

## Glosario de comandos para utilizar en la consola

Los comandos explicados a continuación son comandos de Unix. Funcionan por defecto en un terminal de comandos en cualquier distribución Linux y por eso se recomienda hacer las prácticas en un equipo con Linux. De todos modos, las alumnas y alumnos que utilicéis habitualmente Windows en el aula podéis emplear el terminal de Cygwin, que emula parte del funcionamiento de Unix. Dado que está instalado en todos los equipos del aula de prácticas, sólo tenéis que buscar la aplicación Cygwin en el escritorio o en el menú de aplicaciones<sup>1</sup>. El icono es este:



Comando <sup>2</sup>	Descripción y ejemplos																		
\$ pwd	Devuelve la ruta del directorio actual.																		
\$ ls \$ ls -l	Presenta un listado de los archivos y carpetas en el directorio actual. Para obtener un listado más detallado, con información de los permisos de cada ítem, el usuario propietario, el tamaño en bytes y la fecha de modificación, podéis utilizarlo con la opción -l, es decir, tendríais que teclear ls -l.																		
\$ mkdir	Crea un directorio dentro do directorio actual. Ejemplo: \$ mkdir Documentos																		
\$ cd	Cambia el directorio de trabajo <sup>1</sup> . Este comando recibe como parámetro la ruta del directorio al que queremos movernos, que podrá ser absoluta o relativa. Recordad que las rutas absolutas comienzan con el carácter "/". También se permite el uso de dos parámetros especiales, como el carácter ".", que es sustituido por la ruta del directorio actual, y el carácter "..", que es sustituido por la ruta del directorio padre (directorio que contiene al directorio actual). Cuando las rutas de acceso son largas, es muy útil el uso del TABULADOR, que autocompleta la ruta y, en caso de haber varias posibilidades, ofrece un listado de las mismas. Algunos ejemplos:																		
	<table><tr><th>Directorio de origen</th><th>Comando</th><th>Directorio de destino</th></tr><tr><td>/home/usuario</td><td>cd Documentos cd ./Documentos</td><td>/home/usuario/Documentos</td></tr><tr><td>/home/usuario</td><td>cd /Documentos</td><td>/Documentos</td></tr><tr><td>/home/usuario</td><td>cd ..</td><td>/home</td></tr><tr><td>/home/usuario</td><td>cd ../usuario2</td><td>/home/usuario2</td></tr><tr><td>Cualquiera</td><td>cd</td><td>Vuelve a la carpeta del usuario /home/usuario</td></tr></table>	Directorio de origen	Comando	Directorio de destino	/home/usuario	cd Documentos cd ./Documentos	/home/usuario/Documentos	/home/usuario	cd /Documentos	/Documentos	/home/usuario	cd ..	/home	/home/usuario	cd ../usuario2	/home/usuario2	Cualquiera	cd	Vuelve a la carpeta del usuario /home/usuario
	Directorio de origen	Comando	Directorio de destino																
	/home/usuario	cd Documentos cd ./Documentos	/home/usuario/Documentos																
	/home/usuario	cd /Documentos	/Documentos																
	/home/usuario	cd ..	/home																
	/home/usuario	cd ../usuario2	/home/usuario2																
Cualquiera	cd	Vuelve a la carpeta del usuario /home/usuario																	
\$ mv	Mueve un archivo o carpeta a otra localización. También se puede utilizar para cambiar el nombre de un archivo o carpeta. La sintaxis es la siguiente: \$ mv ORIGEN DESTINO. Algunos ejemplos:																		
	<table><tr><th>Comando</th><th>Orden</th></tr><tr><td>mv enunciado.odt ..</td><td>Mueve el archivo <i>enunciado.odt</i> a la carpeta padre</td></tr><tr><td>mv practica practica2</td><td>Cambia el nombre del archivo o carpeta <i>practica</i> a <i>practica2</i></td></tr><tr><td>mv Makefile.txt Makefile</td><td>Cambia el nombre del archivo <i>Makefile.txt</i> a <i>Makefile</i>. Esta orden será útil cuando creéis el Makefile desde el Bloc de notas de Windows, que</td></tr></table>	Comando	Orden	mv enunciado.odt ..	Mueve el archivo <i>enunciado.odt</i> a la carpeta padre	mv practica practica2	Cambia el nombre del archivo o carpeta <i>practica</i> a <i>practica2</i>	mv Makefile.txt Makefile	Cambia el nombre del archivo <i>Makefile.txt</i> a <i>Makefile</i> . Esta orden será útil cuando creéis el Makefile desde el Bloc de notas de Windows, que										
	Comando	Orden																	
	mv enunciado.odt ..	Mueve el archivo <i>enunciado.odt</i> a la carpeta padre																	
mv practica practica2	Cambia el nombre del archivo o carpeta <i>practica</i> a <i>practica2</i>																		
mv Makefile.txt Makefile	Cambia el nombre del archivo <i>Makefile.txt</i> a <i>Makefile</i> . Esta orden será útil cuando creéis el Makefile desde el Bloc de notas de Windows, que																		

<sup>1</sup> Vuestra carpeta de trabajo al lanzar la consola de cygwin es **C:\cygwin\home\nome\_usuario** pero vuestra carpeta de documentos de Windows es: **C:/Users/nome\_usuario/Documents**. Si queréis acceder a un pendrive o a otra unidad, podéis ir a la carpeta **cygdrive** escribiendo: **cd /cygdrive** y lanzar el comando **ls**. En esa carpeta aparecen los nombres de las unidades actualmente activas. Si estas unidades son **c** y **e**, podéis acceder desde /cygdrive con los comandos **cd c** o **cd e** o directamente desde vuestro home escribiendo **cd c:** o **cd e:**.

<sup>2</sup> El carácter **\$** que aparece antes de cada comando no hay que teclearlo, es simplemente una convención utilizada para indicar que se trata de un comando de consola.

Comando <sup>2</sup>	Descripción y ejemplos		
		por defecto añade la extensión ".txt" al nombre do archivo.	
\$ more	Visualiza el contenido de un archivo, cuya ruta se pasa como parámetro		
\$ rm \$ rm -r	Borra un archivo o directorio cuya ruta se pasa como parámetro. Hay que tener en cuenta que lo que se borra no va a la Papelera de reciclaje, por lo que no se puede recuperar. Si se quiere borrar un directorio, habrá que utilizar a opción -r. Algunos ejemplos:		
	Comando	Orden	
	rm Makefile	Borra el archivo con el nombre Makefile	
	rm *.o	Borra todos los archivos cuyo nombre termine en .o	
	rm -r practica	Borra el directorio practica y todo su contenido	
\$ find	Busca un archivo (o un directorio) dentro de un directorio. Este comando cuenta con numerosas opciones, en este caso veremos un uso sencillo con la siguiente sintaxis: \$ find DIRECTORIO [-type TIPO] -iname EXPRESION La opción -iname, a diferencia de la opción -name, permite que la búsqueda no sea sensible a mayúsculas y minúsculas. Es decir, en el primer ejemplo el comando encontrará tanto el archivo "trabajo.odt", como los archivos "Trabajo.odt", "TRAbajo.odt" o "TrAbAJo.ODT".		
	Directorio origen	Comando	Orden
	/home/usuario	find . -iname "trabajo.odt"	Busca todos los archivos con el nombre trabajo.odt que haya dentro de /home/usuario y sus subdirectorios.
	/home/usuario	find .. -iname "*.c"	Busca todos los archivos con la extensión .c que haya dentro de /home y sus subdirectorios
	/home/usuario	find Documentos -iname ".o"	Busca todos los archivos con la extensión .o que haya dentro de /home/usuario/Documentos y sus subdirectorios.
	/home/usuario	find . -iname "practica"	Busca todos los <u>archivos y directorios</u> con el nombre practica que haya dentro de /home/usuario y sus subdirectorios.
	/home/usuario	find . -type d -iname "practica"	Busca todos los <u>directorios</u> con el nombre practica que haya dentro de /home/usuario y sus subdirectorios.
	/home/usuario	find . -type f -iname "practica"	Busca todos los <u>archivos</u> con el nombre practica que haya dentro de /home/usuario y sus subdirectorios.

### Ejemplo para crear un Makefile y ejecutarlo.

Crea desde la consola de comandos la carpeta "ejemplo" (`$ mkdir ejemplo`). Haz que esa carpeta sea tu carpeta actual (`$ cd ejemplo`). Crea los siguientes archivos dentro de esta carpeta (con el bloc de notas o cualquier otro editor): "programa.c" (que contiene el main), "operaciones.c" y "operaciones.h" (que contienen diversas funciones matemáticas). El contenido de los archivos es el siguiente:

```
//programa.c

#include <stdio.h>
#include "operaciones.h"

int main(){
    printf("El resultado de sumar 3 y 4 es igual a %d\n", suma(3,4));
}
```

```
//operaciones.h

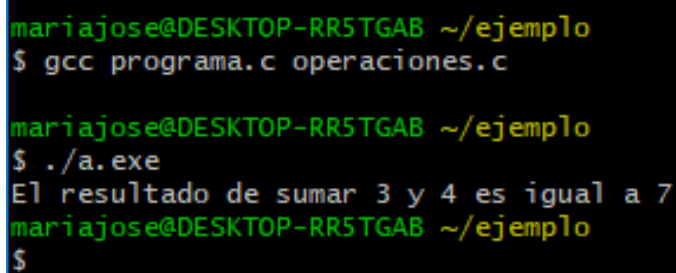
int suma(int n1, int n2);
int resta(int n1, int n2);
```

```
//operaciones.c

int suma(int n1, int n2){
    return n1+n2;
}

int resta(int n1, int n2){
    return n1-n2;
}
```

Este proyecto puede compilarse con la herramienta `gcc`. Esto generará un ejecutable, que suele llamarse por defecto `a.exe` en Windows y `a.out` en Linux, y que podrá ser ejecutado desde el propio terminal.



```
mariajose@DESKTOP-RR5TGAB ~/ejemplo
$ gcc programa.c operaciones.c

mariajose@DESKTOP-RR5TGAB ~/ejemplo
$ ./a.exe
El resultado de sumar 3 y 4 es igual a 7
mariajose@DESKTOP-RR5TGAB ~/ejemplo
$
```

No obstante, para proyectos más complejos, el comando `gcc` puede complicarse y tiene sentido utilizar herramientas para automatizar este proceso. Una de las más empleadas es `Make`. Existen multitud de recursos para aprender a utilizar esta herramienta, y la tenéis explicada en detalle en el guion de la práctica 0, aunque escribiremos un fichero `Makefile` para compilar el proyecto actual. En este documento dejamos a mayores un enlace a una lista de reproducción de Youtube en la que se explican los conceptos básicos: [Tutorial de Make en Youtube](#)

En esencia, la configuración de la compilación se realiza mediante un único archivo, con el nombre "Makefile" (no lleva extensiones). Un ejemplo para nuestro proyecto podría ser:

```
#opciones de compilación (Warnings all)
CC=gcc -Wall

#carpeta de las cabeceras (si están en la actual, ponemos .)
HEADER_FILES_DIR=.
#opciones de compilación, indica dónde están los archivos .h
INCLUDES=-I $(HEADER_FILES_DIR)

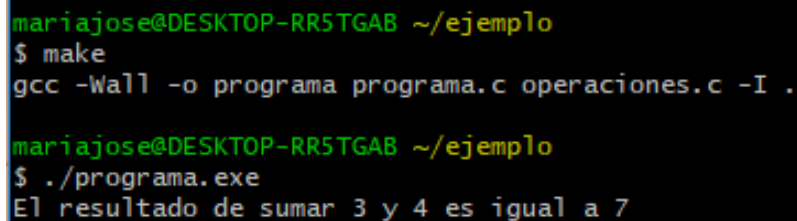
#nombre del ejecutable o archivo de salida (programa.exe)
OUTPUT= programa

#fuentes
SOURCES=programa.c operaciones.c

#ficheros .h
LIB_HEADERS=$(HEADER_FILES_DIR)/operaciones.h

#regla de ejecutable, prerequisites $(SOURCES) y $(LIB_HEADERS)
#IMPORTANTE: la regla debe ir tabulada a la derecha
$(OUTPUT): $(SOURCES) $(LIB_HEADERS)
    $(CC) -o $(OUTPUT) $(SOURCES) $(INCLUDES)
```

Con esta configuración estamos definiendo varias variables para evitar repetir código, y una única regla para compilar. La ejecución del Makefile se realiza mediante el comando *make*.



```
mariajose@DESKTOP-RR5TGAB ~/ejemplo
$ make
gcc -Wall -o programa programa.c operaciones.c -I .

mariajose@DESKTOP-RR5TGAB ~/ejemplo
$ ./programa.exe
El resultado de sumar 3 y 4 es igual a 7
```