Práctica 0. Introducción a herramientas y librerías básicas de I/O en C

Laboratorio de Arquitectura de Redes de Comunicación

Objetivos de la práctica

- En esta práctica se presentarán y recordarán conceptos básicos acerca del trabajo con el lenguaje C, presentando las herramientas de trabajo más comunes.
- En concreto se estudiarán:
  - Órdenes de compilación con **gcc**
  - Compilación de proyectos con la herramienta **make**
  - Uso de la documentación y la ayuda con la herramienta **man**
  - Entrada/Sálida básica en C. Familia de funciones printf/scanf

Configuración de la cuenta de usuario (I)

- Antes de comenzar la práctica, deberá proceder con la configuración de su cuenta de usuario.
- El profesor de prácticas les indicará su login, que será de la forma "larcXX", siendo XX el número de grupo. La password por defecto es "larc2011".

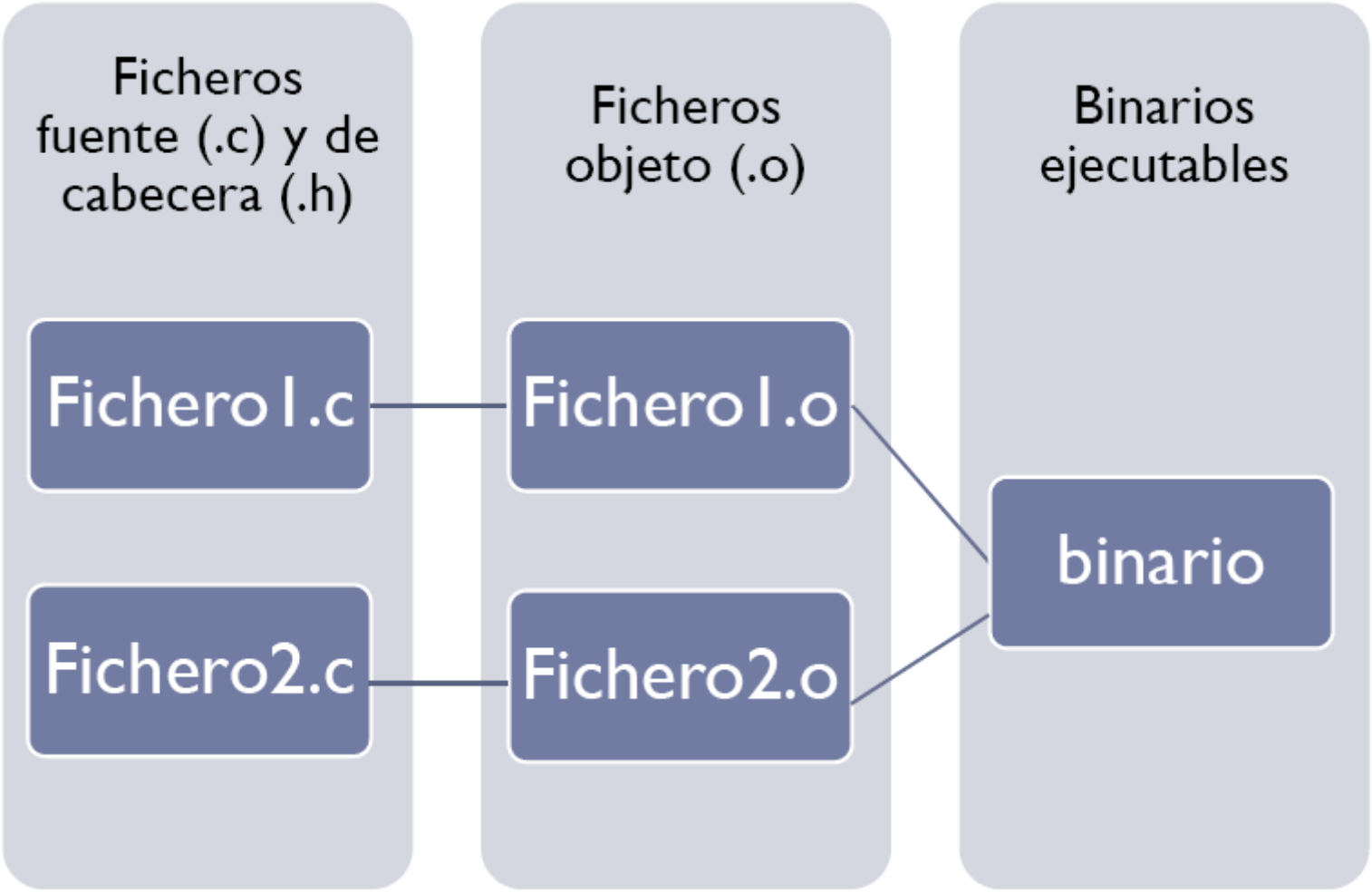
- Tras ello, el primer paso es cambiar la contraseña para la cuenta:

```
> ssh labit601.upct.es    (entre con el login y la contraseña indicados)
> yppasswd                (cambia su contraseña para el dominio NIS del laboratorio)
```

Configuración de la cuenta de usuario (II)

- Una vez establecida la nueva contraseña, deberá crear en el raíz de su cuenta de usuario un directorio llamado "entregas", en este directorio deberá depositar a lo largo del curso los ejercicios.
- Cree dentro del directorio "entregas" un fichero llamado GRUPO. En este fichero deberá indicar el grupo de practicas (LARCXX), el turno (p. ej. Jueves 9-12h) y los nombres de los componentes del grupo.
- Para cada práctica deberá crear dentro de "entregas" un directorio "practicaX", donde X indica el número de la práctica. Esta semana, deberá crear el directorio llamado "practica0".
- **SÓLO DEBE COPIAR A ESTOS DIRECTORIOS LOS FICHEROS INDICADOS EN LOS ENUNCIADOS.**

Compilación con **gcc** (I)



Compilación con **gcc** (II)

- Compilación de los ficheros fuente
  - gcc -c fichero.c (genera el fichero.o correspondiente)
- Linkado de los ficheros objeto
  - gcc -o binario fichero1.o fichero2.o […] (genera el binario)

Compilación con **gcc** (III)

- Opciones más comunes
  - -Wall (activa avisos de problemas en el código, **debe usarse siempre)**
  - -g (activa depuración, sólo se usa si se depurará el programa con el gdb, **no la usaremos**)
  - -OX (activa optimización de código, **nosotros usaremos –O2)**
  - -static (compila el código de modo estático, **no la usaremos)**
- Orden de compilación típica:
  - gcc -Wall -O2 -c fichero1.c

Ejercicio 1

EJERCICIO: Escriba un programa (ejercicio1.c) que obtenga como primer parámetro de entrada el nombre de un fichero. Este fichero tendrá en cada línea un número natural NN. El programa debe mostrar la suma de las líneas de los números del fichero.

¿CÓMO DEPURAR EL CÓDIGO?

Lo haremos volcando información durante la ejecución del programa con órdenes printf.

Ejemplo: printf("El programa ha pasado por este punto\n");

Nota: es obligatorio incluir el salto de línea ("\n") para que se vuelque inmediatamente el mensaje.

Compilación con **make** (I)

- Si existen varios ficheros de código es engorroso compilar manualmente.
- Recompilar siempre todos los ficheros puede ser computacionalmente costoso
- Compilación vía **make**
  - Automatiza el proceso de compilación
  - Sólo compila aquellos archivos de objeto que no han variado

Compilación con **make** (II)

- Para compilar con **make** es necesario crear un fichero llamado Makefile donde se indicarán las dependencias de compilación:

```
# Archivo Makefile, las líneas que comienzan por # son comentarios
# Definiciones globales
CC = gcc
CFLAGS = -Wall -O2

# Reglas, tienen el formato objetivo: dependencia1 … dependenciaN
# Si alguna de las dependencias ha variado se recompila el objetivo
binario: archivo1.o … archivoN.o

archivo1.o: archivo1.c archivo1.h

archivoN.o: archivoN.c archivoN.h

clean:
        rm -f *.o binario

# Hay que dejar siempre una línea en blanco tras cada regla o indicar un
comando tras un TABULADOR
```

Ejercicio 2

EJERCICIO: Cree un Makefile adecuado para la compilación del ejercicio 1. El Makefile debe contener una orden (clean) para borrar los objetos y el binario y una orden (tar) que invoque primero a clean ydespués comprima los ficheros fuente y el propio Makefile a un fichero llamado ejercicio1.tgz.

¿CÓMO COMPRIMIR FICHEROS?

tar fcvz fichero.tgz directorio1 […] fichero1 […]

¿CÓMO DESCOMPRIMIR FICHEROS?

tar fxvz fichero.tgz

Funciones básicas de I/O en C

Dos familias de funciones básicas:

- printf: escribe en la entrada estándar (típicamente pantalla). Variantes sprintf (escribe en una cadena de caracteres) y fprintd (escribe en un fichero).
- scanf: lee de la entrada estándar (típicamente teclado).

Variantes sscanf (lee de una cadena de caracteres) y fscanf (lee de un fichero).

Consulta de ayuda en lenguaje C

- Comando **man**
  - man printf # Primera entrada sobre printf en el manual
  - man –a printf # Todas las entradas, se pasa de una a otra con 'q'
  - man 3 printf # La entrada 3 sobre la printf

- Para cada función C el **man** indica su synopsis. En ella se indica que ficheros es necesario incluir en el .c para que la compilación sea correcta. Por ejemplo, para el caso del man habrá que hacer un **#include <stdio.h>**

Ejercicio 2

EJERCICIO:

Escriba, haciendo uso exclusivamente de las funciones anteriores, un programa (ejercicio2.c) que lea de un fichero líneas con el formato **NN1:NN2 (cadena)** y lo vuelque a un salida estándar con el formato:
La cadena leída es "cadena" y la suma de los números es (NN1+NN2).

Cree un Makefile adecuado para el ejercicio anterior con funciones clean y tar (fichero destino ejercicio2.tgz).

Deposite este fichero (ejercicio2.tgz) en el directorio de entregables de la práctica 0.