc/mdadm.conf
```

test RAID1 by simulating drive failure
```
mdadm --manage --set-faulty /dev/md1 /dev/sdc1
```

check RAID status (will show DEGRADED)
```
mdadm --detail /dev/md1
```

setup alerts for Disk failure, add to /etc/mdadm.conf
```
MAILTO <your email addr>
DEVICE partitions
```

put scan in daemon
```
mdadm --monitor --scan --daemonize
```

add to kernel to start on boot
vi /etc/rc.local

add to bottom
```
/sbin/mdadm --monitor --scan --daemonize
```

add disk to array as hotspare
```
mdadm --grow /dev/md1 --add /dev/sdg1
```

add disk as full device
```
mdadm --grow /dev/md1 --add /dev/sdg1 --raid-devices=3
```

remove disk from array
```
mdadm /dev/md1 --fail /dev/sdc1 --remove /dev/sdc1
mdadm --grow /dev/md1 --raid-devices=2
```

use full size of array
```
mdadm --grow /dev/md1 --size=max
resize2fs /dev/md1
```

stop RAID
```
mdadm --stop /dev/md0
```

start RAID
```
mdadm --assemble /dev/md0 /dev/sdb1 /dev/sdc1
```

```
w # save
```

```
mkswap /dev/nvme3n1p1   # partition name
```

get the UUID of swap
blkid (find the partition UUID)
add to /etc/fstab
```
UUID=<UUID number> swap swap default 0 0

swapon /dev/nvme3n1p1
mount -a
```

BTRFS

check file on inode
```
btrfs inspect-internal inode-resolve 154326924 /mnt/hc
```

Extend diskspace on EC2 instance (T3)
'/' currently at 94% usage

```
/dev/xvda1

go to EC2 console, click on instance > attached volumes > click on Volume >
modify > expand, add the additional disk space

root@host> lsblk
xvda     TYPE=disk
---xvda1 TYPE=part  /

growpart /dev/xvda 1

yum install xfsprogs

xfs_growfs -d /   ## will resize / partition to full
```

Check disk health for bad sectors

1. get disk name
   ```
   lsblk | grep disk
   ```
2. check blocks (read only)
   ```
   sudo badblocks -v /dev/sda > badsector.txt
   ```

check disk read/write speed
```
hdparm -tv --direct /dev/sda
```

Regex / awk / sed / tr

find all lines starting with #
```
^#.*$
```

find blank line
```
^\s*$
```

remove leading and trailing commas from string
```
str=",there was a loud, bang, there,"
str="${str#,}"
str="${str%,}"
echo $str  >> "there was a loud, bang, there"
```

SED

Replace string in a file (write directly to file -i)
```
sed -i -e "s/${prev_version}/${version}/g" bitbucket.service
```

Replace anything between 2 delimeters "!!" with word "super"
```
sed -e 's/!.*!/super/g' /etc/file
```

remove whitespace
```
sed -i "s/ //g" file  # replace inline
sed "s/[[:space:]]//g" file  # replace just on screen
```

remove 2nd line from top, from file
```
sed -i '2,$d' file
```

replace newline with comma
```
sed ':a;N;$!ba;s/\n/ /g'
```

Remove leading spaces and tabs
```
sed 's/^[ \t]*//'
```

Remove single spaces only (leave multiple spaces)
```
sed 's/\(.\) //g'
```

Reduce multiple spaces to one
```
sed 's/ \+/ /g'
```

Replace multiple newlines with a single newline
```
sed '/^$/N;/^\n$/D' file.txt
```

Delete text in a line between two markers {}
```
sed -e 's/\({\).*\(}\)//'
```

Remove empty lines
```
sed '/^\s*$/d'
```

Remove all but the first line matching pattern
```
sed '2,${/pattern/d;}'
```

Remove only the first line matching pattern
```
sed '0,/pattern/{/pattern/d;}'
```

Move the first line to the end of the list
```
sed '1,1{H;1h;d;};$G'
```

Remove non-alphanumeric characters from words
```
sed 's/[^[:alnum:].-\ ]//g'
```

Reduce multiple spaces to one for a line containing a string
```
iostat | sed -n '/^sd/s/ \+/ /gp'
```

insert "apple" into beginning of a file
```
sed 's/^/apple /' file1
```

insert apple into end of file
```
sed 's/$/apple /' file1
```

replace orange with apple, only on 3rd line of text
```
sed '3s/orange/apple/g' file1
```

replace orange with apple from 1st to 3rd line
```
sed '1,3s/orange/apple/g' file1
```

replace multiple words, orange with apple, red with blue
```
sed 's/orange/apple/g; s/red/blue/g' file1
```

remove 1st occurence of specific character "b"
```
sed 's/b//' file1
```

remove all instances of "b" in file
```
sed 's/b//g' file1
```

remove last character of every line
```
sed 's/.$//' file1
```

remove "b" only if its last character in line
```
sed 's/b$//' file1
```

remove "b" only if its last character in line
```
sed 's/b$//' file1
```

remove all numbers in every line of a file
```
sed 's/[0-9]//g' file1
```

TR

remove whitespace from string
```
fields="$(echo -e "${fields}" | tr -d '[:space:]')"
```

or use xargs
```
echo $fields | xargs  (will strip leading and trailing whtiespace)
```

strip double quotes
```
echo '"string"' | tr -d '"'
```

Identify server's primary IP address
/sbin/ifconfig | sed -rn 's/127.0.0.1//;s/.*inet (addr:)?(([0-9]*\.){3}[0-9]*).*//p'

split a string by a delimeter
```
string="apple, cherry, banana"
first=$(echo $string | cut -d',' -f1)
second=$(echo $string | cut -d',' -f2)
```

Remove non-printable characters from files
tr -cd '-6' < infile > outfile

XARGS

find all files with JPG extension and rename each file to be JPEG2 (xargs -i)
```
ls | grep ".jpg" | xargs -I {} mv {} {}.jpeg2
```

examples taken from: https://www.igoroseledko.com/awk-sed-snippets-for-sysadmins/

AWK

Built-In AWK functions full list of functions

to Uppercase
awk -F: '{print toupper($1)}' file.txt
NAME: JOE

print entire line ($0)
joe:employees:123
bob:employees:222
awk -F":" '{print $0}' file.txt

Awk If-Else
```
awk -F: '{if($1=="name") print $2;else print "NONE"}' file.txt
joe
```

find by Regex
```
awk -F: '/ing$/' file.txt
status: thinking
occupation: moving
```

print all values that match 1st column = "color"
cat file.txt
color: red
size: 25
color: blue
size: 50

```
awk -F: '$1 ~ /color$/ {print $2}' file.txt
red
blue
```

split string by delimeter
```
third=$(echo $string | awk -F"," '{print $3}')
```

or use delimeter flag
```
echo $string | awk  '{print $1,$3}' FS=","
```

Remove commas inside double-quotes
awk -F"" -v OFS='' '{ for (i=2; i<=NF; i+=2) gsub(",", "", $i) } 1'

Remove duplicate words in a line
```
awk '{ while(++i<=NF) printf (!a[$i]++) ? $i FS : ""; i=split("",a); print "" }'
```

Remove duplicate lines in a file without sorting
cat file | awk '!a[$0]++'

Print number of characters for each line in a file
awk '{ print length($0)"\t"$0; }' file.txt

Begin and End Functions
```
awk -F: 'BEGIN {print "this is beginning"} {print $0} END {print "End!"}' file.txt
this is beggining
name: joe
End!
```

remove all duplicate entries from a file
```
awk '!x[$0]++' filename
```

```
cat file.txt
apple,300
grape,200
grape,400
apple,500
banana,200
```

print only lines with word "apple"
```
awk '/apple/' file.txt
```

print only "grape" record, print 2nd column only
```
awk '$0 ~ /grape/{print $2}' file.txt
```

print any line that does not contain "apple"
```
awk '!/apple/' file.txt
```

print any line that has grape or banana
```
awk -F, '$1 ~ /^grape|^banana/' file.txt
```

print any line where number is greater than 200
```
awk -F, '$2>200' file.txt

print any line thats greater than 200 or has "grape"
```

Remove entire words containing non-alphabetic characters
awk '{ofs=""; for (i=1; i<=NF; i++) if ($i ~ /^[[:alpha:]]+$/) {printf "%s%s", ofs, $i; ofs=OFS} print "" }'

Sample "temp" file
```
ID1,223
ID2,124
ID3,125
ID2,400
```

Add up values in second column
```
awk -F"," '{s+=$2}END{print s}' temp
```

Add up the values in the second column only for ID2
```
awk -F, '$1=="ID2"{s+=$2;}END{print s}' temp
v="ID2"; awk -F, -v v="${v}" '$1==v{s+=$2;}END{print s}' temp
```

List unique values in 1st column
```
awk -F, '{a[$1];}END{for (i in a)print i;}' temp
```

Add up values in the second column for each ID
```
awk -F, '{a[$1]+=$2;}END{for(i in a)print i", "a[i];}' temp
```

Remove only the first line matching pattern
```
awk '!/pattern/ || f++'
```

Remove all but the first line matching pattern
```
awk '/pattern/&&f++ {next} 1'
```

Show allocated disk space
```
df -klP -t xfs -t ext2 -t ext3 -t ext4 -t reiserfs | grep -oE ' [0-9]{1,}( +[0-9]
{1,})+' | awk '{sum_used += $1} END {printf "%.0f GB\n", sum_used/1024/1024}'
```

Sysdig

install sysdig
```
curl -s https://s3.amazonaws.com/download.draios.com/stable/install-sysdig
   | sudo bash
```

write to scap file, 200MB in size, keep only 5 files
```
sysdig -C 200 -W 5 -w dump.scap
```

search for specific port
```
sysdig fd.port=8335
```

search for specific process
```
sysdig proc.name=sudo
```

see every action done by every user
```
sysdig -c spy_users
```

Sysdig Network

see top procs in network bandwidth usage
```
sysdig -c topprocs_net
```

show network data exchanged with host 192.168.38.5
```
sysdig -s2000 -A -c echo_fds fd.cip=192.168.38.5
```

show top client IPs in terms of established connections
```
sysdig -c fdcount_by fd.cip "evt.type=accept"
```

in terms of total bytes
```
sysdig -c fdbytes_by fd.cip
```

see all GET HTTP requests made
```
sudo sysdig -s 2000 -A -c echo_fds fd.port=80 and evt.buffer contains GET
```

Sysdig disk IO

see top procs in terms of bandwidth usage
```
sysdig -c topprocs_file
```

list procs that use high number of files
```
sysdig -c fdcount_by proc.name "fd.type=file"
```

see top files in r/w bytes
```
sysdig -c topfiles_bytes
```

see top directories in terms of r/w disk activity
```
sysdig -c fdbytes_by fd.directory "fd.type=file"
```

in specific directory
```
sysdig -c fdbytes_by fd.filename "fd.directory=/tmp/"
```

observe IO activity on all files named "passwd"
```
sysdig -A -c echo_fds "fd.filename=passwd"
```

show top procs in terms of IO errors
```
sysdig -c topprocs_errors
```

Sysdig Proc and CPU

show top procs by CPU usage
```
sysdig -c topprocs_cpu
```

observe standard output of proc
```
sysdig -s4096 -A -c stdout proc.name=myproc
```

Sysdig security

show all file opens in /etc directory
```
sysdig evt.type=open and fd.name contains /etc
```

Show the ID of all the login shells that have launched the "tar" command
```
sysdig -r file.scap -c list_login_shells tar
```

Show all the commands executed by the login shell with the given ID
```
sysdig -r trace.scap.gz -c spy_users proc.loginshellid=5459
```

Augeas

example file content: /etc/pam.d/password-auth

```
auth            required        pam_env.so
auth            required        pam_faildelay.so delay=2000000
auth [success=1 default=bad] pam_unix.so nullok try_first_pass
auth            requisite       pam_succeed_if.so uid >= 1000 quiet_success
auth            required        pam_deny.so

account         required        pam_unix.so
account         sufficient      pam_localuser.so
account         sufficient      pam_succeed_if.so uid < 1000 quiet
account         required        pam_permit.so

password        requisite       pam_pwquality.so try_first_pass local_users_only retry=3
authtok_type=
password        sufficient  pam_unix.so sha512 shadow nullok try_first_pass use_authtok


password        required        pam_deny.so

session         optional        pam_keyinit.so revoke
session         required        pam_limits.so
-session        optional        pam_systemd.so
session         [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session         required        pam_unix.so
password sufficient pam_unix.so remember=15
auth    required        pam_faillock.so preauth audit   silent  deny=5  unlock_time=900
auth    [default=die]   pam_faillock.so authfail        audit   deny=5  unlock_time=900
auth    sufficient  pam_faillock.so authsucc    audit   deny=5  unlock_time=900
password requisite pam_pwquality.so try_first_pass retry=3
```

search through tree for specific string
```
augtool> print /files/etc/pam.d/password-auth/*[module='pam_unix.so']/argument[. =
'remember=15']
```

search using a Regex
```
augtool> print /files/etc/pam.d/password-auth/*[module='pam_unix.so']
[type='password'][control='sufficient']/argument[ . =~ regexp("remember=.*")]
```

get last node of a tree
```
augtool> print /files/etc/pam.d/password-auth/*[module='pam_unix.so']
[type='password'][control='sufficient'][last()]
```

add Argument value on last node, last argument position
```
set /files/etc/pam.d/password-auth/*[module='pam_unix.so'][type='password']
[control='sufficient'][last()]/argument[last()+1] 'remember=33'
```

Snap

snap alias
```
snap install python38
snap alias python38 python3.8 (creates symlink /usr/bin/python3.8)
```

Troubleshoot

slow / frozen system

check if procs are in uninterrupted sleep state (waiting for IO and causing slowness)
ps aux (check STAT column, will show procs that are in uninterrupted sleep)

check paging faults

sar -B 2 5 will generate paging report, check majflt column, major faults per second, if high #, means system is out of RAM

SysRQ

enable SysRQ to kill procs that are in 'uninterrupted sleep' state. SysRQ will respond even in frozen state (assuming command line is responsive)

configure server to have sysrq enabled

echo 1 > /proc/sys/kernel/sysrq

add to sysctl

sysctl -w kernel.sysrq=1

1. enable sysrq
2. kill D state procs

Check why a server rebooted
sudo ausearch -i -m system_boot,system_shutdown | tail -4

show all reboots
journalctl --list-boots

VMstat - CPU,Memory, I/O usage

check swap and memory allocation
```
vmstat -S M 5 (updates every 5 seconds)
```

```
Proc
r: The number of runnable processes. These are processes that have been launched and are either running or are waiting for their next time-sliced burst of CPU cycles.
b: The number of processes in uninterruptible sleep. The process isn't sleeping, it is performing a blocking system call, and it cannot be interrupted until it has
completed its current action. Typically the process is a device driver waiting for some resource to come free. Any queued interrupts for that process are handled when
the process resumes its usual activity.

Memory
swpd: the amount of virtual memory used. In other words, how much memory has been swapped out.,
free: the amount of idle (currently unused) memory.
buff: the amount of memory used as buffers.
cache: the amount of memory used as cache.
Swap
si: Amount of virtual memory swapped in from swap space.
so: Amount of virtual memory swapped out to swap space.
IO
bi: Blocks received from a block device. The number of data blocks used to swap virtual memory back into RAM.
bo: Blocks sent to a block device. The number of data blocks used to swap virtual memory out of RAM and into swap space.
System
in: The number of interrupts per second, including the clock.
cs: The number of context switches per second. A context switch is when the kernel swaps from system mode processing into user mode processing.
CPU
These values are all percentages of the total CPU time.
us: Time spent running non-kernel code. That is, how much time is spent in user time processing and in nice time processing.
sy: Time spent running kernel code.
id: Time spent idle.
wa: Time spent waiting for input or output.
st: Time stolen from a virtual machine. This is the time a virtual machine has to wait for the hypervisor to finish servicing other virtual machines before it can come
back and attend to this virtual machine.
```

/proc pseudofiles

/proc/PID/cmdline - Command line arguments.
/proc/PID/cpu - Current and last cpu in which it was executed.
/proc/PID/cwd - Link to the current working directory.
/proc/PID/environ - Values of environment variables.
/proc/PID/exe - Link to the executable of this process.
/proc/PID/fd - Directory, which contains all file descriptors.
/proc/PID/maps - Memory maps to executables and library files.
/proc/PID/mem - Memory held by this process.
/proc/PID/root - Link to the root directory of this process.
/proc/PID/stat - Process status.
/proc/PID/statm - Process memory status information.
/proc/PID/status - Process status in human readable form.
/proc/apm - Advanced power management info.
/proc/bus - Directory containing bus specific information.
/proc/cmdline - Kernel command line.
/proc/cpuinfo - Information about the processor, such as its type, make, model, and performance.
/proc/devices - List of device drivers configured into the currently running kernel (block and character).
/proc/dma - Shows which DMA channels are being used at the moment.
/proc/driver -Various drivers grouped here, currently rtc
/proc/execdomains - Execdomains, related to security.
/proc/fb - Frame Buffer devices.
/proc/filesystems - Filesystems configured/supported into/by the kernel.
/proc/fs - File system parameters, currently nfs/exports.
/proc/interrupts - Shows which interrupts are in use, and how many of each there have been.
/proc/iomem - Memory map.
/proc/ioports - Which I/O ports are in use at the moment.
/proc/irq - Masks for irq to cpu affinity.
/proc/isapnp - ISA PnP (Plug&Play) Info.
/proc/kmsg - Messages output by the kernel. These are also routed to syslog.
/proc/ksyms - Kernel symbol table.
/proc/loadavg - The 'load average' of the system; three indicators of how much work the system has done during the last 1, 5 & 15 minutes.
/proc/locks - Kernel locks.
/proc/meminfo - Information about memory usage, both physical and swap. Concatenating this file produces similar results to using 'free' or the first few lines of 'top'.
/proc/misc - Miscellaneous pieces of information. This is for information that has no real place within the rest of the proc filesystem.
/proc/modules - Kernel modules currently loaded. Typically its output is the same as that given by the 'lsmod' command.
/proc/mounts - Mounted filesystems (same as running 'mount' cmd)
/proc/net - Status information about network protocols.
/proc/stat - Overall/various statistics about the system, such as the number of page faults since the system was booted.
/proc/swaps - Swap space utilization
/proc/sys - This is not only a source of information, it also allows you to change parameters within the kernel without the need for recompilation or even a system reboot. Take care when attempting this as it can both optimize your system and also crash it.
/proc/sys/fs - Contains file system data. This subdirectory contains specific file system, file handle, inode, dentry and quota information.

/proc/sys/vm - The files in this directory can be used to tune the operation of the virtual memory (VM) subsystem of the Linux kernel. In addition, one of the files (bdflush) has some influence on disk usage.

nfract - This parameter governs the maximum number of dirty buffers in the buffer cache. Dirty means that the contents of the buffer still have to be written to disk (as opposed to a clean buffer, which can just be forgotten about). Setting this to a higher value means that Linux can delay disk writes for a long time, but it also means that it will have to do a lot of I/O at once when memory becomes short. A lower value will spread out disk I/O more evenly.
ndirty - Ndirty gives the maximum number of dirty buffers that bdflush can write to the disk at one time. A high value will mean delayed, bursty I/O, while a small value can lead to memory shortage when bdflush isn't woken up often enough.
nrefill - This is the number of buffers that bdflush will add to the list of free buffers when refill_freelist() is called. It is necessary to allocate free buffers beforehand, since the buffers are often different sizes than the memory pages and some bookkeeping needs to be done beforehand. The higher the number, the more memory will be wasted and the less often refill_freelist() will need to run.
nref_dirt - When refill_freelist() comes across more than nref_dirt dirty buffers, it will wake up bdflush.
age_buffer, age_super - Finally, the age_buffer and age_super parameters govern the maximum time Linux waits before writing out a dirty buffer to disk. The value is expressed in jiffies (clockticks), the number of jiffies per second is 100. Age_buffer is the maximum age for data blocks, while age_super is for filesystems meta data.
buffermem - The three values in this file control how much memory should be used for buffer memory. The percentage is calculated as a percentage of total system memory. The values are:

```
min_percent  - This is the minimum percentage of memory that should be spent on buffer memory.

borrow_percent - When Linux is short on memory, and the buffer cache uses more than it has been allotted, the memory management (MM) subsystem will prune the buffer cache more heavily than other
memory to compensate.

max_percent - This is the maximum amount of memory that can be used for buffer memory.
freepages - This file contains three values: min, low and high:

min - When the number of free pages in the system reaches this number, only the kernel can allocate more memory.
low - If the number of free pages falls below this point, the kernel starts swapping aggressively.
high - The kernel tries to keep up to this amount of memory free; if memory falls below this point, the kernel starts gently swapping in the hopes that it never has to do really aggressive
swapping.

kswapd - Kswapd is the kernel swap out daemon. That is, kswapd is that piece of the kernel that frees memory when it gets fragmented or full. Since every system is different, you'll probably want
some control over this piece of the system.  The file contains three numbers:
        tries_base - The maximum number of pages kswapd tries to free in one round is calculated from this number. Usually this number will be divided by 4 or 8 (see mm/vmscan.c), so it isn't as
big as it looks. When you need to increase the bandwidth to/from swap, you'll want to increase this number.
     tries_min - This is the minimum number of times kswapd tries to free a page each time it is called. Basically it's just there to make sure that kswapd frees some pages even when it's being
called with minimum priority.
     swap_cluster - This is probably the greatest influence on system performance. swap_cluster is the number of pages kswapd writes in one turn. You'll want this value to be large so that kswapd
does its I/O in large chunks and the disk doesn't have to seek as often, but you don't want it to be too large since that would flood the request queue.

overcommit_memory - This file contains one value. The following algorithm is used to decide if there's enough memory: if the value of overcommit_memory is positive, then there's always enough
memory. This is a useful feature, since programs often malloc() huge amounts of memory 'just in case', while they only use a small part of it. Leaving this value at 0 will lead to the failure of
such a huge malloc(), when in fact the system has enough memory for the program to run. On the other hand, enabling this feature can cause you to run out of memory and thrash the system to death,
so large and/or important servers will want to set this value to 0.

pagecache - This file does exactly the same job as buffermem, only this file controls the amount of memory allowed for memory mapping and generic caching of files. You don't want the minimum level
to be too low, otherwise your system might thrash when memory is tight or fragmentation is high.

pagetable_cache - The kernel keeps a number of page tables in a per-processor cache (this helps a lot on SMP systems). The cache size for each processor will be between the low and the high value.
On a low-memory, single CPU system, you can safely set these values to 0 so you don't waste memory. It is used on SMP systems so that the system can perform fast pagetable allocations without
having to acquire the kernel memory lock. For large systems, the settings are probably fine. For normal systems they won't hurt a bit. For small systems ( less than 16MB ram) it might be
advantageous to set both values to 0.

swapctl - This file contains no less than 8 variables. All of these values are used by kswapd. The first four variables sc_max_page_age, sc_page_advance, sc_page_decline and sc_page_initial_age are
used to keep track of Linux's page aging. Page ageing is a bookkeeping method to track which pages of memory are often used, and which pages can be swapped out without consequences.
```

/proc/sys/net/core - Network core options
rmem_default - The default setting of the socket receive buffer in bytes.
rmem_max - The maximum receive socket buffer size in bytes.
wmem_default - The default setting (in bytes) of the socket send buffer.
wmem_max - The maximum send socket buffer size in bytes.
message_burst and message_cost - These parameters are used to limit the warning messages written to the kernel log from the networking code. They enforce a rate limit to make a denial-of-service attack impossible. A higher message_cost factor, results in fewer messages that will be written. Message_burst controls when messages will be dropped. The default settings limit warning messages to one every five seconds.
netdev_max_backlog - Maximum number of packets, queued on the INPUT side, when the interface receives packets faster than kernel can process them.
optmem_max - Maximum ancillary buffer size allowed per socket. Ancillary data is a sequence of struct cmsghdr structures with appended data.