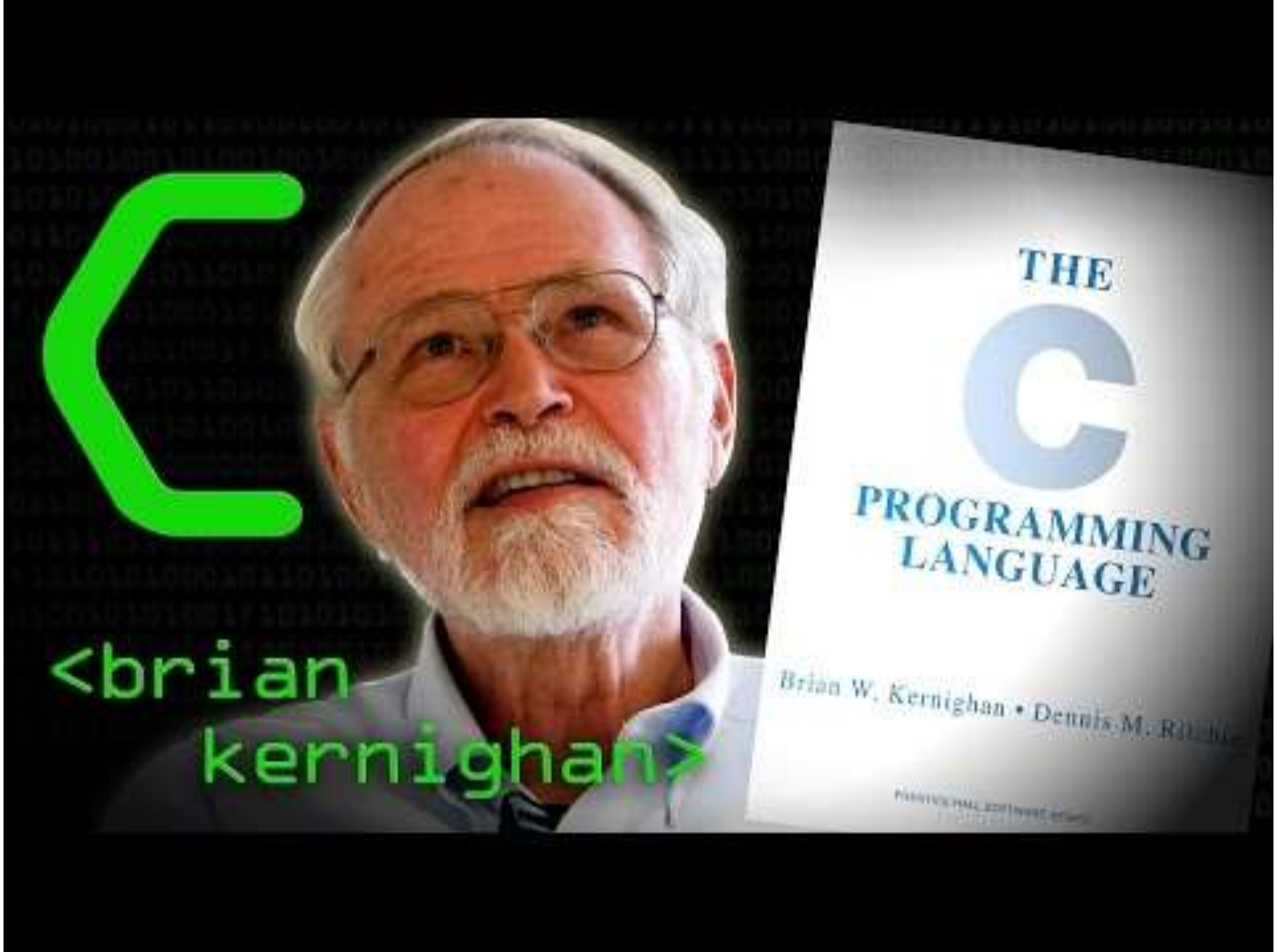
PROGRAMACIÓN DE SISTEMAS

El lenguaje de Programación C

Mg. Edith Giovanna Cano Mamani

El lenguaje de programación C

Basado en el curso: Effective Programming in C and UNIX del CMU


C
<brian
kernighan>
THE
C
PROGRAMMING
LANGUAGE
Brian W. Kernighan • Dennis M. Ri

Objetivos

- Entender la Programación de Alto y Bajo Nivel.
- Historia del Lenguaje de Programación C.
- Ejercicios en Lenguaje de Programación C.

Lenguajes de Bajo y Alto Nivel

Lenguaje de Bajo Nivel.

- Sus instrucciones consisten en ejercer un control directo sobre el hardware, se condiciona por la estructura física de la computadora.
- No implica que sea menos potente que uno Lenguaje de Alto Nivel.
- Usado en Sistemas Operativos o en controladores de dispositivos.
- Ejemplo. Lenguaje Ensamblador.

Lenguajes de Bajo y Alto Nivel

Lenguaje de Alto Nivel.

- Permite una mayor flexibilidad al abstraerse o ser literal. Permite una mejor comunicación entre lenguaje oral y el lenguaje máquina, es decir entre la escritura del programa y su compilación, varios de ellos están orientados a objetos,
- Expresa el algoritmo de acuerdo a la capacidad cognitiva humana.
- Ejemplo: Java, PL/SQL, C#, Cobol.

Historia

- 1969 El Lenguaje B (Ken Thompson), desarrollo Inicial basado en el Lenguaje BLCP (Martin Richard).
- 1970 Dennis Ritchie da el nombre de Unix derivado de MULTICS.
- 1972 Se crea el Lenguaje C (Dennis Ritchie).

Historia (…)

- 1974 Unix hace su primera aparición.
- 1978 Brian Kernighan y Dennis Ritchie publican el libro El Lenguaje de Programación C.
- 1980 Bjarne Stroustrup desarrolla el Lenguaje de Programación C++.
- 1990 Rectificación Estándar ISO.
- Unix inicialmente desarrollado en Lenguaje ensamblador y posteriormente en Lenguaje C.

Dennis Ritchie, Ken Thompson, and Brian Kernighan

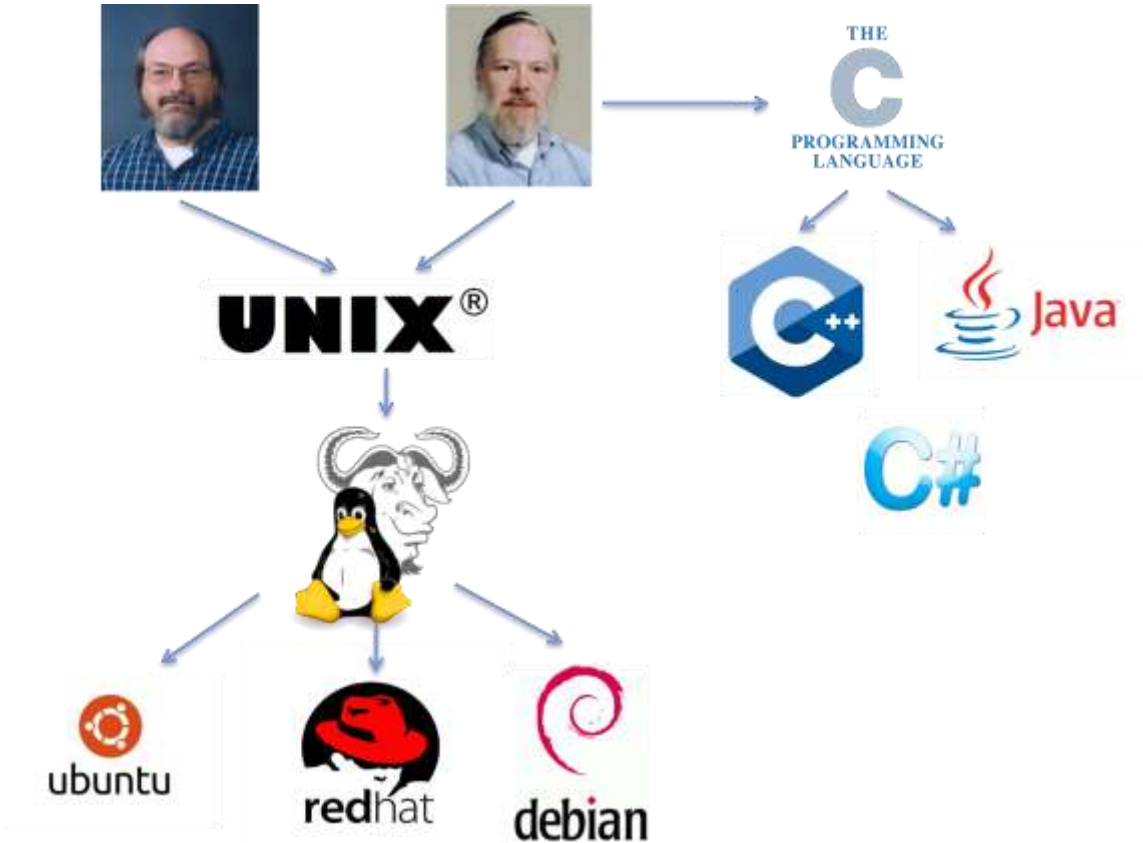
Características del Lenguaje C

- Utilización de Comandos Breves.
- Modelo de programación imperativo.
- Aparición de Funciones.
- Utilización de Punteros.
- Permite una programación de bajo nivel (Lenguaje Intermedio).
- Utilización de Arreglos.

Ventajas del Lenguaje C

- Permite que el programador pueda tener mucho mayor control y detalle sobre la utilización de la memoria por el programa (punteros).
  - Esto también podría verse como una desventaja, por que delega mayores responsabilidades al programador.
- Es un lenguaje de alto nivel.
- El código fuente es portable, aunque las bibliotecas siempre serán nativas del S.O.
- Genera código eficiente.

Derivados del Lenguaje C



Items para revisar en código

- Sintaxis
- Compilación
- Ejecución
- El Operador Sizeof
- Tipos de datos básicos
- Cualificadores
- Constantes entera y reales
- Constantes hexadeciamles
- Constantes Caracter
- Struct

Sintaxis básica

Un programa ejecutable en C requiere de una función denominada main, con el código a ser ejecutado por un programa. En el siguiente ejemplo se escribe un código que deberá ser escrito en un archivo con extensión .c (primero.c)

```
#include <stdio.h>

main(){

  printf(“Hello world\n”);

}
```

Compilación

Para ejecutar este código es necesario contar con un compilador que traduzca este código a algo que el sistema operativo pueda ejecutar. Nosotros usaremos gcc un compilador de software libre que entiende el lenguaje C (y también otros lenguajes)

```
$ gcc primero.c
```

Esto creará un archivo ejecutable denominado a.out, si se desea cambiar el nombre del archivo ejecutable generado se debe usar la opción -o, e indicar el nombre del archivo ejecutable que se desea.

Ejecución

Para ejecutar el programa se debe escribir

```
$ ./a.out
```

El punto slash (./) se usa para indicar que el shell debe buscar el programa ejecutable en el directorio actual y no las rutas señaladas por la variable PATH.

Cuando la compilación involucra varios archivos, es mejor usar una herramienta que automatice este proceso. Tradicionalmente se usó el programa make, que se verá más adelante.

Tipos de datos

Al igual que Java el lenguaje C es fuertemente tipado y posee una gran variedad de tipos y cualificadores, el tamaño de estos tipos será muy importante para nuestro curso, por lo que los describiremos indicando su tamaño. Es importante resaltar que el tamaño puede variar dependiendo del sistema operativo en el que se esté trabajando.

El Operador Sizeof

El operador unario sizeof() se utiliza para obtener el tamaño de un tipo de datos en bytes en bytes. No devolverá el tamaño de las variables o instancias. Al ser un operador hay que considerar la precedencia de operadores en su uso.

Revisar man de operadores:

```
$ man operator
```

Ejemplo:

Input : sizeof(byte);

Output : 1

Input : sizeof(int);

Output : 4

Tipos de datos básicos

Esto depende de la arquitectura del computador que se esté usando

Tipo	Tam. Bits	Dígitos de precisión	Rango	
			Min	Max
Bool	8	0	0	1
Char	8	2	-128	127
Signed char	8	2	-128	127
unsigned char	8	2	0	255
short int	16	4	-32,768	32,767
unsigned short int	16	4	0	65,535
Int	32	9	-2,147,483,648	2,147,483,647
unsigned int	32	9	0	4,294,967,295
long int	32	9	-2,147,483,648	2,147,483,647
unsigned long int	32	9	0	4,294,967,295
long long int	64	18	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long int	64	18	0	18,446,744,073,709,551,615
Float	32	6	1.17549e-38	3.40282e+38
Double	64	15	2.22507e-308	1.79769e+308

Cualificadores

Los calificadores de tipo son las palabras clave que describen semánticas adicionales sobre un tipo. Son una parte integral de las firmas de tipo.

Los calificadores de tipo tienen la misión de modificar el rango de valores de un determinado tipo de variable. Estos calificadores son cuatro:

signed

Le indica a la variable que va a llevar signo. Es el utilizado por defecto.

Tipo		Tamaño	Rangos de valores
signed char		1 byte	-128 a 127
signed int		2 bytes	-32768 a 32767

Cualificadores

unsigned

Le indica a la variable que no va a llevar signo (sin valores negativos).

Tipo	Tamaño	Rangos de valores
unsigned char	1 byte	0 a 255
unsigned int	2 bytes	0 a 65535

short

Rango de valores en formato corto (limitado). Es el utilizado por defecto.

Tipo	Tamaño	Rangos de valores
short int	2 bytes	-32768 a 32767

Cualificadores

long

Rango de valores en formato largo (ampliado).

Tipo	Tamaño	Rangos de valores
long int	4 byte	-2.147.483.648 a 2.147.483.647
long double	10 bytes	-3'36 E-4932 a 1'18 E+4932

También es posible combinar calificadores entre sí:

signed long int = long int = long

unsigned long int = unsigned long 4 bytes 0 a 4.294.967.295 (El mayor entero permitido en 'C')

Constantes Enteras

Cualquier número escrito es una constante, sin embargo hay tratamientos distintos para enteros, reales y caracteres.

En el caso de números enteros, se puede especificar si se requiere algo de memoria extra para almacenarlos agregando una l o L al final del número; por ejemplo, 65535L.

También se puede especificar si el número se almacenará sin signo agregando la letra U o u al final; se pueden combinar las letras ul o UL si se necesita.

Constantes reales

Para el caso de los números de punto flotante, el punto decimal define claramente su tipo de valor, sin embargo también se pueden escribir usando la notación científica agregando la letra e como indicador del inicio del número del exponente; por ejemplo 5e-2 equivale a 0.05; agregando la letra l o L se puede especificar que la constante se debe almacenar como un long double

Constantes Hexadecimales y Octales

La representación de números hexadecimales y octales es importante porque permite una escritura abreviada de números binarios, que son los que realmente entiende el computador. Para representar números constantes octales es suficiente poner un 0 (cero) a la izquierda los dígitos.

Para representar números hexadecimales se debe agregar `0X` o `0x` a la izquierda de los dígitos.

Constantes Caracter

A diferencia de Java, en C la representación directa de un caracter es directamente como número, aunque su impresión sea de un caracter de texto. Al ser números, es posible sumarles valores o restarlos directamente, o compararlos como si fueran números, sin ningún tipo de conversión. Esto tiene un impacto importante en la concepción de programas que trabajen con Strings, que como se verá más adelante, son arreglos de caracteres.

Struct

struct ejemplo { char c; int i;}; La palabra reservada struct indica se está definiendo una estructura. El identificador ejemplo es el nombre de la estructura. Las variables declaradas dentro de las llaves de la definición de estructura son los miembros de la estructura.

struct ejemplo { char c; int i;};

Struct y clases

De manera similar a las clases de Java, en C es posible agrupar tipos de datos distintos en una sola entidad usando struct, sin embargo struct no crea un nuevo tipo de datos y tampoco permite asociar métodos, así que sólo se usa para **la agrupación de datos**.

```
struct Complejo {

  double real;

  doulbe img;

};
```

Es importante hacer notar, que estas variables estarán en regiones de memoria contigua, aunque esto no parece muy relevante en este momento, será una de las propiedades más importantes que ofrece el lenguaje C y que permite usar las estructuras de datos en la solución de problemas reales.

Variables de tipo Struct

La declaración de una variable del tipo de la estructura requiere que se use la palabra struct necesariamente

```
struct Complejo c1;
```

El acceso a los datos de una estructura es idéntico al acceso de los atributos de un objeto en java, es decir, usando el operador punto (.).

```
c1.real = 1.5;
```

```
c2.img = 3.2;
```

Las estructuras no cuentan con un constructor, pero también es posible inicializar una nueva estructura con valores iniciales.

```
struct Complejo c2 = {4.3, 5.7};
```

Funciones que devuelven struct

Es posible crear funciones que devuelvan nuevas estructuras inicializadas, al estilo de los constructores de java, pero estos no formarán parte de la estructura, serán funciones simples sin ninguna relación sintáctica con el tipo de dato struct.

```
struct Complejo creaComplejo(double real, double img){

  struct Complejo c;

  c.real = real;

  c.img = img;

  return c;

}
```

Estructuras de datos con Struct

Se puede usar la agrupación de datos que crea Struct para crear estructuras de datos como listas, por ejemplo

```
struct Node {

  int data;

  struct Node* next;

};

struct Node *list = null;
```

En este ejemplo el dato next, será un puntero al siguiente nodo, el tema de punteros se verá en la próxima clase.

typedef

Permite crear nuevos tipos de datos, por ejemplo podemos crear el siguiente nuevo tipo

```
typedef unsigned int size_t
```

En este caso size_t es un sinónimo para unsigned int.

```
size_t x = 2;
```

Debido a que un código de C puede ser ejecutado en distintas arquitecturas (máquinas) en algunas de ellas los tipos de datos pueden tener distinto tamaño, sin embargo size_t puede ser usado para focalizar este problema únicamente en la declaración de **`size_t`**.

Typedef y struct

La combinación de typedef y struct puede ser muy útil en la implementación de estructuras de datos, recordando el ejemplo anterior

```
typedef struct Node {
  int data;
  ListNode* next;
} ListNode;
ListNode *list = null;
```

¿Preguntas?

Procesos

https://tldp.org/LDP/intro-linux/html/sect_04_06.html

Entrada y salida

https://tldp.org/LDP/intro-linux/html/sect_05_05.html