UNIVERSIDAD NACIONAL DE HURLINGHAM

Universidad Nacional de San Martín

Overview of C Programming language

Leandro Luciano Gagliardi
lgagliardi@unsam.edu.ar

Lenguaje C

Luego del desarrollo de lenguajes de propósito específico, se sintió la necesidad de un lenguaje que pudiera soportar la mayoría de los propósitos.
El lenguaje **C** fue desarrollado en los años 70 en los laboratorios Bell por Dennis Ritchie. Inicialmente fue diseñado para programar en el sistema operativo llamado **UNIX**. Después de la llegada de **C**, todo el sistema operativo **UNIX** fue reescrito con él. Ahora casi todo el sistema operativo **UNIX** y las herramientas que lo acompañan, incluido el compilador de **C**, están escritos en **C**. El lenguaje **C** se deriva del lenguaje **B**, que fue escrito por Ken Thompson en los laboratorios AT&T Bell. En 1982, el **ANSI** (American National Standards Institute) formó un comité para estandarizar el lenguaje **C**. Finalmente, en 1989, se introdujo el estándar para el lenguaje C, conocido como **ANSI C**. En general, la mayoría de los compiladores modernos se ajustan a este estándar.

¿Por qué estudiamos C?

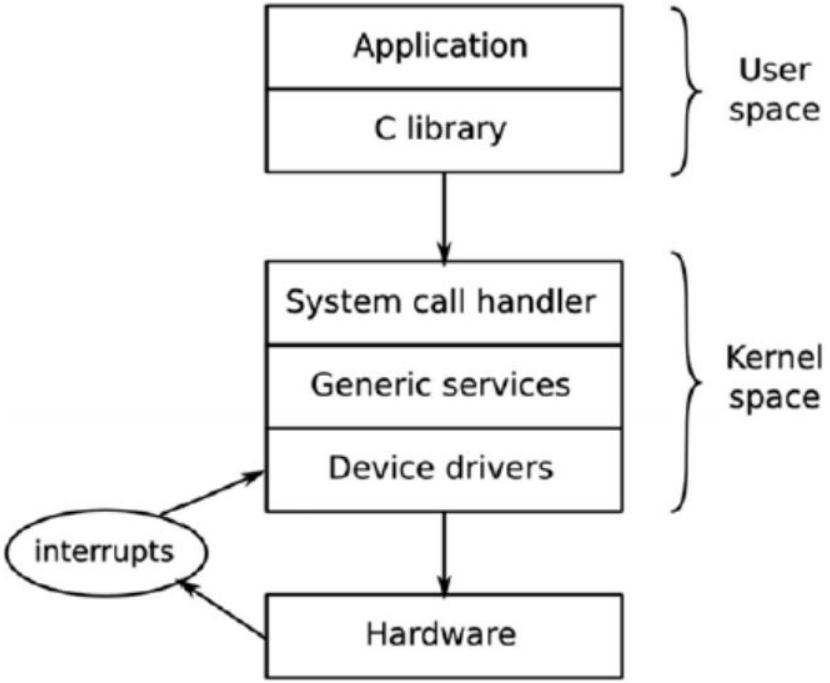
- Eficiencia y Control de Recursos

```
C Programming -> time ./a.out

real    0m0,003s
user    0m0,000s
sys     0m0,002s
```

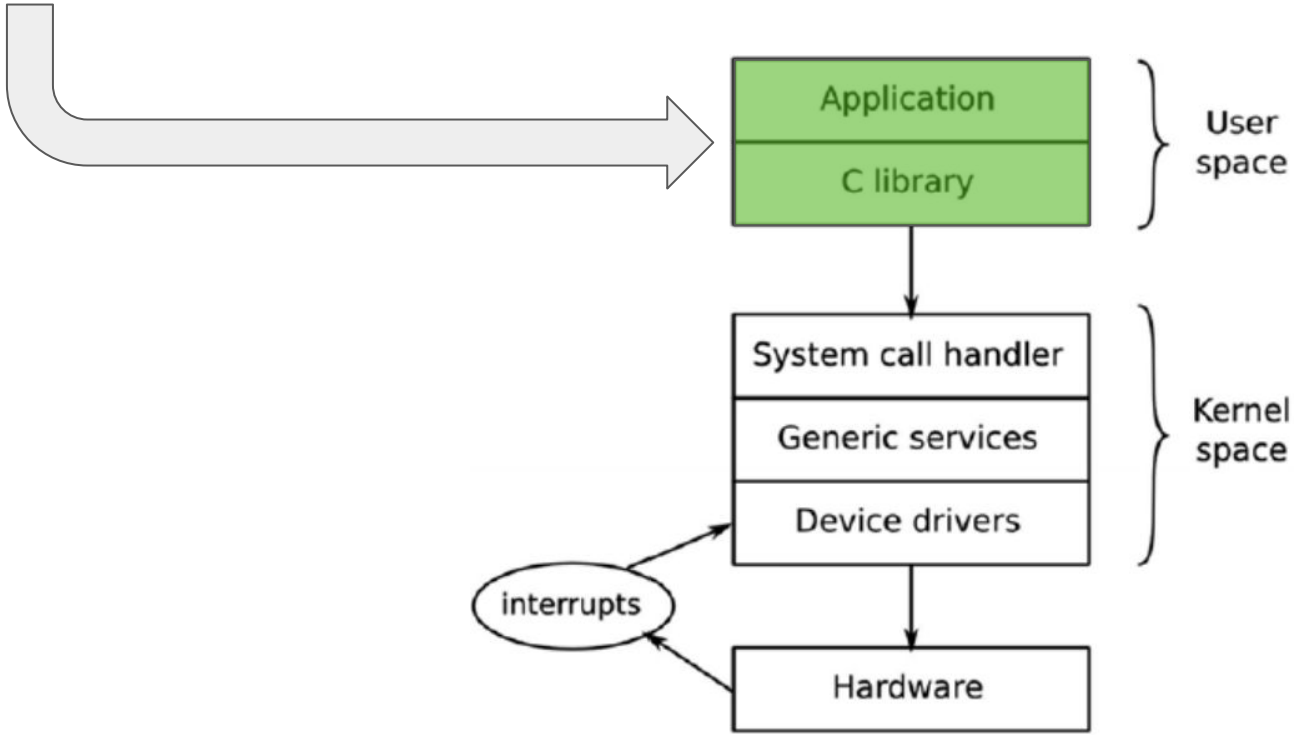
```
C Programming -> time python3 for.py

real    0m0,130s
user    0m0,126s
sys     0m0,004s
```

- Acceso Directo al Hardware
- Portabilidad y Estandarización
- Lenguaje de Uso Común en Microcontroladores
- Tamaño de Código Reducido
- Amplia Comunidad y Recursos
- Determinismo y Bajo Nivel de Abstracción
- Compatibilidad con Lenguaje Ensamblador
- Conocimiento y Experiencia Prevalentes

¿En dónde vamos a trabajar?



Aquí


Application
C library
User space
System call handler
Generic services
Device drivers
Kernel space
interrupts
Hardware

Pre requisitos

- Linux. Preferentemente **V.22**.
- Editor de texto. Por defecto ya debería estar instalado **VIM**. Podemos optar también por una versión más nueva llamada **neoVIM**.

  $ sudo apt-get install vim
  $ sudo apt install neovim

- Compilador de C. El más conocido es **gcc**.

  $ sudo apt-get install gcc

- Utilidad Make. Para automatizar el proceso de compilación mediante el **makefile**.

  $ sudo apt-get install make

- Debugger. **GDB** que nos sirve tanto para C como para C++.

  $ sudo apt-get install gdb

Estructura de un programa en C

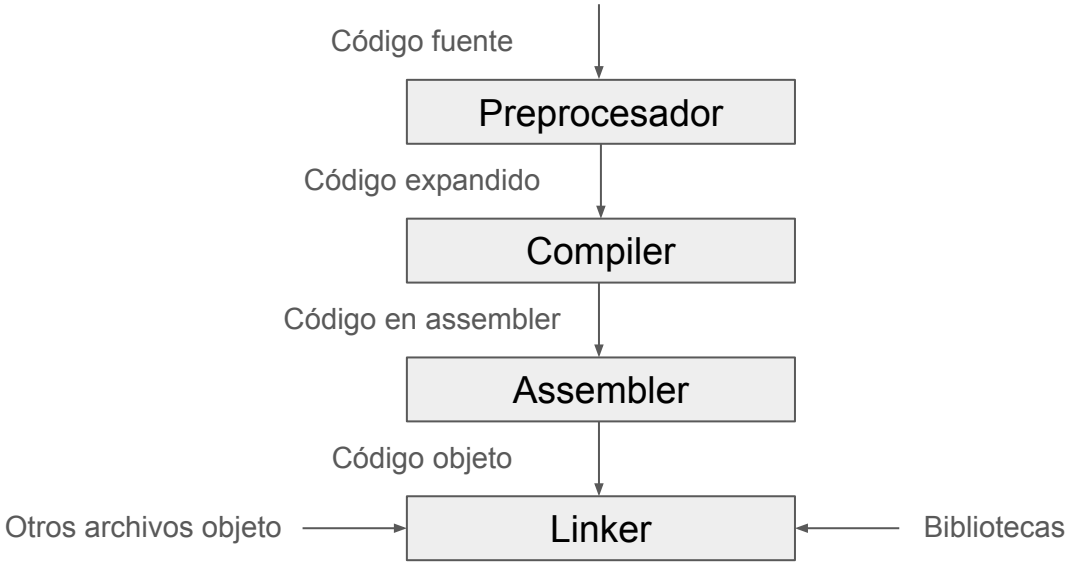
```
directivas del preprocesador
variables globales
declaración de funciones

int main(void){
        variables locales
        sentencias
        …
        …
        return 0;
}

func1() {
        variables locales
        sentencias
        …
}
```

Estructura de un programa en C

```c
/* Este es un
 * programa de prueba */

// También se puede comentar usando barras dobles.

// En esta sección se encuentrn las directivas del
// preprocesador.
#include <stdio.h>
#define LOOPS   1000000

// Acá podemos declarar variables globales:
long int i;

// También podemos declarar funciones:
void forfunc(void);

int main(){
        long int count = 0;     // Esta es una variable local.
        forfunc();              // Esta es una sentencia y la llamada a la función.
        return 0;
}

// Esta es la definición de la función:
void forfunc(void){
        long int forCount = 0;  // Variable local.
        for(i = 0; i < LOOPS; i++);     // Sentencias
                forCount++;
}
```

Compilación y ejecución de un programa en C

Existen diferentes fases a través de las cuales nuestro programa pasa antes de ser convertido en un ejecutable.



Compilación y ejecución de un programa en C

- **Preprocesador**:
  El código fuente es el código que escribimos usando cualquier editor de texto y el archivo de código fuente tiene la extensión '**.c**'. Este código fuente primero pasa por el **preprocesador**, quien expande este código y lo pasa al **compilador**.
- **Compilador**:
  El código expandido se pasa al **compilador**, que convierte el código en assembler.
- **Assembler**:
  En esta etapa se convierte el **lenguaje ensamblador** en **código objeto**. El nombre del archivo objeto es el mismo que el del archivo fuente. En UNIX la extensión es "**.o**".

Compilación y ejecución de un programa en C

- **Linker**:

  Generalmente todos los programas escritos en **C** utilizan funciones de biblioteca. Las funciones de biblioteca están precompiladas y su código objeto se almacena en archivos de biblioteca con extensión '**.lib**' (o '**.a**'). El **linker** combina este código objeto de las funciones de biblioteca con el código objeto de nuestro programa. Nuestro programa también puede contener referencias a funciones que están definidas en otros archivos. El **linker** enlaza el código objeto de estos archivos también a nuestro programa. Por lo tanto, el trabajo del **linker** es combinar el código objeto de nuestro programa con el código objeto de otros archivos y el código objeto de las funciones de biblioteca. La salida del **linker** es un archivo **ejecutable**. En UNIX, el archivo ejecutable se llama **a.out** o el nombre de la salida con la opción **-o**.

UNIVERSIDAD NACIONAL DE HURLINGHAM

Universidad Nacional de San Martín

Extra Slides

Leandro Gagliardi
lgagliardi@unsam.edu.ar