



# **INTRODUCCIÓN**



**Docentes:**

- **Bottallo, Diego**
- **Ramis, Pablo**
- **Villar, Fernando**

● **WEB de la materia**

<http://www.fceia.unr.edu.ar/~diegob/taller1>

● **Plan de la materia**

<http://www.fceia.unr.edu.ar/~diegob/taller1/Plan.html>

● **MAIL**

[diego.bottallo@gmail.com](mailto:diego.bottallo@gmail.com)



## **Bibliografía**

### **Cabecera**

- ◆ El Lenguaje de Programación C (ANSI C) - Kernighan & Ritchie
- ◆ Apuntes de clase

### **Complementarios**

- ◆ Deitel & Deitel – Como Programar en C
- ◆ Advanced Linux Programming  
Mitchell, Oldham & - New Riders Publishing  
<http://www.advancedlinuxprogramming.com>



- **Como se aprueba la materia**

- ◆ Entregando **prácticas** en tiempo y forma, desarrollo individual o grupal

- ◆ Aprobando el **parcial** o **recuperatorio**

- ◆ Presentando el **Trabajo Práctico Final** (*también individual o grupal*)

- ◆ **Final**

- Condición **regular**:

- solo entrega el **Trabajo Práctico Final**

- Condición **libre**:

- además del **Trabajo Práctico Final** relizará un programa en PC



## ● **Recomendaciones**

- ◆ Asistencia a las clases teóricas
- ◆ Aprovechar los laboratorios para realizar las prácticas
- ◆ Practicar y practicar  
Consultar  
Programar – Equivocarse – Aprender de los errores  
Ser curioso
- ◆ Usar recursos de Internet con óptica crítica



## ● Introducción a C

### ◆ Lenguaje de Programación

Es un lenguaje formal, que permite interactuar con los sistemas, mediante la creación de programas.

Los **programas** pueden controlar el comportamiento lógico y/o físico de la máquina donde se ejecutan.

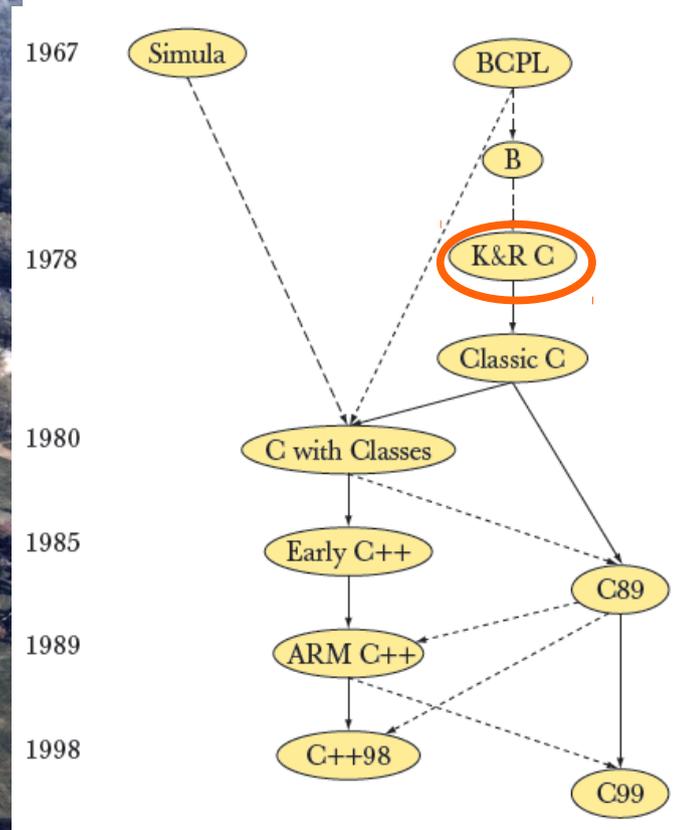
Con los programas se expresan los **algoritmos**, que son conjuntos de pasos lógicos relacionados para resolver un problema o situación específica.

Existen muchos lenguajes de programación diferentes, los distinguen su expresividad, nivel, sintaxis y semántica, o bien su ductilidad para ser aplicado en un contexto determinado.

## ● Introducción a C

- ◆ Lenguaje de **alto nivel** desarrollado entre 1969-1973 por Dennis Ritchie y Brian Kernighan

Evolución del lenguaje B, a su vez descendiente de BCPL



**C “born” in the Computer Science Research Department of Bell Labs in Murray Hill, NJ**

## ● Introducción a C

### ◆ Lenguaje de alto nivel

#### Assembler

```
.model small
.stack
.data
    saludo    db "Hola mundo!!!", "$"

.code

main proc
    mov     ax,seg saludo
    mov     ds,ax

    mov     ah,09
    lea     dx,saludo
    int     21h

;mensaje en pantalla

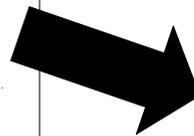
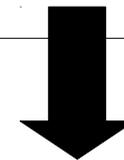
    mov     ax,4c00h
    int     21h

main endp
end main
```

#### C

```
#include <stdio.h>

int main() {
    printf("Hola mundo!!!");
    exit 0;
}
```





### ● Introducción a C

- ◆ C es “un assembler de alto nivel”, originalmente orientado a la implementación de sistemas operativos (manipulación de bits, punteros)
- ◆ C NO es un lenguaje **orientado a objetos**
- ◆ Es un lenguaje de programación muy **portable**, existen compiladores para casi todos los procesadores
- ◆ C es un lenguaje de programación **estructurado** (funciones, módulos)
- ◆ C es un lenguaje de programación **imperativo** (Pascal, Java) versus los lenguajes de programación declarativos (Haskell, ML)
- ◆ Java heredó gran parte de su sintaxis y semántica (función main, condicionales, bucles, funciones)

[http://es.wikipedia.org/wiki/C\\_\(lenguaje\\_de\\_programación\)](http://es.wikipedia.org/wiki/C_(lenguaje_de_programación))



## ● Programación imperativa vs declarativa

### ◆ Programacion imperativa

- Como
- Tipado
- Sentencias

### ◆ Programacion declarativa

- Que, no como
- Tipado / no-tipado
- Funciones

[http://es.wikipedia.org/wiki/Programacion\\_imperativa](http://es.wikipedia.org/wiki/Programacion_imperativa)

[http://es.wikipedia.org/wiki/Programacion\\_declarativa](http://es.wikipedia.org/wiki/Programacion_declarativa)



### ● Programación imperativa vs declarativa

```
#include<stdio.h>
```

```
long factorial(int n) {  
    if (n==0)  
        return n * factorial(n-1);  
};
```

```
int main() {  
    int num;  
    long f;  
  
    printf("Ingresar un numero entero para hallar su factorial: ");  
    scanf("%d",&num);  
  
    if(num<0)  
        printf("No se permiten numero negativos!");  
    else {  
        f = factorial(num);  
        printf("%d!=%ld",num,f);  
    }  
    return(0);  
}
```

### Declarativa (funcional)

```
fun factorial 0 = 1  
  | factorial n = n * factorial (n - 1)
```

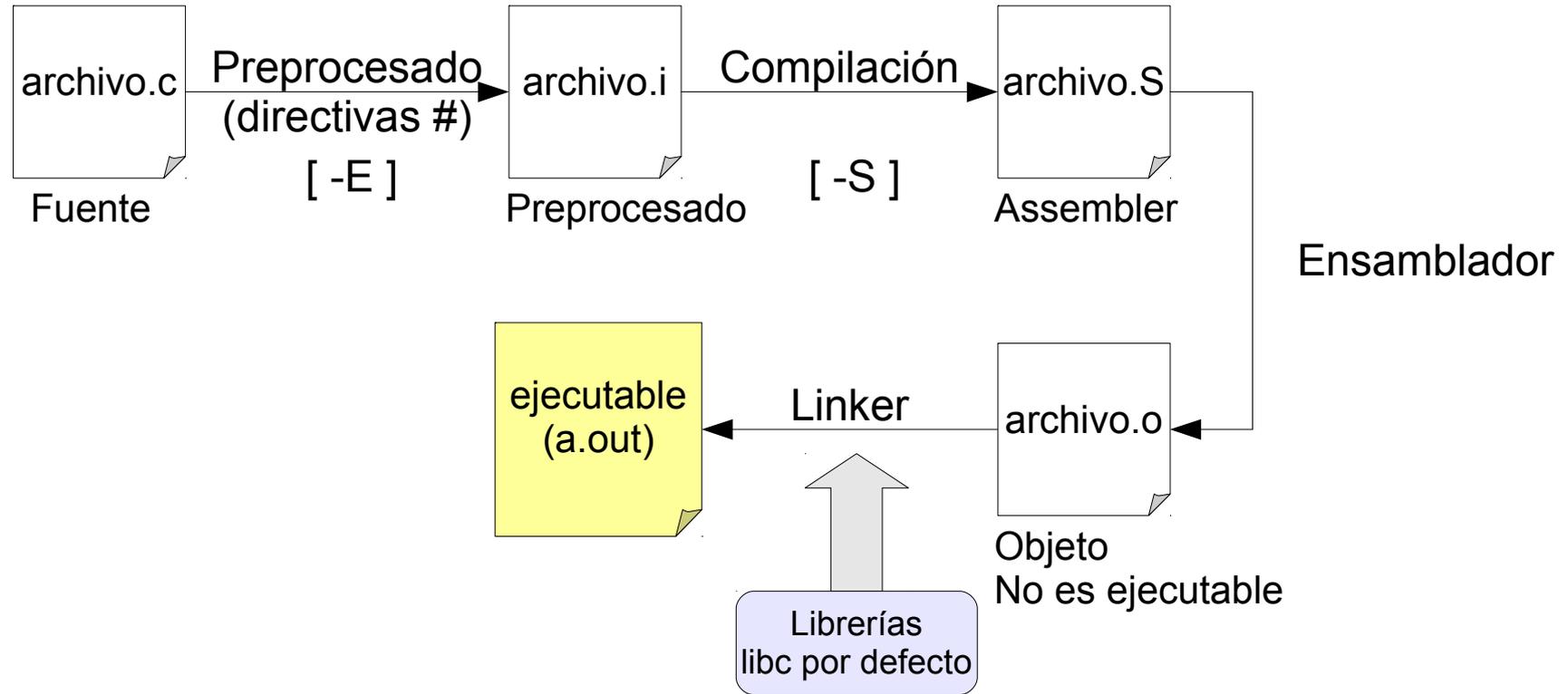


## ● **Compilación**

- ◆ C es un lenguaje **compilado**
- ◆ El **compilador** es el **gcc** (GNU project C and C++ Compiler)
- ◆ El **proceso de compilación** consta de las etapas
  - . preprocesamiento
  - . compilación
  - . assembly
  - . linking



# ● Preproceso de compilación



## Opciones de compilación

<http://gcc.gnu.org/onlinedocs/gcc-4.3.2/gcc/Overall-Options.html#Overall-Options>



## ● Compilación

◆ Es posible detener el proceso de compilación en cualquiera de las etapas citadas

◆ Ejemplo compilación

```
$ gcc -Wall programa.c -o programa
```

-Wall: Habilita todos los warnings sobre construcciones que resulten cuestionables y sean simples de evitar o modificar para evitar estos mensajes

Programa: **hola-2.c**

Ejercicio: Compilarlo sin y con la opción **-Wall**

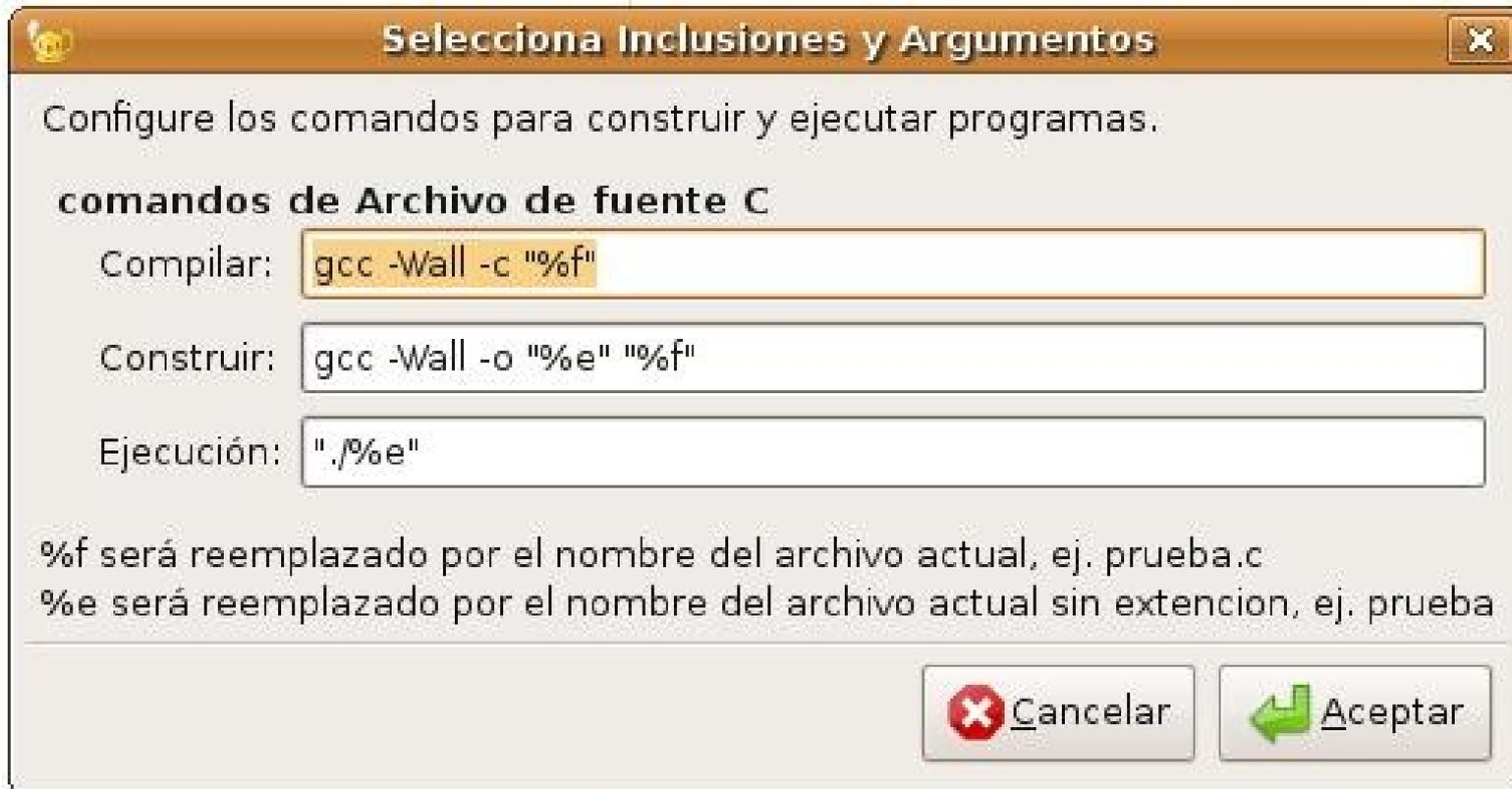


## ● Linker (enlazador)

- ◆ Cuando una función es invocada, el linker la localiza en la librería  
Emplea el path estándar para las librerías o el que se le pase mediante “gcc -L”
- ◆ Inserta la o las funciones en código objeto
- ◆ Si la función no es encontrada (nombre o parámetros incorrectos) produce un error

## ● Herramientas de desarrollo

- ◆ gcc
    - Ubuntu: `apt-get install build-essential`
- IDE (Integrated Development Environment)**
- ◆ Geany / Code::Blocks
    - Geany – Menu Construir (Inclusiones y argumentos) – se indica compilador





### ● Preprocesador

- ◆ Previo a la compilación tiene lugar el **pre-procesamiento**
- ◆ El **pre-procesador** reconoce órdenes para manipular constantes y macros, incluir archivos en el *fuelle* y dirigir la compilación posterior
- ◆ Tiene su propio lenguaje, muy sencillo, y es “independiente” del lenguaje C, aunque es absolutamente *estándar*.
- ◆ Todas las órdenes al preprocesador comienzan con un **carácter #** (#include, #define, #pragma, etc)
- ◆ Un programa C luego de ser procesado es “únicamente” C



## ● Header files (Archivos de cabecera)

- ◆ Especificados mediante las entradas **#include**
- ◆ Son archivos pre-compilados con funciones y declaraciones definidas en ellos
- ◆ Concepto: **modularidad**
- ◆ Poseen *extensión* **.h**
- ◆ Generalmente ubicados en el directorio **/usr/include**
- ◆ Pueden incluirse otros paths o rutas mediante varias opciones **-I** de **gcc**  
\$ gcc -I. -I/mis/headers programa.c -o programa



### ■ CASO DE ESTUDIO: Uso del lenguaje C en Sistemas Operativos Kernel Monolítico 4.4BSD

#### Código independiente de la plataforma

Category	Lines of code	Percentage of kernel
headers	9,393	4.6
initialization	1,107	0.6
kernel facilities	8,793	4.4
generic interfaces	4,782	2.4
interprocess communication	4,540	2.2
terminal handling	3,911	1.9
virtual memory	11,813	5.8
vnode management	7,954	3.9
filesystem naming	6,550	3.2
fast filestore	4,365	2.2
log-structure filestore	4,337	2.1
memory-based filestore	645	0.3
cd9660 filesystem	4,177	2.1
miscellaneous filesystems (10)	12,695	6.3
network filesystem	17,199	8.5
network communication	8,630	4.3
internet protocols	11,984	5.9
ISO protocols	23,924	11.8
X.25 protocols	10,626	5.3
XNS protocols	5,192	2.6
<b>total machine independent</b>	<b>162,617</b>	<b>80.4</b>

