comandos bash

linux, bash

comandos al detalle

- awk
- alias
- curl
- dd
- find
- free
- grep
- less
- locale
- lsof
- mount
- rclone
- rsync
- sed
- sudo
- sshd
- tar
- vi(m)
- xargs
- linux comandos dispositivos de bloque
- xev
- desactivar touchpad mientras escribes

ls

caracteres ocultos

```
printf "[%s]\n" *
printf "%s\n" * | cat -vte
printf "%s\n" * | od -bc
```

/vía: https://stackoverflow.com/questions/19604384/unixtwo-directories-with-same-name-on-the-same-path

exclusión

excluir ficheros del listado (y otras cosas??):
http://askubuntu.com/questions/333640/cp-command-to-exclude-certain-files-from-being-copied

```
$ ls !(unlisted1|unlisted2)
$ ls !(unlisted*)
```

```
$ export GLOBIGNORE='unlisted1:unlisted2'
$ ls *
```

```
$ export GLOBIGNORE='unlisted*'
$ ls *
```

> **Glob**
> https://en.m.wikipedia.org/wiki/Glob_(programming)

filesystem

- `mkdir -p /jail/{dev,bin,sbin,etc,usr,lib,lib64}`: crea estructura directorios en **jail**
- `echo /usr/dir1 /var/dir2 /nas/dir3 | xargs -n 1 cp -v /path/to/file`: copia **file** en **dir1,dir2,dir3**
- `diff /tmp/r/ /tmp/s/`: diferencias entre 2 directorios
- du → disk used
  - -c : totales
  - -h : human readable
  - -s : solo totales
  - –max-depth=x : cuanto de profundo baja en el árbol para mostrar información detallada
  - –one-file-system : o -x, muestra información solo del mismo filesystem (evita otros filesystems en el arbol)
  - –exclude '*.obj'
  - listar los 3 directorios con más utilización: `$ du -sk * | sort -nr | head -3`
- `df`
  - -h : human readable
- `chown`:
  - cambiar propietario de un fichero: `$ chown root tmpfile`
  - cambiar grupo de un fichero: `$ chown :friends tmpfile`
  - cambiar propietario y grupo de un fichero: `$ chown himanshu:friends tmpfile`
  - cambiar fichero de origen de un link simbólico: igual que cualquier fichero
  - cambiar propietario/grupo del link simbólico en sí: `$ chown -h root:friends tmpfile_symlnk`
  - cambiar el propietario de un fichero que tenga un propietario concreto: `$ chown –from=root himanshu tmpfile` ← cambia tmpfile a himanshu si el propietario es root
  - cambiar el grupo de un fichero que tenga un propietario concreto: `$ chown –from=:friends :family tmpfile` ← cambia tmpfile a family si el grupo es friends
  - copiar el propietario/grupo de un fichero a otro: `$ chown –reference=file tmpfile`
  - cambiar propietario/grupo recursivamente en directorios: `$ chown -R himanshu:family linux/`
  - cambiar propietario/grupo recursivamente de un enlace simbólico a directorio: `$ chown -R -H guest:family linux_symlnk`
  - mostrar los cambios realizados por el comando: `$ chown -v <resto de comando>`
- dd → comando para flujos de datos
  - `sudo dd if=/dev/disk2s1 of=/Users/admin/imagen.iso` → graba disco a imagen ISO
  - `sudo dd bs=4M if=image.iso of=/dev/sdxx` → graba una imagen ISO a disco (USB)
  - `sudo dd status=progress if=… of=…` → muestra estadísticas de transferencia (solo GNU Coreutils 8.24+)
  - `sudo dd if=… | pv ←s size> | sudo dd of=…` → usar utilidad PV para progreso (y si le pasas el «size», hace cálculo de restante)
  - http://askubuntu.com/questions/215505/how-do-you-monitor-the-progress-of-dd
- `cpio` : como dd pero copiando solo la zona donde hay datos
- `split -b<tamaño> <fichero> <output>.@`
  - corta un *fichero* en trozos de *tamaño*

    - tamaño pueden ser bytes (b), Kilobytes (k), Megabytes (m), gigabytes(g)[1)]
        - `split -b100m <fichero> <output>.@`: genera ficheros de 100Mg
    - genera ficheros con el nombre <output>.@aa, <output>.@ab, <output>.@ac... hasta acabar
    - se puede reconstruir con un `cat` (o entiendo que un `type` en MSDOS)
        - `cat <output>.* <fichero>`
    - http://quechilero.com/blog/2009/10/26/cortar-archivos-grandes-en-linux/
- `rename` : renombrar ficheros con REG_EXP
    - `-n` : no-action (modo test)
- `rm !(*.tgz|*.zip)` : elimina todos los ficheros que no sean .tgz o .zip
    - se puede aplicar como filtro en cualquier otro comando (ls, por ejemplo)
    - `shopt -s extglob` para que funcione, si no intantará expandir el `!`

preservar ficheros

- `cp <FICHERO>{,.bak}`
    - crea un fichero idéntico llamado **<FICHERO>.bak**
    - ideal para hacer un backup previo a editar.
- `mv {,prefix_}<FICHERO>`
    - pone prefijo a un fichero (o grupo de ellos)
- `cp -r .ssh{,_backup}`
    - crea una copia de la carpeta **.ssh** como **.ssh_backup**
- `cp -r .ssh{,_`date +«%Y.%-m.%-d»`}`
    - crea una copia de la carpeta como **.ssh_2018.4.28**
- `install -b /path1/fichero1 /path2/`
    - copia fichero1 en path2, si existe, renombra el fichero de destino como **fichero1~**

simples

- `cat`
    - -T : muestra tabuladores.
    - -E : muestra finales de línea.
    - -v : muestra ^ y M-
    - -A : muestra todos los caracteres no imprimibles
    - -n o –number : añade número de línea.
    - más ejemplos: https://www.cyberciti.biz/faq/linux-unix-appleosx-bsd-cat-command-examples/
- `uptime` : tiempo que lleva la máquina levantada
- `tail` → muestra 10 últimas líneas
    - -n → número de líneas a mostrar
- head → muestra 10 primeras líneas
    - -n → número de líneas a mostrar
- `sort`
- `xargs`
- `cut`
- `watch` : ejecuta un segundo comando cada n segundos
    - `watch -n 5 ls -la`
- wc → word count
    - -c : cuenta caracteres
    - -l : cuenta líneas
    - -w : cuenta palabras
- `tr` → translate characters (sustituye cadenas sobre la entrada standard, uso como filtro en scripts)
    - :lower
    - :upper
- `expr <operación aritmética>`
    - importante separación entre números y operación

    - expr 1 + 1 → devuelve 2
    - expr 2 * 2
    - valor=`expr 1 + 1`
    - valor=$((1 * 1)) ← no importa espacio separación entre números, los paréntesis están separados por el WIKI, no han de ir así.
- `bc` → calculadora
    - `$ echo «scale=300;4*a(1)» | bc -l` → 300 cifras de PI
    - http://www.basicallytech.com/blog/index.php?/archives/23-command-line-calculations-using-bc.html
- `nl` → numbering line
- `cat`
    - -n : muestra líneas
    - -A : elimina caracteres de control
- `file` → intenta identifiar el formato del fichero (o programa del que procede). Falsos «positivos»
- `time (comando)` → para calcular cuanto tarda en ejecutarse el comando

du

http://www.manpagez.com/man/1/du/

- `du -csh <carpeta>`
    - c → display total
    - s → no muestra resumen por directorios, solo total
    - h → human readable

comunicaciones

- `nc` → netcat
- `dig` → estilo `nslookup`, se le da el nombre del dominio y ofrece información en formato BIND
    - -x : inverso, te da la IP
- `ping` como `traceroute`:

```
for i in {1..30}; do ping -t $i -c 1 google.com; done | grep "Time to live exceeded"
```

miscelaneos

- `hdiutil` → comando con verbos, relacionado con CD/DVD + ISO
    - mount : para montar una imagen ISO

ejemplos

varios

- borrar ficheros excepto los que cumplan patrón:

```
for a in `ls | grep -v PATRON`; do rm -fr $a; done
```

- desmontar todas los volúmenes montados que tengan la cadena «home_ERE»:

```
for i in $(mount | cut -d" " -f1 | grep /home_ERE/); do umount $i; done
```

- Borra todos los ficheros que NO sean los indicados:

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

- Borra los ficheros de una lista TXT:

```
while read file; do rm "$file"; done < fichero_entrada
```

- Borra los ficheros de una lista TXT:

```
rm -f $(<fichero_entrada)
```

- Copia los permisos de un fichero (file1) en otro (file2):

```
chmod --reference file1 file2
```

- Crear un a serie de directorios anidados:

```
mkdir -p directorios/a/crear
```

- Calcular espacio de un directorio:

```
du -sh <directorio>
```

- ejecutar comando sobre ficheros de directorio sin scripts:

```
for f in * ; do comando; done;
```

- ignorar cierto tipos de archivos con '*':

```
export GLOBIGNORE='<match_ficheros_a_ignorar>'
ls * <- no mostrará los ficheros que cumplan el criterio, también funciona con
cp o mv (y puede que otros, mientras dependan del '*'
```

- mostrar diferencias 2 archivos (que se ordenan): `comm -12 <(sort -u File1) <(sort -u File2)`

tiempo

- sincroniza la fecha entre 2 servidores, en detrimento de NTP:

```
$ date --set="$(ssh user@server date)"
```

- ejecuta <comando> y si en el tiempo <tiempo> no ha finalizado, hace un kill (de hecho, se le puede enviar cualquier señal):

```
$ timeout <tiempo> <comando>
```

- Ejecutar comando diferido, sin consola abierta ¿?:

```
((sleep 2h;comando argumentos)$)
```

- repite un comando cada intervalo de tiempo:

```
watch -n <segundos> <comando>
```

    - existen alternativas para repetir indefinidamente comandos desde CLI:

- `while [ 1 ]; do foo; sleep 2; done`
- `while true; do foo; sleep 2; done`
- `for ((;;)); do foo; sleep 2; done`
- `until [ ]; do foo; sleep 2; done`

[1)]
a mi esté no me funcionó