Facultad de Ingeniería en Electricidad y Computación

# Programación de Sistemas CCPG1051

Federico Domínguez, PhD.

Unidad 1 – Sesión 1: Introducción al curso

# Agenda

1. El Instructor
2. Los estudiantes
3. La asignatura: ¿Qué es programación de sistemas?
4. Políticas del Curso
5. Syllabus

# El Instructor

## Federico Domínguez, PhD

## Estudios

- Ingeniería en Computación
  - **ESPOL,** 2005
- Maestría en Ciencias Computacionales Aplicadas
  - Vrije Universiteit Brussel (**VUB**), 2009
- Doctorado en Ciencias Ingenieriles
  - Vrije Universiteit Brussel (**VUB**), 2014

# El Instructor

Profesor en la **FIEC**

Jefe de Laboratorio Tecnologías Ambientes Inteligentes (**TAI**) en el Centro de Tecnologías de Información (**CTI**)

Áreas de investigación:

- Internet de las Cosas
- Ambientes Inteligentes
- Redes sensoriales
- Sistemas Embebidos
- Monitoreo Ambiental

# Los estudiantes

# ¿Qué es programación de sistemas?

Programación de sistemas ≠ Programación de aplicaciones

Programación de aplicaciones

- Software para usuarios: sitios web, procesador de palabras, chat … etc.
- Fácil de usar, interface de usuario intuitiva y llamativa …
- Lenguajes de alto nivel: Java, Ruby, Python, PHP …

Programación de sistemas

- Software para otro software: servicio web, motor de juegos de video, sistemas operativos, drivers …
- Alto rendimiento, eficiente, usualmente no tiene interface de usuario, provee servicios …
- Lenguajes de bajo nivel: C, C++, Assembler

# ¿Qué es **Programación de Sistemas**?

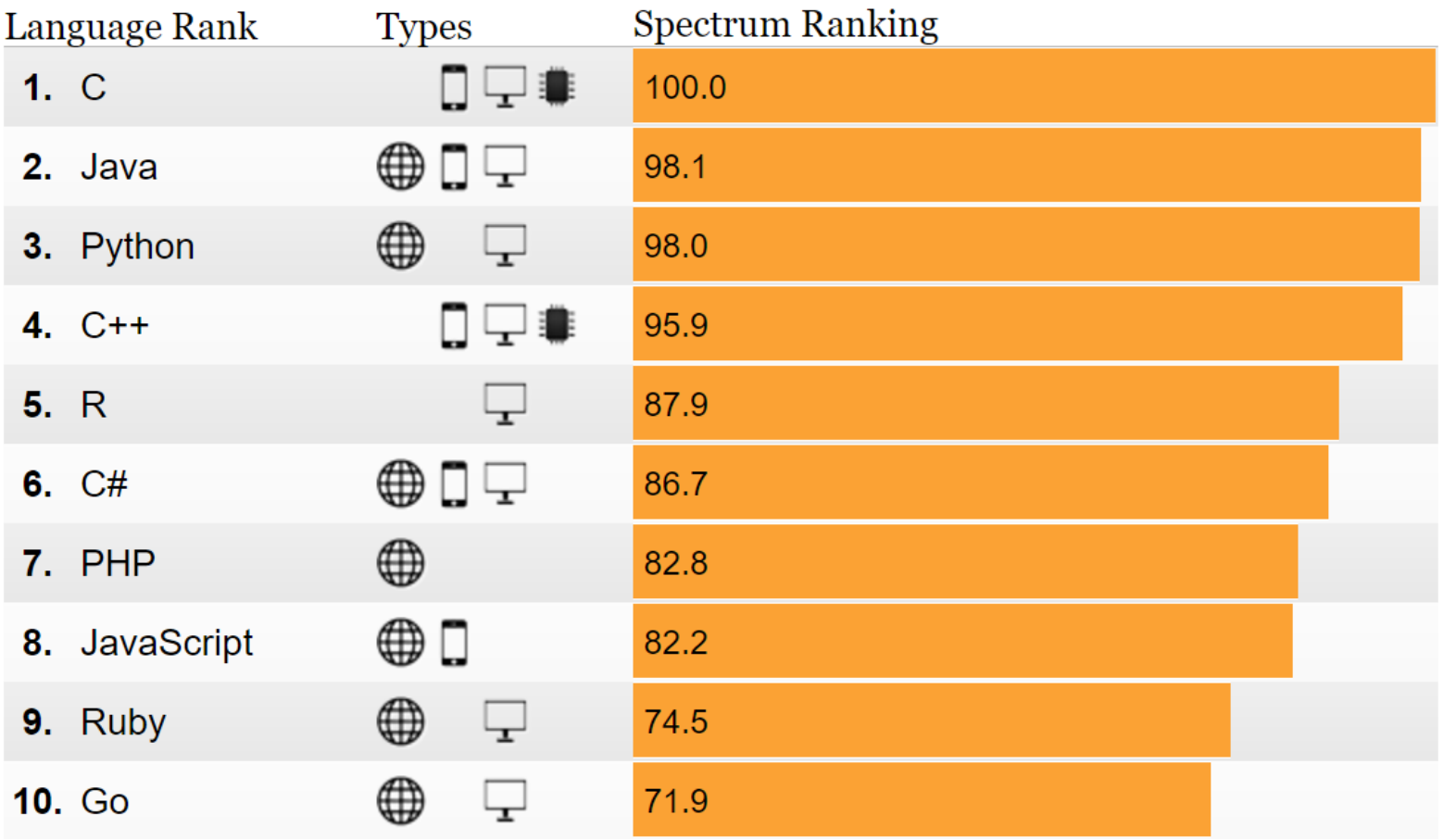
Objetivo general del Syllabus:

*Desarrollar software de bajo nivel computacional usando el lenguaje C y herramientas de gestión de código fuente para la interacción directa y eficaz con el sistema operativo y el hardware en sistemas basados en UNIX / LINUX.*

# ¿Qué es **Programación de Sistemas**?

El curso se enfocará en usar **C** (se mencionará C++) y **Linux** (se mencionará UNIX) para aplicar los conceptos de la programación de sistemas.

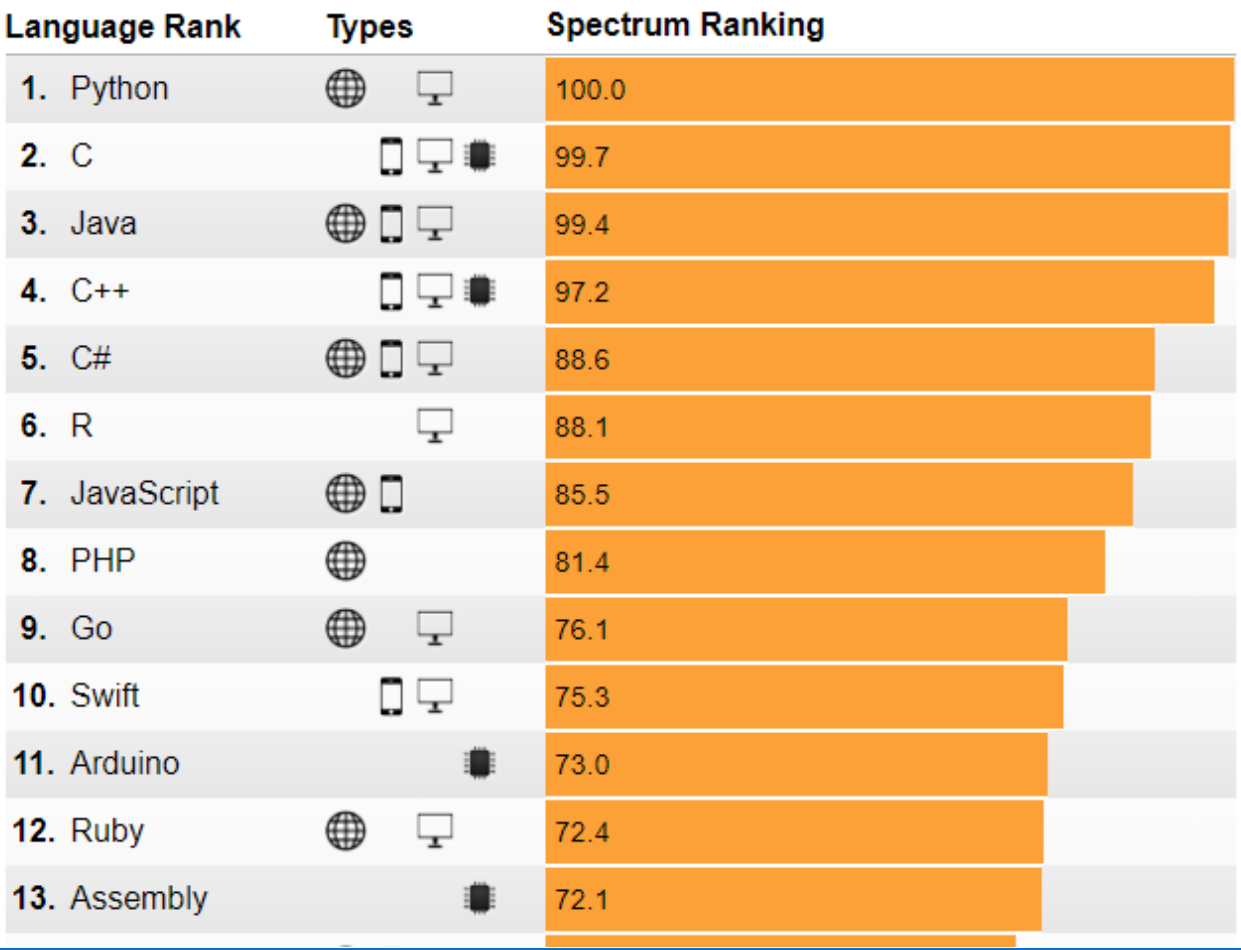
# ¿Por qué aprender C?




*C is used to write software where speed and flexibility is important, such as in embedded systems or high-performance computing.*

Fuente:
IEEE. The 2016 Top Programming Languages.
http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages

# Ranking en 2017

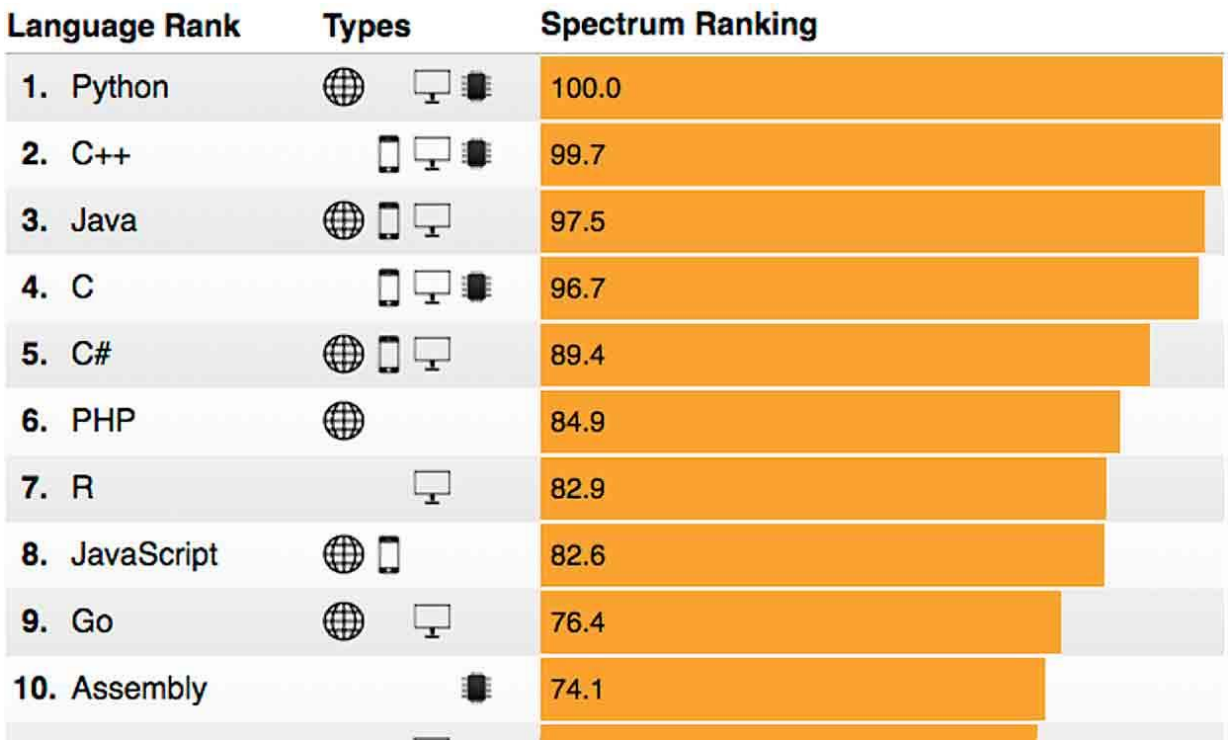


| Language Rank | Types | Spectrum Ranking |
|---|---|---|
| 1. Python | | 100.0 |
| 2. C | | 99.7 |
| 3. Java | | 99.4 |
| 4. C++ | | 97.2 |
| 5. C# | | 88.6 |
| 6. R | | 88.1 |
| 7. JavaScript | | 85.5 |
| 8. PHP | | 81.4 |
| 9. Go | | 76.1 |
| 10. Swift | | 75.3 |
| 11. Arduino | | 73.0 |
| 12. Ruby | | 72.4 |
| 13. Assembly | | 72.1 |

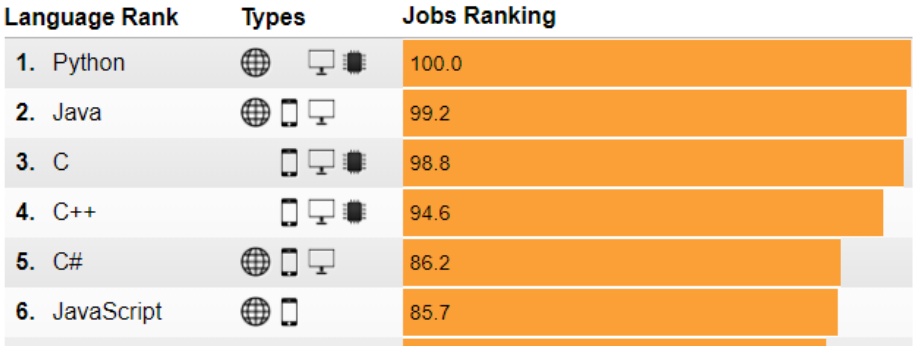
*Although Python has moved to the top of the default Spectrum ranking, if we instead go purely by the volume of openings that mention a language, we find that C beats Python by a ratio of 3.5 to 1 …*

Fuente:
IEEE. The 2017 Top Programming Languages. https://spectrum.ieee.org/computing/software/top-programming-languages-2017-focus-on-jobs

# Ranking en 2018



| Language Rank | Types | Spectrum Ranking |
|---|---|---|
| 1. Python | | 100.0 |
| 2. C++ | | 99.7 |
| 3. Java | | 97.5 |
| 4. C | | 96.7 |
| 5. C# | | 89.4 |
| 6. PHP | | 84.9 |
| 7. R | | 82.9 |
| 8. JavaScript | | 82.6 |
| 9. Go | | 76.4 |
| 10. Assembly | | 74.1 |



| Language Rank | Types | Jobs Ranking |
|---|---|---|
| 1. Python | | 100.0 |
| 2. Java | | 99.2 |
| 3. C | | 98.8 |
| 4. C++ | | 94.6 |
| 5. C# | | 86.2 |
| 6. JavaScript | | 85.7 |

Fuente:
IEEE. The 2018 Top Programming Languages.
https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

# Ranking en 2019

| Rank | Language | Type | Score |
|---|---|---|---|
| 1 | Python | web, desktop, system | 100.0 |
| 2 | Java | web, mobile, desktop | 96.3 |
| 3 | C | mobile, desktop, system | 94.4 |
| 4 | C++ | mobile, desktop, system | 87.5 |
| 5 | R | desktop | 81.5 |
| 6 | JavaScript | web | 79.4 |
| 7 | C# | web, mobile, desktop, system | 74.5 |

Fuente:
IEEE. The 2019 Top Programming Languages.
https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019

# Ranking en 2020

**Language Ranking: IEEE Spectrum**

| Rank | Language | Type | Score |
|---|---|---|---|
| 1 | Python | | 100.0 |
| 2 | Java | | 95.3 |
| 3 | C | | 94.6 |
| 4 | C++ | | 87.0 |
| 5 | JavaScript | | 79.5 |
| 6 | R | | 78.6 |
| 7 | Arduino | | 73.2 |

**Language Ranking: Jobs**

| Rank | Language | Type | Score |
|---|---|---|---|
| 1 | Python | | 100.0 |
| 2 | C | | 98.0 |
| 3 | Java | | 97.1 |
| 4 | Go | | 87.2 |
| 5 | C++ | | 85.2 |
| 6 | JavaScript | | 81.8 |
| 7 | R | | 80.4 |

Fuente:
IEEE. The 2020 Top Programming Languages. https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020

# “Hello World” en C

```
/* Hello World program */

#include<stdio.h>

int main()
{
   printf("Hello World\n");
}
```

# ¿Qué hace esto?

```
void show_squares()
{
    int x;
    for (x = 5; x <= 5000000; x = x*10)
        printf("x = %d x^2 = %d\n", x, x*x);
}
```

Explosión Ariane 5: https://www.youtube.com/watch?v=kYUrqdUyEpI

# Políticas del curso

Usaremos herramientas como **Zoom** o **Microsoft Teams** para las sesiones de clases en línea.

Las clases serán grabadas, se proporcionará acceso a los videos mediante enlaces en **SIDWEB**.

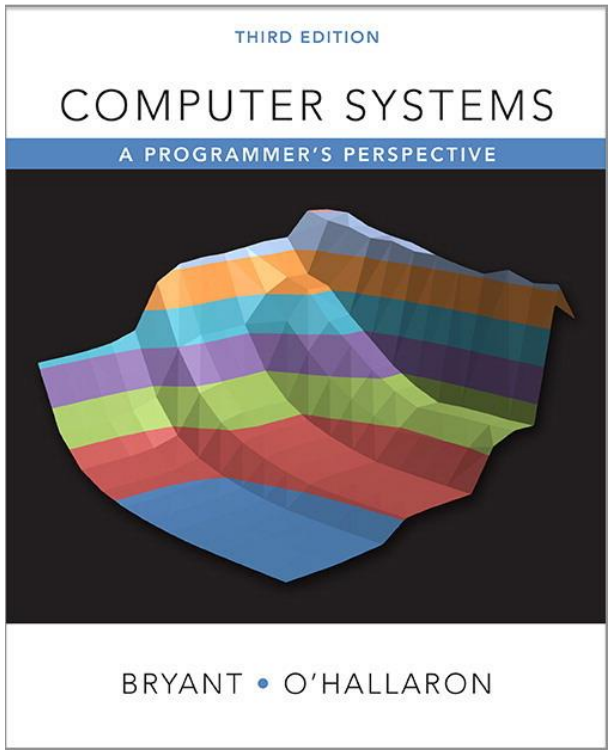
Se usará el SIDWEB para distribución de material de aprendizaje, tareas, calificaciones, etc.

# Políticas del curso

## Libro guía

Computer Systems: A Programmer's Perspective, 3/E (CS:APP3e)

Randal E. Bryant and David R. O'Hallaron, Carnegie Mellon University



# Políticas del curso

| COMPONENTE | ACTIVIDAD | 1era. Evaluación | 2da. Evaluación | 3era. Evaluación |
|---|---|---|---|---|
| EHD / Teórico 35% | Examen | n.a. | 30% | 70% |
| | Lecciones | 10%+25% | 5% | |
| EHTA / Autónomo 35% | Tareas | 35% | 35% | |
| EHP / Práctico 30% | Talleres | 30% | | n.a. |

# Syllabus: Objetivos

Resultados de aprendizaje:

1. Construir un programa simple en C usando métodos de división de capas, detección de errores y reflexión de estatus de errores para la creación de un sistema robusto y de mínimo mantenimiento.
2. Implementar programas con paralelismo computacional usando eventos, hilos, procesos y otros paradigmas de concurrencia para el uso eficiente de los recursos provistos por el hardware y el sistema operativo de un computador.
3. Implementar una aplicación cliente-servidor simple usando sockets y un API básico para la creación de un sistema escalable con clara separación de competencias.
4. Programar un sistema computacional usando un paradigma orientado a eventos para la gestión de eventos asincrónicos externos.
5. Usar herramientas de colaboración de software, depuración e integración para la gestión en equipo del desarrollo de productos de software de mediano tamaño.

# Syllabus: Unidades

1. Introducción a Linux y C

*En esta unidad se revisan los conceptos básicos de sistemas operativos UNIX / LINUX, del funcionamiento del interpretador de comandos de estos sistemas (conocido como shell) y del lenguaje C.*

2. Compiladores y Herramientas de programación

*En esta unidad se cubre la cadena de herramientas de C y C++, desde los entornos de desarrollo integrado (IDE) más usados y los compiladores hasta las herramientas de depuración y versionamiento.*

# Syllabus: Unidades

3. Representación de Datos y Gestión de Memoria

*En esta unidad se revisan los conceptos de gestión de memoria en el lenguaje C desde el uso básico de punteros hasta el uso de la herramienta GNU Debugger (GDB) para depuración avanzada de código fuente. El estudiante aprende los errores comunes en la gestión de memoria y técnicas para depurarlos.*

4. Librerías

*En esta unidad se revisa el uso y creación de librerías estáticas y dinámicas en el lenguaje C. Se enfatiza además el uso de librerías como una forma para distribuir y compartir soluciones computacionales a terceros.*

**FIN PRIMER PARCIAL**

# Syllabus: Unidades

5. Entrada/Salida

*En esta unidad se revisa la arquitectura cliente - servidor como un patrón de diseño básico para aplicaciones distribuidas. Se utilizan llamadas del sistema de entrada/salida para comunicación entre procesos usando sockets y archivos, creando un sistema distribuido.*

6. Programación en Paralelo

*En esta unidad se revisan los conceptos de programación que usan concurrencia. Específicamente se exploran tres paradigmas de concurrencia a bajo nivel computacional: eventos, procesos e hilos. Además se utilizan patrones de diseño que permiten explotar el paralelismo computacional de una manera robusta y eficiente.*