comm

```
comm <(seq 1 4 | sort) <(seq 2 5 | sort)
```

```
1
        2
        3
        4
    5
```

DD

```
dd
```

```
if=/dev/zero        input file
of=/dev/null        output file
bs=128              batch size in bytes
count=500           cantidad de escrituras
iflag=sync          flag lectura sincrona
oflag=sync          flag
```

dnsmasq

`dnsmasq -d --port=53` normalmente utiliza `/etc/hosts`

servidor dhcp

```
dhcp-option=3,<gateway-ip>                      definir gateway
dhcp-option=6,<dns-ip>                          definir dns
dhcp-range=eth1,10.10.10.1,10.10.10.200         interfaz y rango de ip
dhcp-host=00:00:00:00:00:00,10.10.10.9          reservar ip para mac
```

funcionar como autoridad de zona walala.com

```
--auth-zone=walala.com                  define cual zona
--cname=*.walala.com,walala.com         registro CNAME
--auth-ttl=10                           ttl de la zona de autoridad
--host-record=walala.com,50.55.45.60    registro A
--mx-host=walala.com,50.55.45.60,10     registro MX
```

contestar ip acorde a segmento de red (interfaz de origen)

`--localise-queries`

archivo hosts adicional

`--addn-host=/path/file`

carpeta con multiples archivos hosts

`dnsmasq` realizará `poll` buscando actualizaciones.

`--hosts-dir=/hosts-dir`

log

```
--log-queries
--log-dhcp
```

find

archivos que superen 1 MB

`find -size +1m`

directorios (un solo nivel)

‘find . -mindepth 1 -maxdepth 1 -type d

ejecutar comando 1 comando por archivo

`find -exec du -sh {} \;`

ejecutar comando en un solo comando

`find -exec du -sh {} \+` # IPerf3

`iperf3` del paquete linux `iperf3`

```
-c <ip>     client mode
-s          server mode
-p <num>    puerto
-f K        formato en kilobytes
-R          orden inverso
```

jq

```
jq  '.id = "3" | .id += "6"'
        { "id": "36" }
jq  '.id = [3] | .id += [6]'
        { "id": [3,6] }
```

merge

```
jq '.[0] + .[1]' c.json f.json
echo '{}' | jq '. + { "id":9 }'
```

extraer paths

```
echo '{ "project" : { "id":2 } }' | jq 'paths | map(tostring) | join(".")'
```

kernel panic

```
sysctl -w kernel.panic=3
echo 1 | tee /proc/sys/kernel/sysrq
echo c | tee /proc/sysrq-trigger
```

mkfs

```
mkfs <path>
mkfs
mkfs.bfs
mkfs.cramfs
mkfs.ext2
mkfs.ext3
mkfs.ext4
```

```
mkfs.fat
mkfs.minix
mkfs.msdos
mkfs.ntfs
mkfs.vfat
```

nslookup

```
nslookup walala.com
```

```
nslookup walala.com <ip>
```

parametros

```
-q=mx           consultar registros mx
-port=3000      escuchar en el puerto 3000
```

Pandoc

Dependencias archivo .deb (apt-get install)

- texlive
- texlive-latex-extra
- librsvg2-bin

Transformar markdown a PDF

```
pandoc --lua-filter=diagram.lua --toc \
-s --self-contained -t pdf documento.md -o documento.pdf \
--highlight-style haddock
```

Transformar markdown a REVEALJS

```
pandoc --lua-filter=diagram.lua --toc \
-s --self-contained -t revealjs documento.md -o documento.htm \
--variable revealjs-url=./reveal.js \
--variable theme=solarized \
--highlight-style zenburn
```

ss

`ss -s` del paquete linux `iproute2` # tcpdump

`tcpdump`

```
-i ppp0         definir interfaz
-w p.pcap       almacenar captura en archivo p.cap
-s 0            desactivar limite de paquetes
```

tshark

capturar paquetes

```
tshark -i ppp0 -w captura.pcap
```

reporte http

```
tshark -r captura.pcap -R http -2
```

```
tshark -r captura.pcap -q -z http_req,tree # unzip
```

list files

```
unzip -l <archivo> *.properties
```

dump to stdout

`unzip -p <archivo> *.properties` # xmlstarlet

`xmlstarlet`

argumento	consejo
–break	permite romper anidacion del ultimo selector (-m)
-nl	genera nueva linea
-v ‘text()’	ejecuta una funcion en el nodo actual

dump paths

```
xmlstarlet el pom.xml
```

formatear xml

```
xmlstarlet fo pom.xml | sponge pom.xml
```

seleccionar texto

```
xmlstarlet sel -N m=http://maven.apache.org/POM/4.0.0 -t -m
'//m:project/m:build/m:finalName' -v 'text()' $POM
```

seleccionar texto (count function)

```
xmlstarlet sel -N m=http://maven.apache.org/POM/4.0.0 -t -c
'count(//m:project/m:build/m:finalName)' $POM
```

añadir elemento (inplace)

```
xmlstarlet ed -S --inplace -N m=http://maven.apache.org/POM/4.0.0
-s '//m:project' --type elem -n name -v $1 $POM
```

eliminar elemento (inplace)

```
xmlstarlet ed --inplace -N m=http://maven.apache.org/POM/4.0.0 -d
'//m:dependencyManagement/m:dependencies/m:dependency/m:artifactId[text()="'$DEP'"]/..'
$POM
```

actualizar valor (inplace)

```
xmlstarlet ed -S --inplace -N m=http://maven.apache.org/POM/4.0.0
-u '//m:project/m:build/m:finalName' -v $1 $POM
```

edit options

```
XMLStarlet Toolkit: Edit XML document(s)
Usage: xmlstarlet ed <global-options> {<action>} [ <xml-file-or-uri> ... ]
where
  <global-options>  - global options for editing
  <xml-file-or-uri> - input XML document file name/uri (stdin otherwise)

<global-options> are:
  -P, or -S           - preserve whitespace nodes.
```

```
     (or --pf, --ps)    Note that space between attributes is not preserved
  -O (or --omit-decl) - omit XML declaration (<?xml ...?>)
  -L (or --inplace)   - edit file inplace
  -N <name>=<value>   - predefine namespaces (name without 'xmlns:')
                        ex: xsql=urn:oracle-xsql
                        Multiple -N options are allowed.
                        -N options must be last global options.
  --net               - allow network access
  --help or -h        - display help

where <action>
  -d or --delete <xpath>
  --var <name> <xpath>
  -i or --insert <xpath> -t (--type) elem|text|attr -n <name> [-v (--value) <value>]
  -a or --append <xpath> -t (--type) elem|text|attr -n <name> [-v (--value) <value>]
  -s or --subnode <xpath> -t (--type) elem|text|attr -n <name> [-v (--value) <value>]
  -m or --move <xpath1> <xpath2>
  -r or --rename <xpath1> -v <new-name>
  -u or --update <xpath> -v (--value) <value>
                          -x (--expr) <xpath>

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see http://xmlstar.sourceforge.net/)
```

sel options

```
XMLStarlet Toolkit: Select from XML document(s)
Usage: xmlstarlet sel <global-options> {<template>} [ <xml-file> ... ]
where
  <global-options> - global options for selecting
  <xml-file> - input XML document file name/uri (stdin is used if missing)
  <template> - template for querying XML document with following syntax:

<global-options> are:
  -Q or --quiet             - do not write anything to standard output.
  -C or --comp              - display generated XSLT
  -R or --root              - print root element <xsl-select>
  -T or --text              - output is text (default is XML)
  -I or --indent            - indent output
  -D or --xml-decl          - do not omit xml declaration line
  -B or --noblanks          - remove insignificant spaces from XML tree
  -E or --encode <encoding> - output in the given encoding (utf-8, unicode...)
  -N <name>=<value>         - predefine namespaces (name without 'xmlns:')
                              ex: xsql=urn:oracle-xsql
                              Multiple -N options are allowed.
```

```
  --net                     - allow fetch DTDs or entities over network
  --help                    - display help

Syntax for templates: -t|--template <options>
where <options>
  -c or --copy-of <xpath>   - print copy of XPATH expression
  -v or --value-of <xpath>  - print value of XPATH expression
  -o or --output <string>   - output string literal
  -n or --nl                - print new line
  -f or --inp-name          - print input file name (or URL)
  -m or --match <xpath>     - match XPATH expression
  --var <name> <value> --break or
  --var <name>=<value>      - declare a variable (referenced by $name)
  -i or --if <test-xpath>   - check condition <xsl:if test="test-xpath">
  --elif <test-xpath>       - check condition if previous conditions failed
  --else                    - check if previous conditions failed
  -e or --elem <name>       - print out element <xsl:element name="name">
  -a or --attr <name>       - add attribute <xsl:attribute name="name">
  -b or --break             - break nesting
  -s or --sort op xpath     - sort in order (used after -m) where
  op is X:Y:Z,
      X is A - for order="ascending"
      X is D - for order="descending"
      Y is N - for data-type="numeric"
      Y is T - for data-type="text"
      Z is U - for case-order="upper-first"
      Z is L - for case-order="lower-first"

There can be multiple --match, --copy-of, --value-of, etc options
in a single template. The effect of applying command line templates
can be illustrated with the following XSLT analogue

xmlstarlet sel -t -c "xpath0" -m "xpath1" -m "xpath2" -v "xpath3" \
        -t -m "xpath4" -c "xpath5"

is equivalent to applying the following XSLT

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <xsl:call-template name="t1"/>
  <xsl:call-template name="t2"/>
</xsl:template>
<xsl:template name="t1">
  <xsl:copy-of select="xpath0"/>
  <xsl:for-each select="xpath1">
```

```
    <xsl:for-each select="xpath2">
      <xsl:value-of select="xpath3"/>
    </xsl:for-each>
  </xsl:for-each>
</xsl:template>
<xsl:template name="t2">
  <xsl:for-each select="xpath4">
    <xsl:copy-of select="xpath5"/>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see http://xmlstar.sourceforge.net/)

Current implementation uses libxslt from GNOME codebase as XSLT processor
(see http://xmlsoft.org/ for more details)
```