Instituto
Politécnico
Superior

Lic. Diego A. Bottallo

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

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

- **WEB**
  http://www.fceia.unr.edu.ar/~diegob/

- **Bibliografía**
  - Tanenbaum, Andrew - Sistemas Operativos Modernos
  - Silberschatz & Galvin - Sistemas Operativos
  - Nichols, Buttlar & Proulx Farrell - Pthreads Programming - O'Reilly
  - Mitchell, Oldham & - Advanced Linux Programming - New Riders Publishing
    http://www.advancedlinuxprogramming.com
  - Apuntes de clase

Como se aprueba la materia

- Rindiendo la parte de Guido y la mia de manera independiente

MI PARTE

- Entregando prácticas en tiempo y forma

  Prácticas de programación (sobre entorno Linux / Unix)

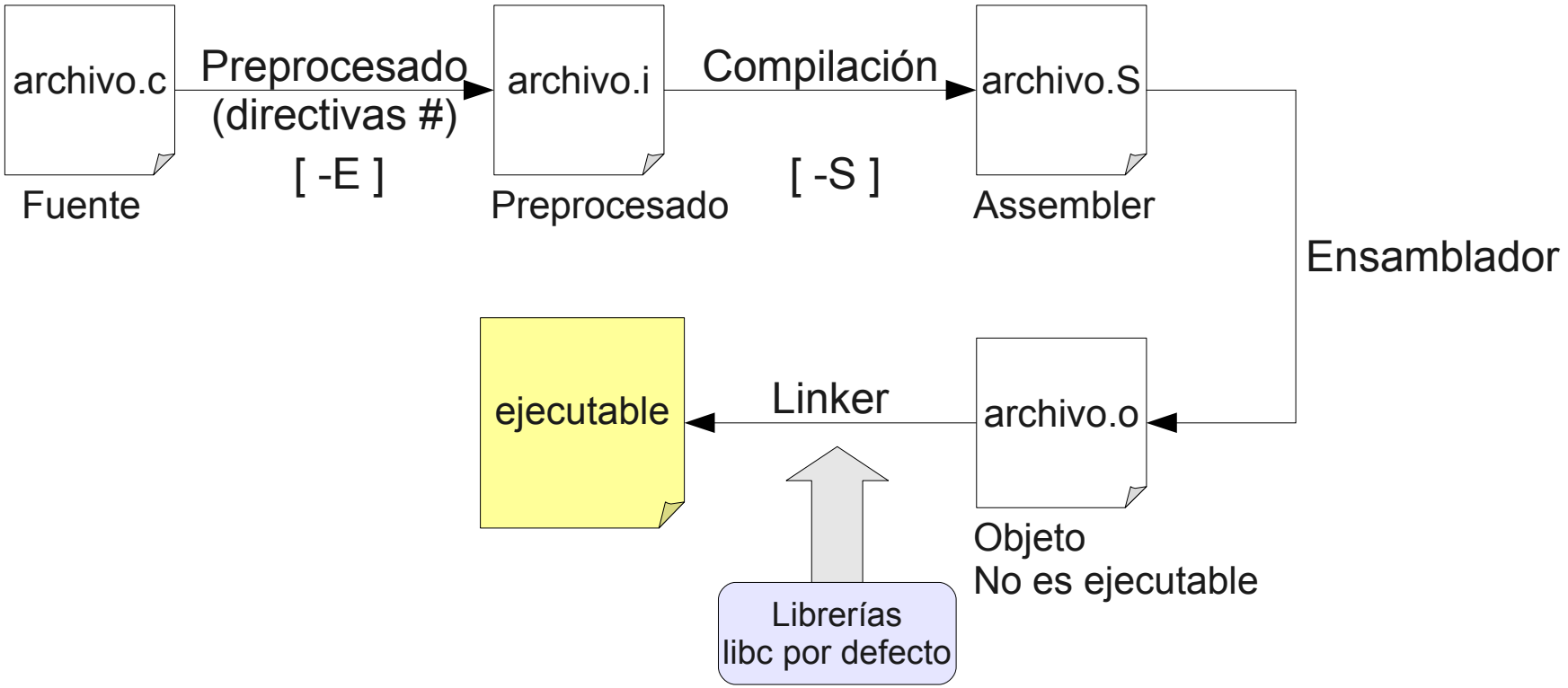
  Se entregan vía mail en grupo ***(2 ≤ integrantes ≤ 4)***

- Aprobando el parcial/recuperatorio

- Aprobando el final

Breve introducción a C

- **Compilación**
  Comando gcc

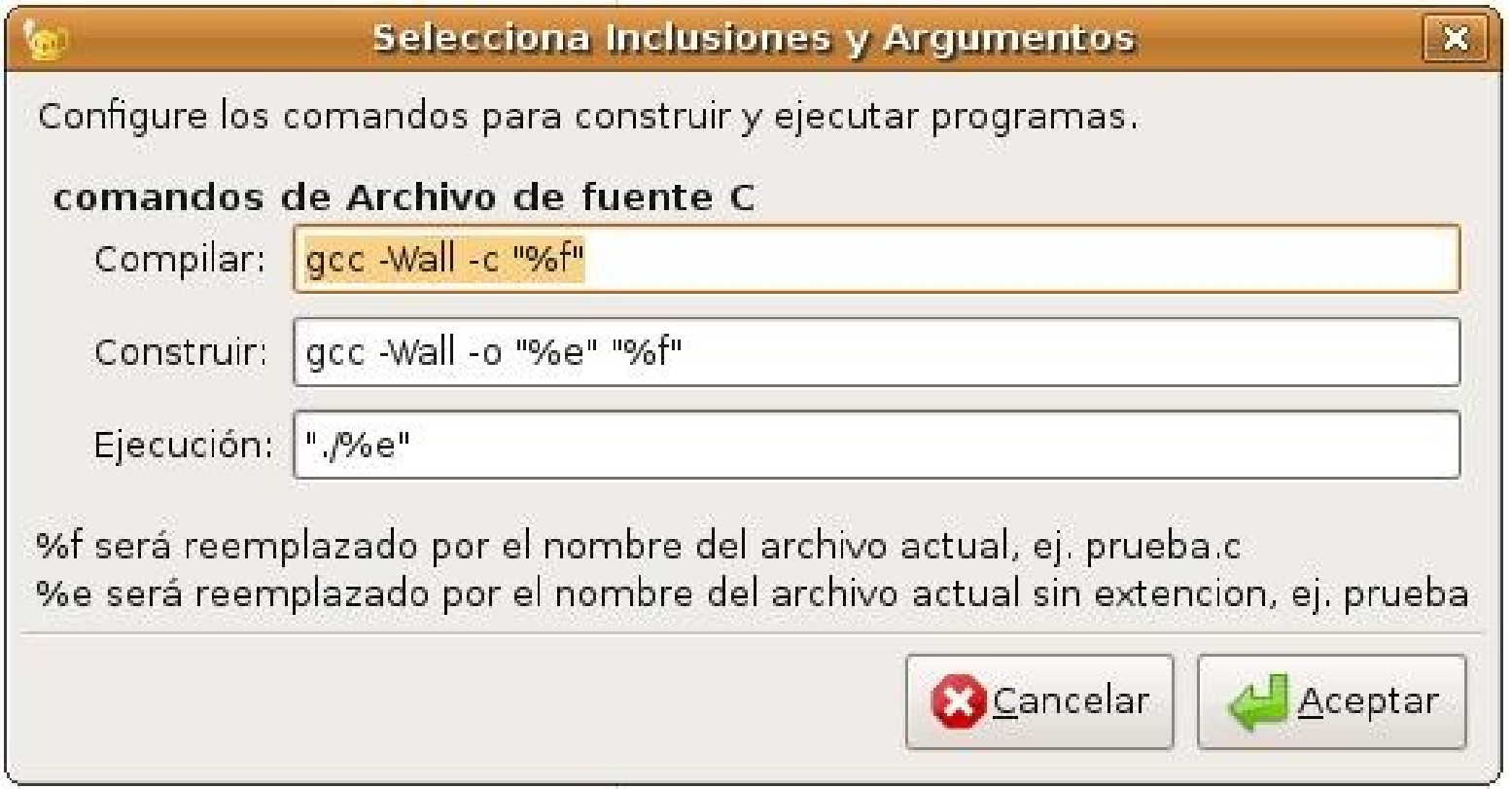


Opciones de compilación
`http://gcc.gnu.org/onlinedocs/gcc-4.3.2/gcc/Overall-Options.html#Overall-Options`

Herramientas de desarrollo

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

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



Breve introducción a C

- **Preprocesador**

  Doc: http://gcc.gnu.org/onlinedocs/cpp

  `# include <archivo.h>`

  Inserta `archivo.h` buscándolo en `/usr/include` y los directorios de búsqueda estándar (`/usr/local/include`) o mediante las opciones de línea de comando `-Idir` y `-I.`

  `# include "archivo.h"`

  Busca archivo en .

Macros: Sustitución de texto

```
#define identificador texto_reemplazo
#define identificador texto_reemplazo(arg, arg, ...)
```

No hay revisión de tipos

No va "="

Sin espacio

Ejemplos:

```
#define PI 3.141592
#define min(X, Y)  ((X) < (Y) ? (X) : (Y))
#define vabs(X) \
            ((X)>=0 ? (X) : -(X))
```

Breve introducción a C

- **Macros:** Problemas

```
#define prod(X, Y)  (X * Y)
a = prod(1, 2);        ⇨  a = (1 * 2);        // a=2
b = prod(1+2, 3+4);    ⇨  b = (1+2 * 3+4);    // 1+6+4=11

#define vabs(X) ((X)>=0 ? (X) : -(X))
vabs(a++)
 a ⇦ 4
```

Breve introducción a C

- Macros: Entrecomillado

Un texto pasado a una macro se puede tratar como si estuviese entre comillas usando la directiva "#".

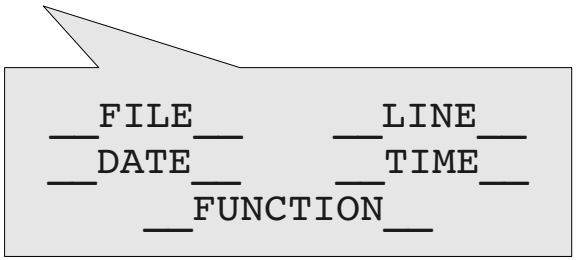
```
#define ENTRECOMILLAR(x) #x
```

el código `printf("%s\n", ENTRECOMILLAR(1+2));`
se expande a `printf("%s\n", "1+2");`

Útil para la concatenación de cadenas automáticas para depurar macros.

```
#define ver(x, format) printf("%s:%u: %s="format, __FILE__, __LINE__, #x, x)

int una_funcion() {
    int var=1;
    /* aquí va código que manipula a var */
    ver(var, "%d");
}
```



imprime el nombre de la expresión y su valor así como el nombre del archivo y la línea donde se ejecuta.

Breve introducción a C

- Macros: Pasting

```
#define B(x,y) xy
#define C(x,y) x##y

B(pthreads_,mutex) → pthreads_ mutex
C(pthreads_,mutex) → pthreads_mutex
```

Breve introducción a C

- Macros:

```
$ gcc -DVALOR=100 -W -Wall -o programa programa.c
```

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

int main () {
   printf("%f\n", sqrt(2));
   return 0;
}


$ gcc -W -Wall raiz.c -Dsqrt=ceil
```

Breve introducción a C

- **Macro assert**

- El header file **<assert.h>** define esta macro
- *Assert* = aserto (aserción, afirmación, confirmación)
- Evalua una condición 'test' y dependiendo del resultado, puede abortar el programa. Útil para *debugging*

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

int main () {
    assert(1>2);
    return 0;
}
$ gcc -W -Wall assert.c -o assert
$ ./assert
assert: assert.c:5: main: Assertion `1>2' failed.
Cancelado
```

- Si se coloca **#define NDEBUG** antes de **#include <assert.h>** o se compila con **-DNDEBUG** (no depurar) todas las sentencias **assert** se ignoran

Breve introducción a C

- Macros: Compilación condicional
  - Existe un conjunto de directivas que puede usarse para realizar compilaciones condicionales

`#if, #ifdef, #ifndef, #else, #elif y #endif (también #define #undef)`

```
#define __WINDOWS__
#ifdef __WINDOWS__
#include <windows.h>
#else
#include <unistd.h>
#endif
```

  - `#if` permite utilizar `&&, ||, !, ==, <, >,` etc.

Ejemplo:

```
#if VERSION==3.0 && !defined(TEXT)
  ...
  ...
#endif
```

Breve introducción a C

- Macros: vs. Funciones
  - Eficiencia

    Menos sobrecarga (pasar parámetros a la pila, realizar un salto, recibir parámetros ...)

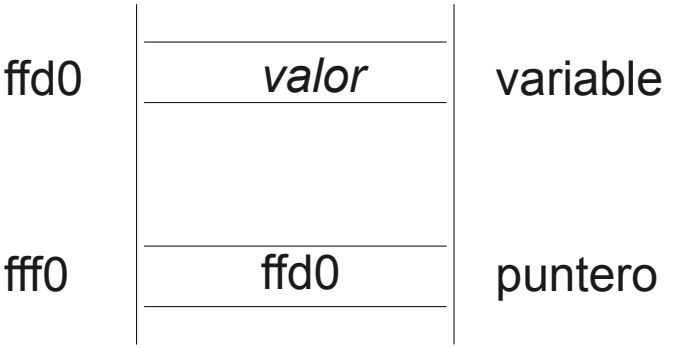
    Cálculos en *compilation time* en lugar de *run time*
  - Menor potencia y expresividad
  - Concurrencia limitada
  - Preprocesadas
  - C tiene muchas macros en sus librerías ejemplo: **`getchar()`**

    Indice de macros **`http://c.conclase.net/librerias/macros.php`**

Breve introducción a C

- **Punteros en C**
  - Un puntero es una variable que almacena direccione de memoria

```
tipo *puntero;
tipo variable;
puntero = &variable;
```



  - Un puntero está asociado a un tipo y contiene como valor una dirección

  - Operaciones
    - Asignarle un valor `puntero=direccion` `puntero=(cast)puntero`

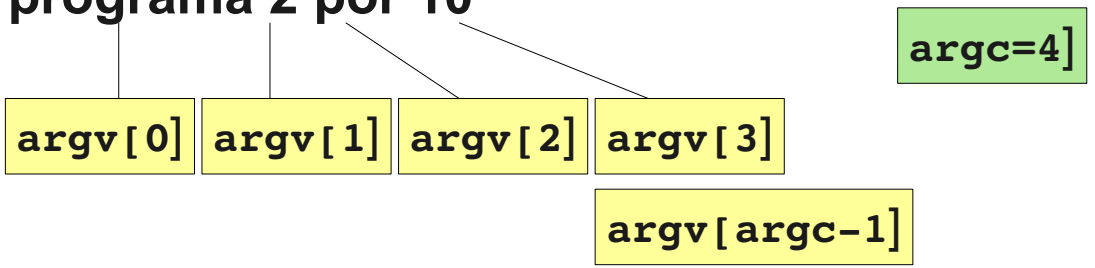
Breve introducción a C

- **Punteros en C**
  - Operadores * y &
  - Asignación y comparación entre punteros
  - Operadores: Aritmética ++, +=, --, -=, -
  - Punteros NULL y void
  - Punteros a función

Breve introducción a C

- **Argumentos en línea de comandos**
  - Al invocar a main se pueden emplear dos argumentos
    - **argc** (argument count) **int**
    - **argv** (argument vector) **char *[]** puntero a un arreglo de cadenas una cadena == un argumento

programa 2 por 10




```
#include <stdio.h>

int main (int argc, char *argv[]) {
   int i;
   for (i = 0; i<argc; i++)
      printf("argv[%d]=%s\n", i, argv[i]);
   return 0;
}
```

```
$ ./args 4 + cadena
argv[0]=./args
argv[1]=4
argv[2]=+
argv[3]=cadena
```