A few GNU/Linux basics

System information tools

top

- Basically a CLI taskmanager for Linux
- Live load information

```
TOP(1)                          User Commands                          TOP(1)

NAME
       top - display Linux processes

SYNOPSIS
       top -hv|-bcHiOSs -d secs -n max -u|U user -p pid -o fld -w [cols]

       The traditional switches '-' and whitespace are optional.

DESCRIPTION
       The  top program provides a dynamic real-time view of a running system.  It can dis-
       play system summary information as well as a list of processes or threads  currently
       being  managed  by  the Linux kernel.  The types of system summary information shown
       and the types, order and size of information displayed for processes  are  all  user
       configurable and that configuration can be made persistent across restarts.

       The  program  provides  a  limited interactive interface for process manipulation as
       well as a much more extensive interface for personal configuration  --  encompassing
       every  aspect  of its operation.  And while top is referred to throughout this docu-
       ment, you are free to name the program anything you wish.  That new  name,  possibly
       an  alias, will then be reflected on top's display and used when reading and writing
       a configuration file.
```

```
top - 07:40:46 up 1 min,  1 user,  load average: 0.24, 0.11, 0.04
Tasks: 122 total,   1 running, 121 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1015852 total,   186132 free,   356156 used,   473564 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   490912 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1610 arangodb  20   0 1514028 274404 136908 S   0.7 27.0   0:01.34 arangodb [serve
 1367 root      20   0  299104  21652  13236 S   0.3  2.1   0:00.31 docker-containe
    1 root      20   0   38000   6008   3960 S   0.0  0.6   0:02.43 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.02 ksoftirqd/0
    4 root      20   0       0      0      0 S   0.0  0.0   0:00.03 kworker/0:0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kworker/u4:0
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.04 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/1
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   16 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   17 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 perf
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.00 khungtaskd
```

ps

- Shows a snapshot of current processes
- Defaults to only user's processes that have associated terminals
  - 'a' flag removes user restriction
  - 'x' flag removes tty restriction

```
PS(1)                          User Commands                          PS(1)

NAME
       ps - report a snapshot of the current processes.

SYNOPSIS
       ps [options]

DESCRIPTION
       ps displays information about a selection of the active processes.  If you want a
       repetitive update of the selection and the displayed information, use top(1)
       instead.

       This version of ps accepts several kinds of options:

       1   UNIX options, which may be grouped and must be preceded by a dash.
       2   BSD options, which may be grouped and must not be used with a dash.
       3   GNU long options, which are preceded by two dashes.

       Options of different types may be freely mixed, but conflicts can appear.  There are
       some synonymous options, which are functionally identical, due to the many standards
       and ps implementations that this ps is compatible with.

       Note that "ps -aux" is distinct from "ps aux".  The POSIX and UNIX standards require
       that "ps -aux" print all processes owned by a user named "x", as well as printing
       all processes that would be selected by the -a option.  If the user named "x" does
       not exist, this ps may interpret the command as "ps aux" instead and print a
       warning.  This behavior is intended to aid in transitioning old scripts and habits.
       It is fragile, subject to change, and thus should not be relied upon.

       By default, ps selects all processes with the same effective user ID (euid=EUID) as
       the current user and associated with the same terminal as the invoker.  It displays
       the process ID (pid=PID), the terminal associated with the process (tname=TTY), the
       cumulated CPU time in [DD-]hh:mm:ss format (time=TIME), and the executable name
       (ucmd=CMD).  Output is unsorted by default.
```

```
vagrant@ubuntu-xenial:~$ ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.4  0.5  38000  6008 ?        Ss   07:39   0:02 /sbin/init
root          2  0.0  0.0      0     0 ?        S    07:39   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        S    07:39   0:00 [ksoftirqd/0]
root          4  0.0  0.0      0     0 ?        S    07:39   0:00 [kworker/0:0]
root          5  0.0  0.0      0     0 ?        S<   07:39   0:00 [kworker/0:0H]
root          6  0.0  0.0      0     0 ?        S    07:39   0:00 [kworker/u4:0]
root          7  0.0  0.0      0     0 ?        S    07:39   0:00 [rcu_sched]
root          8  0.0  0.0      0     0 ?        S    07:39   0:00 [rcu_bh]
root          9  0.0  0.0      0     0 ?        S    07:39   0:00 [migration/0]
root         10  0.0  0.0      0     0 ?        S    07:39   0:00 [watchdog/0]
root         11  0.0  0.0      0     0 ?        S    07:39   0:00 [watchdog/1]
root         12  0.0  0.0      0     0 ?        S    07:39   0:00 [migration/1]
```

df and du

- df
  - Shows the filesystems present, what they are mounted and other information
- du
  - Shows estimated disk usage of the given file/directory

```
vagrant@ubuntu-xenial:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             498596        0    498596   0% /dev
tmpfs            101588     3180     98408   4% /run
/dev/sda1      10098468  4669232   5412852  47% /
tmpfs            507924        4    507920   1% /dev/shm
tmpfs              5120        0      5120   0% /run/lock
tmpfs            507924        0    507924   0% /sys/fs/cgroup
vagrant        30031232 12358712  17672520  42% /vagrant
vagrantShare   30031232 12358712  17672520  42% /vagrantShare
tmpfs            101588        0    101588   0% /run/user/1000
vagrant@ubuntu-xenial:~$ du
8       ./.ssh
4       ./.cache
8       ./.config/htop
12      ./.config
8       ./.vim
4       ./.ansible/tmp
8       ./.ansible
84      .
```

```
vagrant@ubuntu-xenial:~$ du -c -d 1 /usr/
293564  /usr/share
4       /usr/games
765380  /usr/bin
493948  /usr/src
73020   /usr/sbin
3256    /usr/include
108     /usr/local
368484  /usr/lib
1997768 /usr/
1997768 total
vagrant@ubuntu-xenial:~$
```

Finding files

find

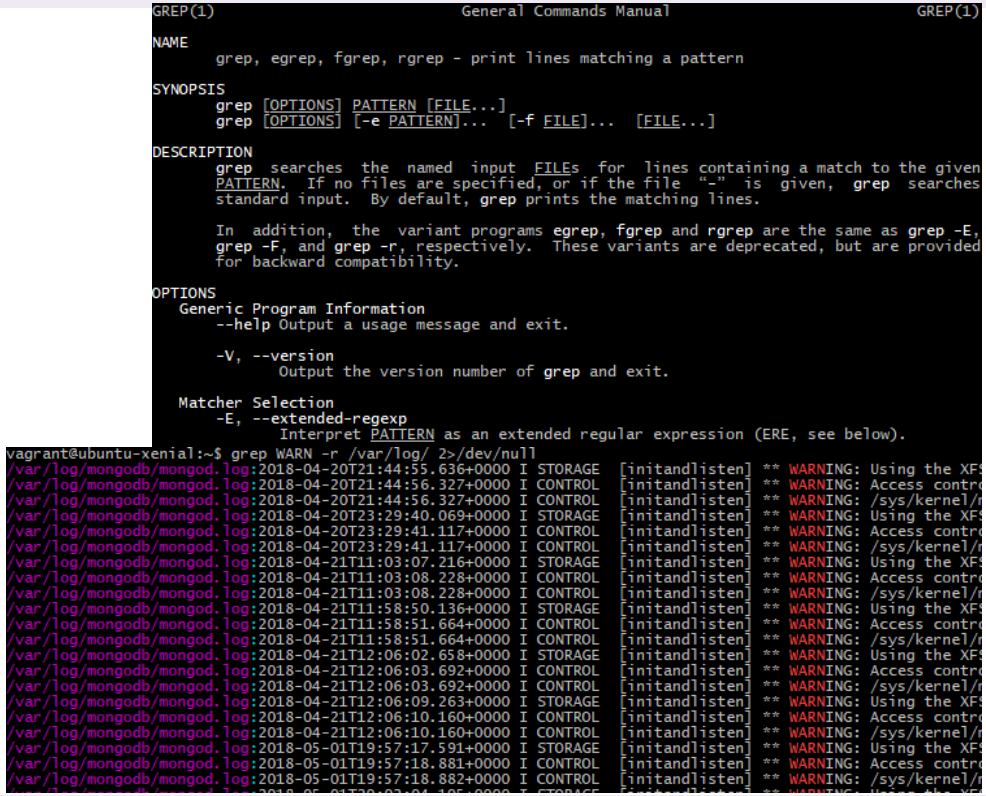
- Allows you to find files based on specified criteria:
  - File size
  - File type
  - Name pattern
  - Path pattern
- Useful for piping into other tools

```
FIND(1)                          General Commands Manual                          FIND(1)

NAME
       find - search for files in a directory hierarchy

SYNOPSIS
       find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...] [expression]

DESCRIPTION
       This manual page documents the GNU version of find.  GNU find searches the direc-
       tory tree rooted at each given starting-point by evaluating the given  expression
       from left to right, according to the rules of precedence (see section OPERATORS),
       until the outcome is known (the left hand side is false for and operations,  true
       for  or),  at  which  point find moves on to the next file name.  If no starting-
       point is specified, `.' is assumed.

       If you are using find in an environment where security is important (for  example
       if  you are using it to search directories that are writable by other users), you
       should read the "Security Considerations" chapter of the findutils documentation,
       which  is  called  Finding  Files  and comes with findutils.  That document also
       includes a lot more detail and discussion than this manual page, so you may  find
       it a more useful source of information.

OPTIONS
       The  -H, -L and -P options control the treatment of symbolic links.  Command-line
       arguments following these are taken to be names of files  or  directories  to  be
       examined,  up  to the first argument that begins with `-', or the argument `(' or
       `!'.  That argument and any following arguments are taken to  be  the  expression
       describing what is to be searched for.  If no paths are given, the current direc-
       tory is used.  If no expression is given, the expression -print is used (but  you
       should probably consider using -print0 instead, anyway).
```

```
vagrant@ubuntu-xenial:~$ find / -path **/*js 2>/dev/null
/var/lib/dpkg/alternatives/nodejs
/etc/alternatives/nodejs
/home/vagrant/.mongorc.js
/usr/share/doc/nodejs
/usr/share/doc/nodejs/api_assets/sh_javascript.min.js
/usr/share/doc/nodejs/api_assets/sh_main.js
/usr/share/doc/nodejs/api_assets/dnt_helper.js
/usr/share/doc/nodejs/api/assets/sh_javascript.min.js
/usr/share/doc/nodejs/api/assets/sh_main.js
/usr/share/doc/nodejs/api/assets/dnt_helper.js
/usr/share/arangodb3/js
/usr/share/arangodb3/js/node/stream.js
```

grep

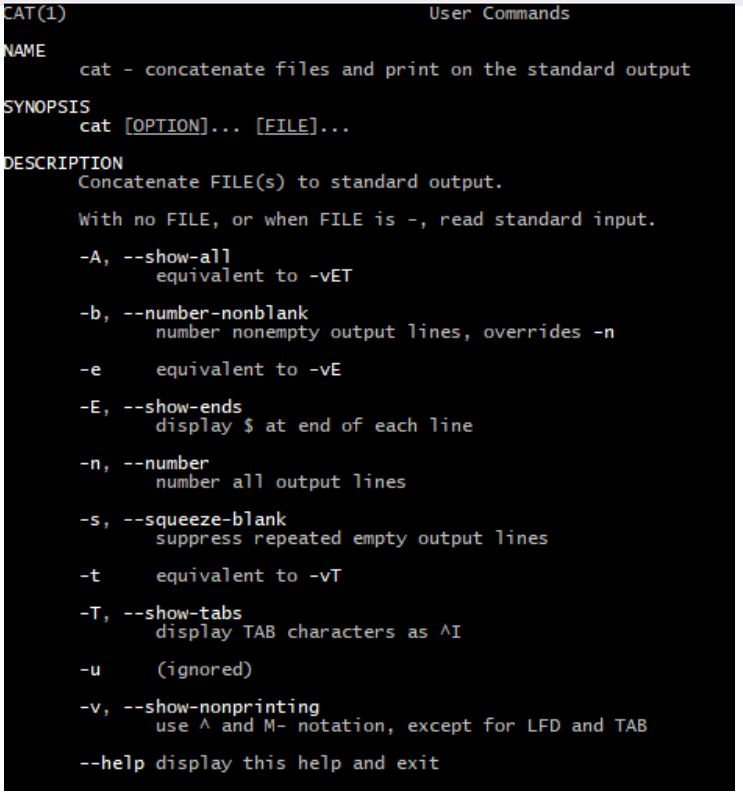
- Grep allows you to perform string searches (and regexes) in files and directories
- Can be used to grab:
  - File names containing the string
  - Matched lines
  - Lines *near* matched lines



View file contents

cat

- Print contents of a text file to terminal
- The simplest text display



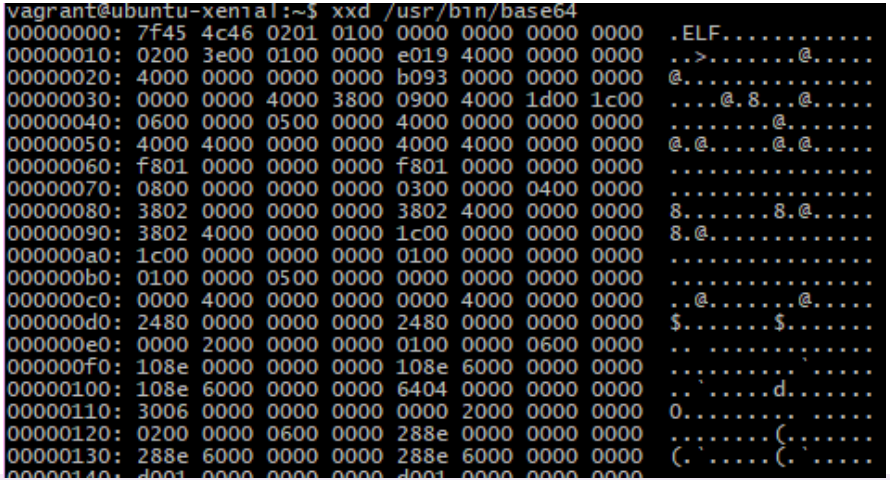
```
CAT(1)                                  User Commands

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...

DESCRIPTION
       Concatenate FILE(s) to standard output.

       With no FILE, or when FILE is -, read standard input.

       -A, --show-all
              equivalent to -vET

       -b, --number-nonblank
              number nonempty output lines, overrides -n

       -e     equivalent to -vE

       -E, --show-ends
              display $ at end of each line

       -n, --number
              number all output lines

       -s, --squeeze-blank
              suppress repeated empty output lines

       -t     equivalent to -vT

       -T, --show-tabs
              display TAB characters as ^I

       -u     (ignored)

       -v, --show-nonprinting
              use ^ and M- notation, except for LFD and TAB

       --help display this help and exit
```

Less /more

- Both are paging systems (they don't print the whole file to terminal, they allow you to do it page by page.
- 'Less is more'
  - Less is a slightly more updated and powerful version of more
- Both have vi like commands available:
  - '/' to search for text

xxd

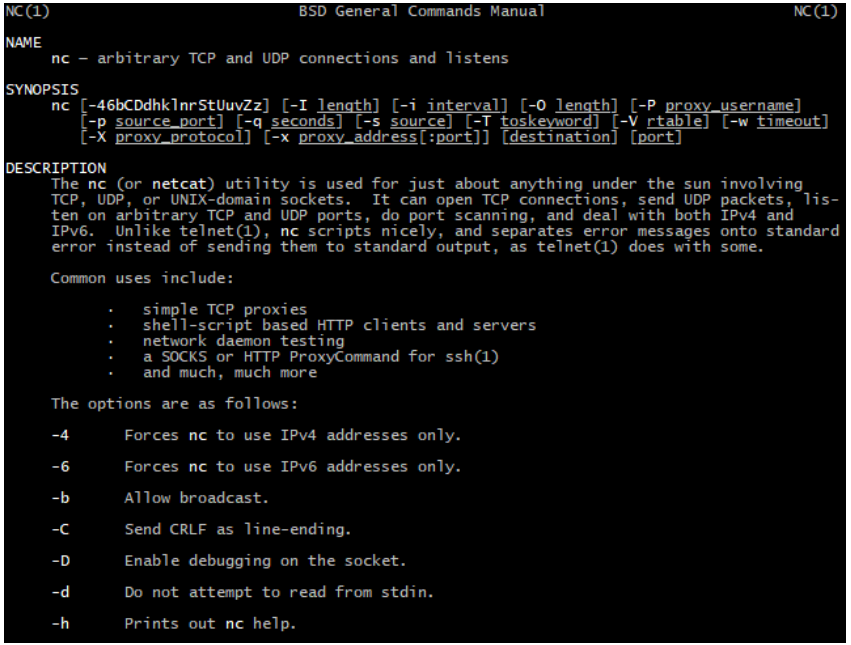
- Perform a hexdump of any file
- Can be useful to see inside binaries and executables
  - You can also use it to manual check for a files magic number or type signature.



Network utils

nc (netcat)

- Allows you to make an arbitrary TCP / UDP connection.
- You can then type in the protocol strings for the higher layer (ie/ HTTP)
  - You can also use terminal redirection to get this from elsewhere
- You can also use it to listen for TCP/UDP connections (`-l`)



```
NC(1)                    BSD General Commands Manual                    NC(1)

NAME
     nc – arbitrary TCP and UDP connections and listens

SYNOPSIS
     nc [-46bCDdhklnrStUuvZz] [-I length] [-i interval] [-O length] [-P proxy_username]
        [-p source_port] [-q seconds] [-s source] [-T toskeyword] [-V rtable] [-w timeout]
        [-X proxy_protocol] [-x proxy_address[:port]] [destination] [port]

DESCRIPTION
     The nc (or netcat) utility is used for just about anything under the sun involving
     TCP, UDP, or UNIX-domain sockets.  It can open TCP connections, send UDP packets, lis-
     ten on arbitrary TCP and UDP ports, do port scanning, and deal with both IPv4 and
     IPv6.  Unlike telnet(1), nc scripts nicely, and separates error messages onto standard
     error instead of sending them to standard output, as telnet(1) does with some.

     Common uses include:

           ·   simple TCP proxies
           ·   shell-script based HTTP clients and servers
           ·   network daemon testing
           ·   a SOCKS or HTTP ProxyCommand for ssh(1)
           ·   and much, much more

     The options are as follows:

     -4      Forces nc to use IPv4 addresses only.

     -6      Forces nc to use IPv6 addresses only.

     -b      Allow broadcast.

     -C      Send CRLF as line-ending.

     -D      Enable debugging on the socket.

     -d      Do not attempt to read from stdin.

     -h      Prints out nc help.
```

ifconfig

- Provides information on current network interfaces present, such as what ips they're bound to, their hardware address, etc

```
IFCONFIG(8)                 Linux Programmer's Manual                 IFCONFIG(8)

NAME
       ifconfig - configure a network interface

SYNOPSIS
       ifconfig [-v] [-a] [-s] [interface]
       ifconfig [-v] interface [aftype] options | address ...

DESCRIPTION
       Ifconfig is used to configure the kernel-resident network interfaces.  It is used at
       boot time to set up interfaces as necessary.  After that, it is usually only  needed
       when debugging or when system tuning is needed.

       If  no  arguments  are  given,  ifconfig displays the status of the currently active
       interfaces.  If a single interface argument is given, it displays the status of  the
       given  interface  only;  if a single -a argument is given, it displays the status of
       all interfaces, even those that are down.  Otherwise, it configures an interface.

Address Families
       If the first argument after the interface name is recognized as the name of  a  sup-
       ported  address  family, that address family is used for decoding and displaying all
       protocol addresses.  Currently supported  address  families  include  inet  (TCP/IP,
       default), inet6 (IPv6), ax25 (AMPR Packet Radio), ddp (Appletalk Phase 2), ipx (Nov-
       ell IPX) and netrom (AMPR Packet radio).
```

```
vagrant@ubuntu-xenial:~$ ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:df:b6:cc:ea
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

enp0s3    Link encap:Ethernet  HWaddr 02:b9:14:72:32:5c
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::b9:14ff:fe72:325c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1139 errors:0 dropped:0 overruns:0 frame:0
          TX packets:776 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:104691 (104.6 KB)  TX bytes:94522 (94.5 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:50:42:4c
          inet addr:192.168.13.37  Bcast:192.168.13.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe50:424c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

telnet

- Opens shell on target
- Insecure by default
  - No authentication system
  - Raw text transmission
- Port 23

ssh

- Opens an encrypted shell connection to a host using SSL
- Can be used for connection tunnelling
- Port 22

Terminal redirection

Piping: '|' (Shift+\)

- Piping allows you to feed the output of one command as the input to another
- Say you have a some executable that performs actions and spews a lot of text to screen
  - But you need to know if, amongst those lines, a certain line exists
  - ./my_verbose_executable | grep "needle"

Terminal redirection

- Output from a command can be redirected elsewhere, to any pipe or file pointer
  - ./this_script > output.txt
  - A single '>' creates/overwrites the file
  - A double '>>' creates/appends to the file
- A terminal consists of 3 channels:
  - 0: STDIN Standard In
  - 1: STDOUT Standard Out
  - 2: STDERR Standard Error
- You can redirect some or all of these:
  - find / -path node_modules 2>/dev/null
  - ./this_script &2>>OutputAndError.txt

Summary

There's a lot more to linux, but you can get by with this

Also look into

- man
- Bash scripting
- xargs
- Linux device management and volume mounting