

# INTRO



# Lic. Diego A. Bottallo

- **WEB de la materia**
  **http://www.fceia.unr.edu.ar/~diegob/taller2**

- **Plan de la materia**
  **http://www.fceia.unr.edu.ar/~diegob/taller2/Plan.html**

- **MAIL**
  **diego.bottallo@gmail.com**

- **Bibliografía**
  - El Lenguaje de Programación C (ANSI C) - Kernighan & Ritchie
  - Advanced Linux Programming
    Mitchell, Oldham & - New Riders Publishing
    `http://www.advancedlinuxprogramming.com`
  - Apuntes de clase



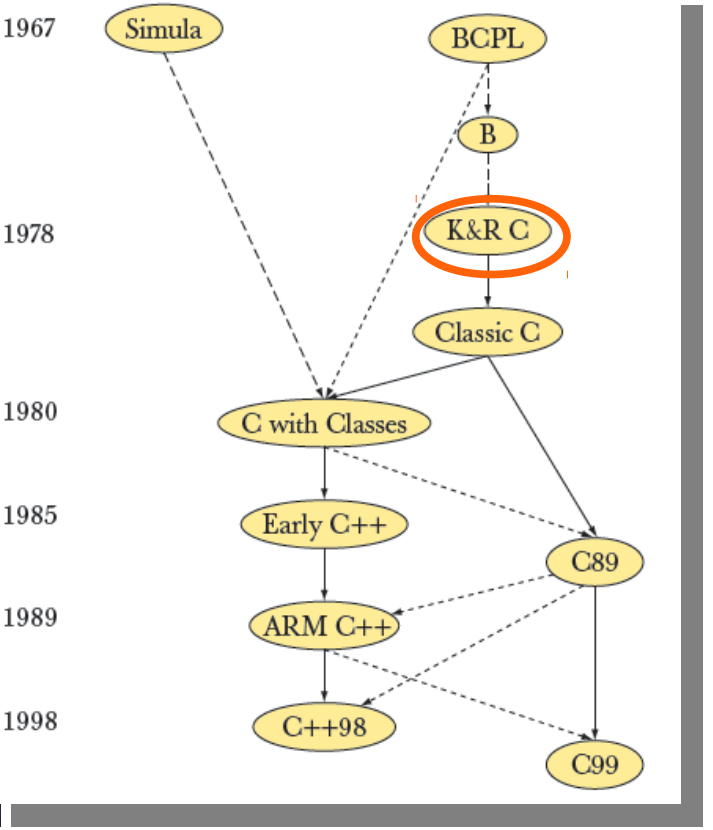
## Como se aprueba la materia

- Entregando prácticas o trabajo práctico en tiempo y forma – Cantidad a definir
- Aprobando el parcial/recuperatorio
- Presentando el trabajo final

Instituto Politécnico Superior

# 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

- 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)



# 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_declarativa



# Programación imperativa vs declarativa

```
#include<stdio.h>

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

int main() {
   int num;
   long f;

   printf("ENTER A NUMBER TO FIND FACTORIAL: ");
   scanf("%d",&num);

   if(num<0)
      printf("NEGATIVE NUMBERS ARE NOT ALLOWED");
   else {
      f = factorial(num);
      printf("%d!=%ld",num,f);
   }
   return(0);
}
```

```
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
- 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



# CASO DE ESTUDIO: Kernel Monolítico 4.4BSD

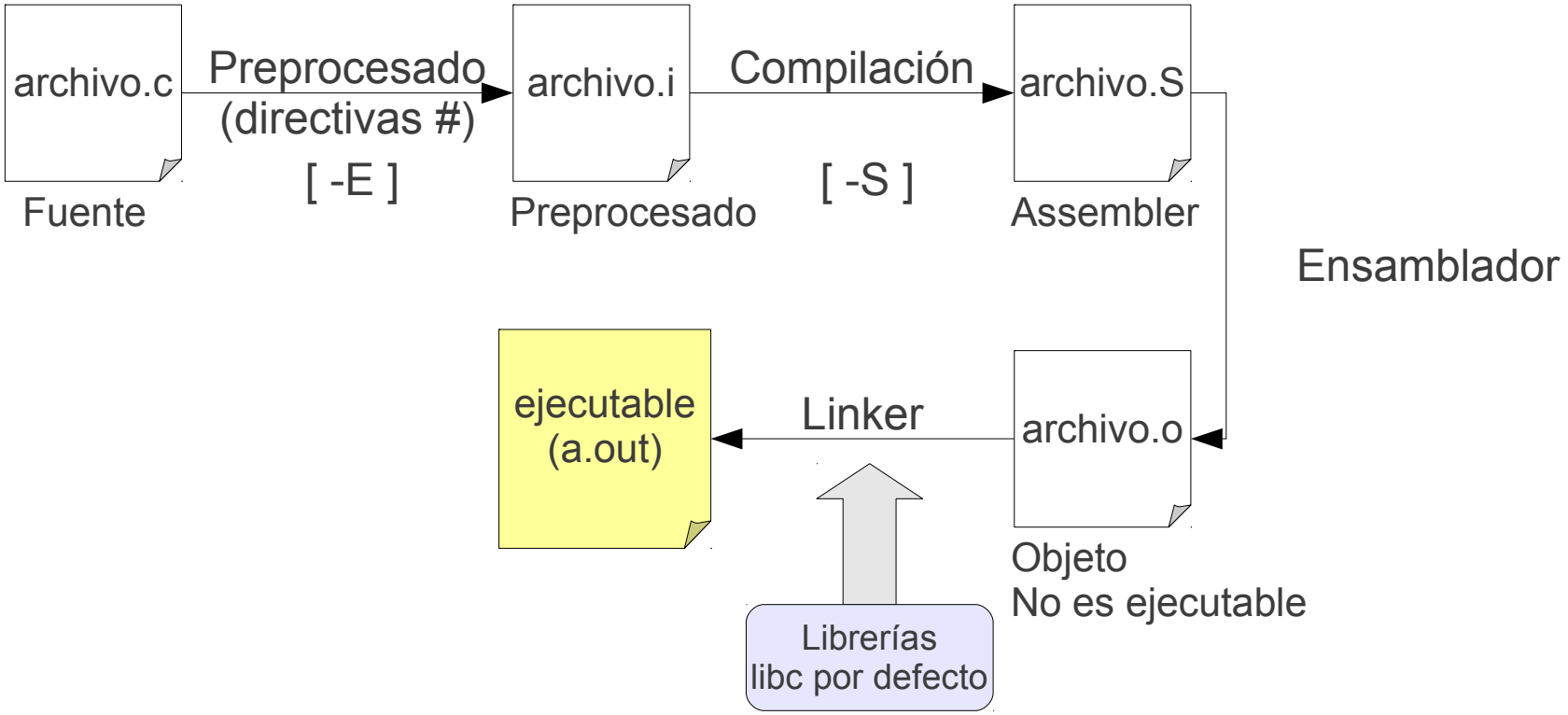
## Código independiente de la plataforma

| 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 |
| **total machine independent** | **162,617** | **80.4** |

**Code**:
C
Header files
Assembly

C 98%
Assembly 2%

# Preproceso de compilación

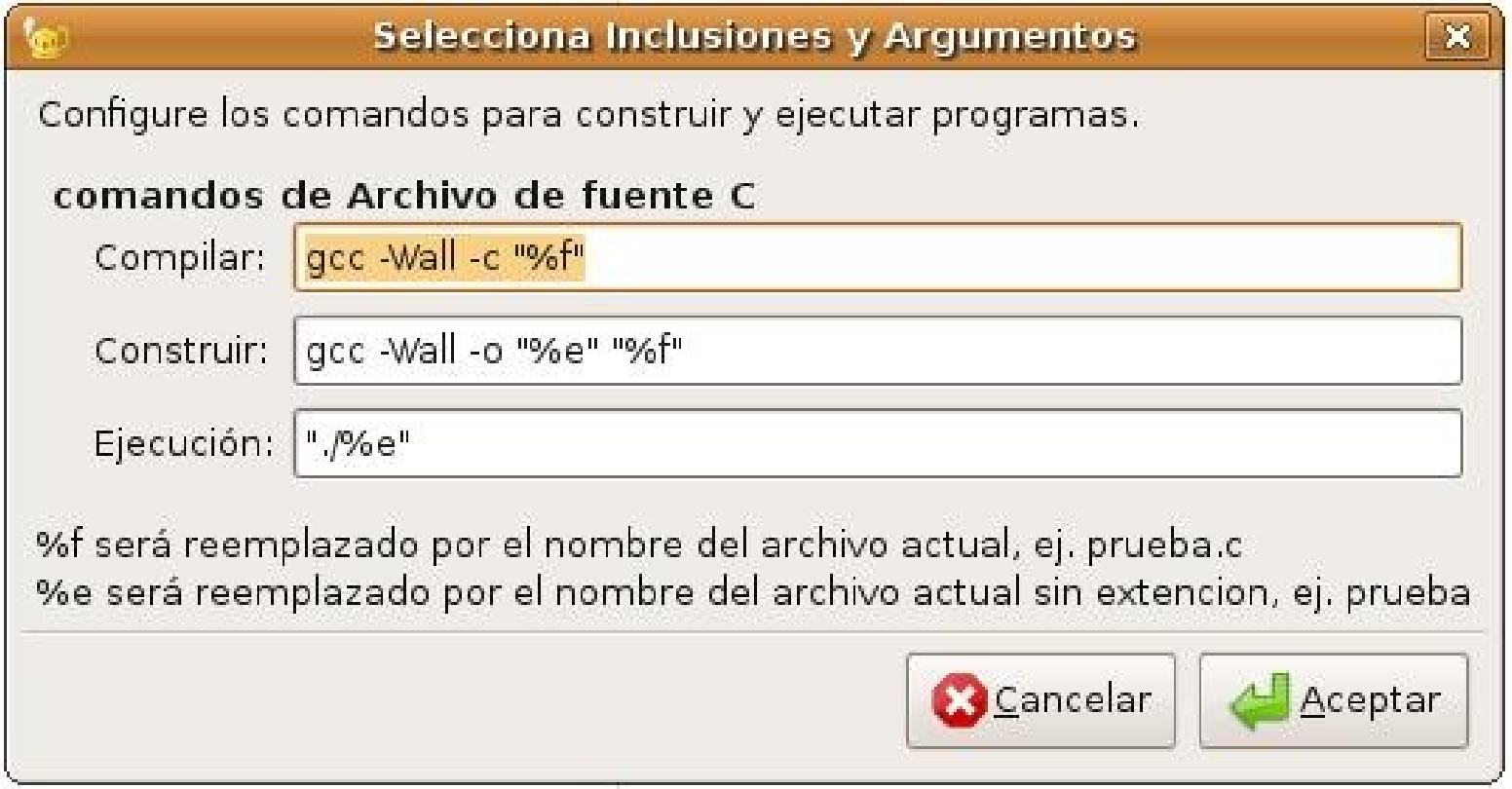



**Opciones de compilación**
http://gcc.gnu.org/onlinedocs/gcc-4.3.2/gcc/Overall-Options.html#Overall-Options

# Herramientas de desarrollo

- **gcc**
  - Ubuntu: **apt-get install build-essential**

- **Geany**
  - Construir (Inclusiones y argumentos)





# Preprocesador

- Previa a la compilación tiene lugar el pre-procesamiento
- El pre-procesador reconoce órdenes para manipular constantes y macros, incluir archivos en el fuente 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 # en la primera columna del texto (#include, #define)
- Un programa C luego de ser procesado es “únicamente” C



# Header files

- Especificados mediante las entradas #include
- Son archivos pre-compilados con funciones y declaraciones definidas en ellos
- Poseen extensión .h
- Generalmente ubicados en /usr/include
- Pueden incluirse otros paths mediante varias opciones -I de gcc

```
$ gcc -I. -I/mis/headers programa.c -o programa
```



# 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