- ºNo poner respuesta de ejercicios.

SHELL

* pwd: saber en que directorio estás
* mkdir nombre\_directorio: crear directorios.
* ls : mostrar listados
  + ls -l: listado largo del directorio actual
  + ls -U: ordena por creación
  + ls -t: ordena por la última modificación
  + ls -a: listar e incluye los elementos ocultos
* cd: moverte por los directorios
* rm: borrar archivos
* rmdir: borrar directorios
  + r [-f]: eliminar carpetas con contenido (conificaciones de fecha/hora -f no solicita confirmación)
  + Touch: crear archivo/ mod
  + touch *nombre del fichero*: creas fichero
  + touch -t aaaammddhhmm *nombre del fichero*: modificar fecha
  + touch -ht aaaammddhhmm *nombre del fichero*: modificar fecha directorio
* cat: leer fichero
* cat >> editar fichero
* ln: linkar (vincular) ficheros/directorios
  + ln *fichero1 fichero2* asi creas un vinculo “duro” entre dos o más ficheros (no pueden ser directorios). Si modificas uno (con touch -t) se modifica el otro.
  + ln -s *fichero1/directorio fichero2* así creas un vínculo simbólico (como un acceso directo). Puedes modificar el *fichero1/directorio* (con touch -t) pero no se cambiará el *fichero2*. Para modificar el *fichero2*, hay que utilizar touch -ht
* ln test3 test5 haces copia
* find: <https://www.hostinger.es/tutoriales/como-usar-comando-find-locate-en-linux/>
  + -name
* magicfile tuto: <https://openpreservation.org/blog/2012/08/09/magic-editing-and-creation-primer/>
* mv: mover archivos o directorios
  + <https://ss64.com/osx/mv.html>
* cp: copiar archivos
  + <https://ss64.com/osx/cp.html>
* LINK- Pones tu linea y te explica la función de cada cosa.
  + <https://explainshell.com/>
* tr: es una utilidad de línea de comandos para traducir o eliminar caracteres
  + -d: elimina los caracteres del primer conjunto de la salida.
* tar
  + tar -cf *nombre*.tar *nombre del fichero* : crear archivo tar de un fichero.
  + tar -cf *nombre*.tar \* : crear archivo tar con todos los ficheros de la carpeta
  + tar -xvf *archivo.tar*: descomprimir archivo
* sed: editor de flujo para filtrar y transformar texto
* ctrl + a : te envia al principio de la linea que estes editando
* kinit + espacio + login: obtiene ticket kerberos de tu usuario

GIT

* git init: transforma nuestro directorio en un archivo git
* git status: para saber por dónde estamos
* git add: prepara el proyecto para hacer un commit
* git rm --cached archivo: elimina lo que hemos creado en add
* git commit:
* git log: muestra una lista de los commits hechos en el repositorio (en orden cronológico inverso si no se añaden argumentos).
* cd¿Cómo eliminar un repositorio de Git creado con ‘git init’ en un directorio?

cd carpeta/

rm -rf .git

* subir al repo:
  + git init(en la carpeta que queremos que sea nuestro repo): iniciar repo
  + git add . : Añadir archivo
  + git commit -m “descripción del commit): identifica los cambios hechos con tu repo antes de subirlo definitivamente
  + git config --global user.email escribe\_tu\_mail: Indicar tu mail
  + git config --global user.name “tu\_nombre”(Hay que poner el nombre entre comillas): Indicar tu nombre
  + git config --global --edit: comprobar que has metido bien la información anterior o modificarlo.
  + git remote add origin *direccion\_de\_tu\_repo:* añadir la url del repo
  + git remote set-url origin direccion\_de\_tu\_repo: cuando ya tenemos repo para modificarlo usamos este
  + git config --get remote.origin.url: Comprueba la url del repositorio
  + git push origin master: Sube archivos al repo
* git clone url\_repo: clonar el repositorio
* git log: Para ver la información de los commits
* git ls-tree -r master --name-only: muestra los archivos subidos en el repositorio remoto.
* git rev-list origin/master: Lista los ID de los commit recientes.

C

* Para crear la cabecera (school header) abrimos con vim el archivo.c y pulsamos fn+f1
* Función Main: que es el punto de entrada

int main()

{}

* gcc archivo.c: Compila el programa
* return 0;: se utiliza para que te devuelva 0 e indicar que todo es ok.
* ejecutar norminette: norminette -R CheckForbiddenSourceHeader *nombre\_archivo*

**Atajos**

gg=(para escribir el = shift+0)G: alinear

ctrl+n: para autocompletar.

**Estructura de una función:**

v

tipo\_de\_retorno nombre\_de\_la\_función (lista\_de\_parámetros)

{

cuerpo\_de\_la\_función

return expresión

}

**Donde:**

• tipo\_de\_retorno: es el tipo del valor devuelto por la función, o, en caso de que la función no devuelva valor alguno, la palabra reservada void.

• nombre\_de\_la\_función: es el nombre o identificador asignado a la función.

• lista\_de\_parámetros: es la lista de declaración de los parámetros que son pasados a la función. Éstos se separan por comas. Debemos tener en cuenta que pueden existir funciones que no utilicen parámetros.

• cuerpo\_de\_la\_función: está compuesto por un conjunto de sentencias que llevan a cabo la tarea específica para la cual ha sido creada la función.

• return expresión: mediante la palabra reservada return, se devuelve el valor de la función, en este caso representado por expresión.

**Arrays**

Pequeña introducción al maravilloso mundo de los arrays <http://www.uco.es/grupos/eatco/informatica/metodologia/cadenasyarrays.pdf>

**Manual de programación en lenguaje C**:

http://isa.uniovi.es/~ialvarez/Curso/descargas/lenguaje\_c.pdf

:)

:((

PUNTEROS

El puntero es el que apunta al espacio donde está la variable, así de fácil :)