



Programación de Sistemas

CCPG1008

Federico Domínguez, PhD.

Unidad 1 – Sesión 1: Introducción al curso

Contenidos

1. El Instructor
2. Los estudiantes
3. La asignatura: ¿Qué es programación de sistemas?
4. Políticas del Curso
5. Syllabus
6. Introducción a la arquitectura de Linux/Unix

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

Docente en la FIEC

Investigador en el Centro de Tecnologías de Información (**CTI**)

Coordinador del Laboratorio de Prototipado Rápido

Áreas de investigación: sistemas embebidos, redes sensoriales,
Internet de las Cosas, monitoreo ambiental ...

Los estudiantes

¿Qué es programación de sistemas?

Programación de sistemas \neq 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 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?



























Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9

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>

“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);
}
```



Políticas del curso

La asistencia a clases es obligatoria, se tomará lista cada clase. No se considerará como asistencia después de 30 minutos empezada la clase.

Si después de 30 minutos no se presenta el profesor, no se realizará clase ese día.

En lo posible se usara el SIDWEB para distribución de material de aprendizaje, tareas, calificaciones, etc.

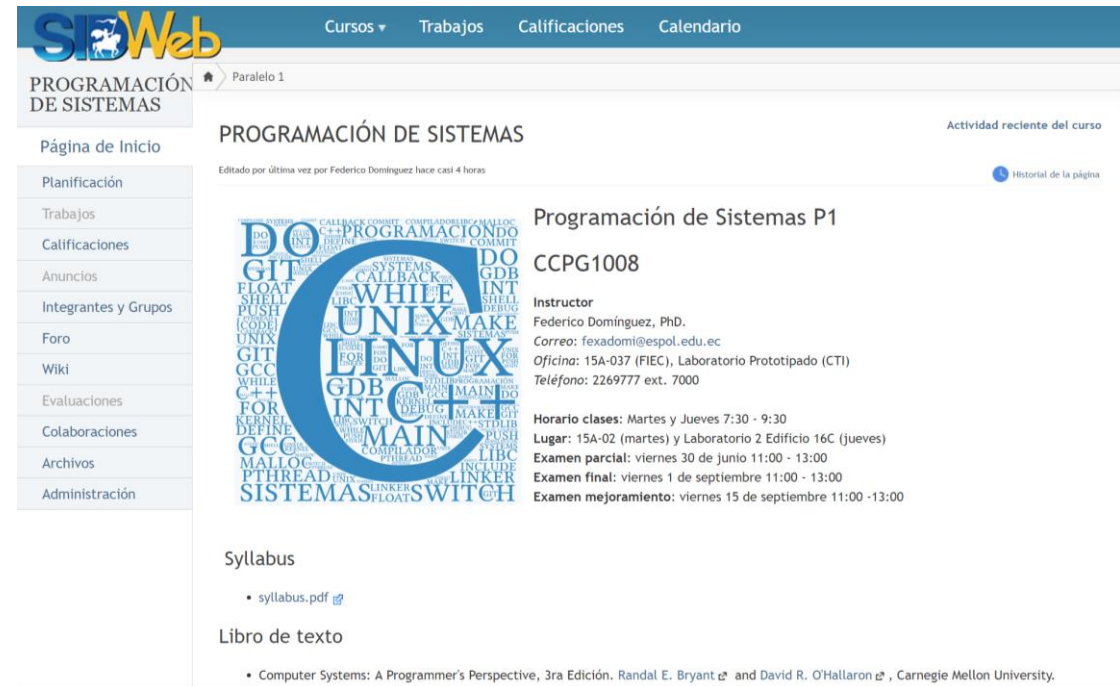
Políticas del curso

Se utilizará SIDWEB

- www.sidweb.espol.edu.ec

Podrá encontrar:

- Diapositivas
- Libro digital
- Recursos
- Otros recursos interesantes



The screenshot shows the SIDWEB interface for the course 'PROGRAMACIÓN DE SISTEMAS'. The top navigation bar includes links for 'Cursos', 'Trabajos', 'Calificaciones', and 'Calendario'. The course title is 'PROGRAMACIÓN DE SISTEMAS' under 'Paralelo 1'. A sidebar on the left lists various course activities: 'Página de Inicio', 'Planificación', 'Trabajos', 'Calificaciones', 'Anuncios', 'Integrantes y Grupos', 'Foro', 'Wiki', 'Evaluaciones', 'Colaboraciones', 'Archivos', and 'Administración'. The main content area displays the course title, a word cloud graphic, and details for 'Programación de Sistemas P1' (CCPG1008). The instructor is Federico Domínguez, PhD., with contact information for email, office, and phone. It also lists class times, location, and exam dates. At the bottom, there are links for the syllabus and a digital book.

PROGRAMACIÓN DE SISTEMAS

Paralelo 1

PROGRAMACIÓN DE SISTEMAS

Actividad reciente del curso

Historial de la página

Programación de Sistemas P1

CCPG1008

Instructor
Federico Domínguez, PhD.
Correo: feadomi@espol.edu.ec
Oficina: 15A-037 (FIEC), Laboratorio Prototipado (CTI)
Teléfono: 2269777 ext. 7000

Horario clases: Martes y Jueves 7:30 - 9:30
Lugar: 15A-02 (martes) y Laboratorio 2 Edificio 16C (jueves)
Examen parcial: viernes 30 de junio 11:00 - 13:00
Examen final: viernes 1 de septiembre 11:00 - 13:00
Examen mejoramiento: viernes 15 de septiembre 11:00 - 13:00

Syllabus

- [syllabus.pdf](#)

Libro de texto

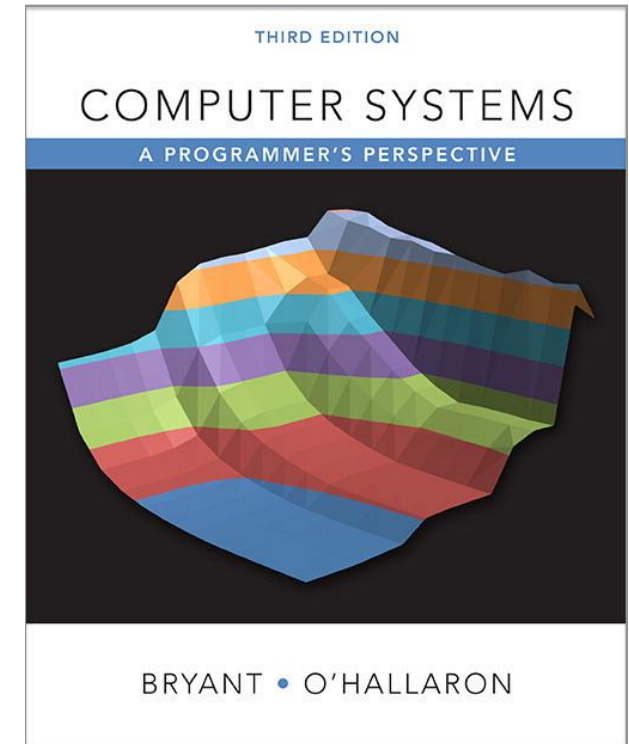
- Computer Systems: A Programmer's Perspective, 3ra Edición. Randal E. Bryant and David R. O'Hallaron, Carnegie Mellon University.

Políticas del curso

Libro de texto

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

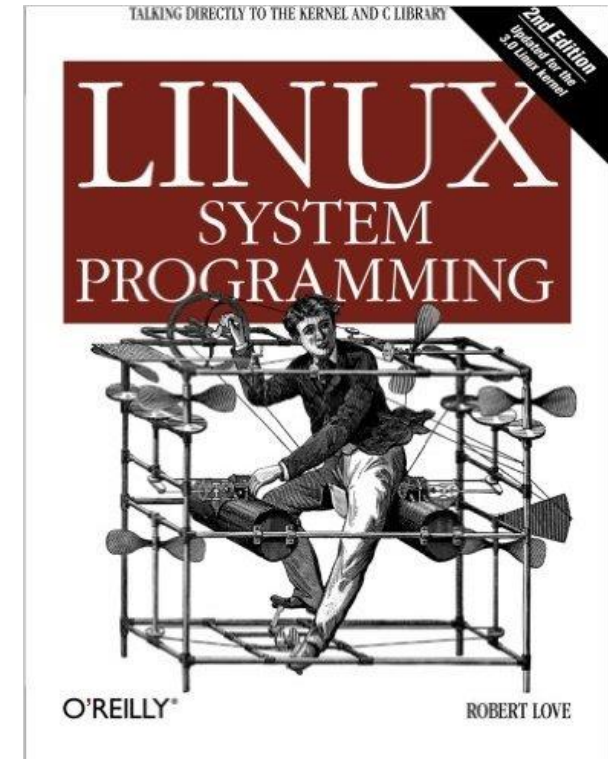
[Randal E. Bryant](#) and [David R. O'Hallaron](#),
Carnegie Mellon University



Políticas del curso

Libros guías

- LINUX System Programming, 2/E. Robert Love
- The Linux Command Line, 3/E. William Shotts



Políticas del curso

Calificación

- 50% Examen
- 20% Proyecto (un proyecto por parcial, 2 personas)
- 10% Lecciones (libro abierto, en SIDWEB al final de cada unidad, ~3 unidades por parcial)
- 20% Laboratorios (una práctica cada semana)

Carga de trabajo: 9 horas por semana

- Distribuidas en: 3 horas docencia, 1 hora práctica, 5 horas trabajo autónomo

Syllabus: Objetivos

Objetivos 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 asíncronos 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 al Shell 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 cubrirá la cadena de herramientas de C y C++, desde los IDEs 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 revisarán los conceptos de gestión de memoria en el lenguaje C desde el uso básico de punteros hasta el uso de la herramienta GDB para depuración avanzada de código fuente. El estudiante aprenderá los errores comunes en la gestión de memoria y técnicas como depurarlos.

EXAMEN PARCIAL

4. Librerías

En esta unidad se revisa el uso y creación de librería 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.

Syllabus: Unidades

5. Entrada/Salida

En esta unidad se revisará la arquitectura cliente - servidor como un patrón de diseño básico para aplicaciones distribuidas. El estudiante aprenderá a usar llamadas del sistema de entrada/salida para comunicación entre procesos usando sockets y archivos, creando un sistema distribuido.

6. Programación en Paralelo

En esta sección se revisarán los conceptos de programación usando concurrencia. Específicamente se exploran tres paradigmas de concurrencia a bajo nivel: eventos, procesos e hilos. Además el estudiante aprenderá a usar patrones de diseño que permiten explotar el paralelismo computacional de una manera robusta y eficiente.

7. Sistemas de bajo nivel

En esta unidad se revisará la creación e instalación de módulos del kernel de LINUX. Se usará como ejemplo la creación de un driver simple para un sistema embebido.

Introducción a la arquitectura de Linux/Unix

Breve historia de UNIX

- 1970: Creado en Bell Labs por Ken Thompson, Dennis Ritchie, Doug McIlroy y Joe Ossanna.
- 1973: Dennis Ritchie inventa el lenguaje **C** y lo usa para reescribir UNIX.
- 1980s: UNIX se hace popular en universidades y se crean numerosas versiones.
 - Unix 4.xBSD (Berkeley Software Distribution)
 - System V Unix (Bell Labs)
 - Solaris (Sun Microsystems)
 - AIX (IBM)
 - ...
- 1988: IEEE crea IEEE Std 1003.1-1988 -- mejor conocido como *Portable Operating System Interface* (**POSIX**) -- para estandarizar todas las distribuciones de UNIX.
- Actualidad: Usado ampliamente como sistema operativo de servidores comercial por IBM, Oracle, HP y otros. FreeBSD es una versión código abierto de UNIX.

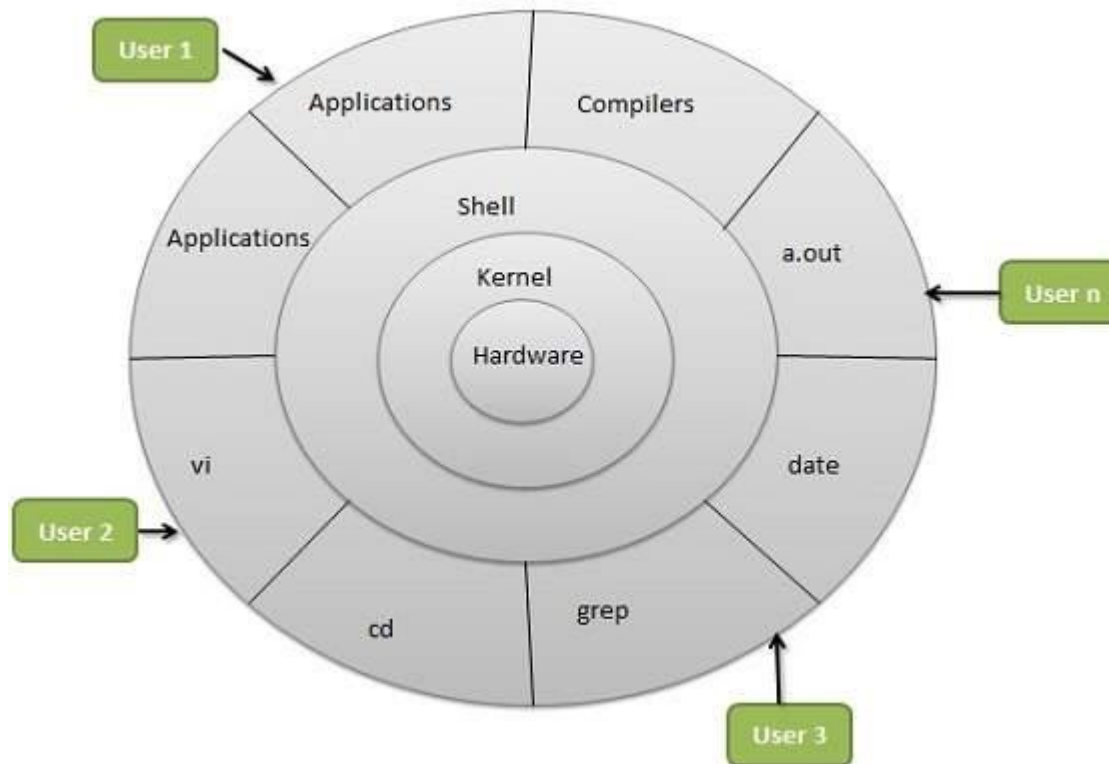
Introducción a la arquitectura de Linux/Unix

Breve historia de LINUX

- 1991: Linus Torvalds crea, como hobby, un sistema operativo código abierto basado en MINIX (una versión educativa de UNIX).
- Un juego de palabras entre UNIX y Linus, LINUX técnicamente se refiere tan solo al *kernel* del sistema operativo. LINUX se fusiona con el proyecto GNU de Richard Stallman para crear un sistema operativo completo.
- En lo posible, el *kernel* de LINUX y el software que lo acompaña es POSIX, manteniendo compatibilidad con UNIX.
- Software libre: El *kernel* de LINUX y varios otros componentes del sistema operativo usan la licencia GNU General Public License (GPL).
- Actualidad: Extremadamente popular, es el sistema operativo más usado. Desde supercomputadoras hasta relojes, el *kernel* de Linux está instalado en miles de millones de dispositivos en todo el mundo.

Introducción a la arquitectura de Linux/Unix

Arquitectura de LINUX



Hardware: CPU y periféricos

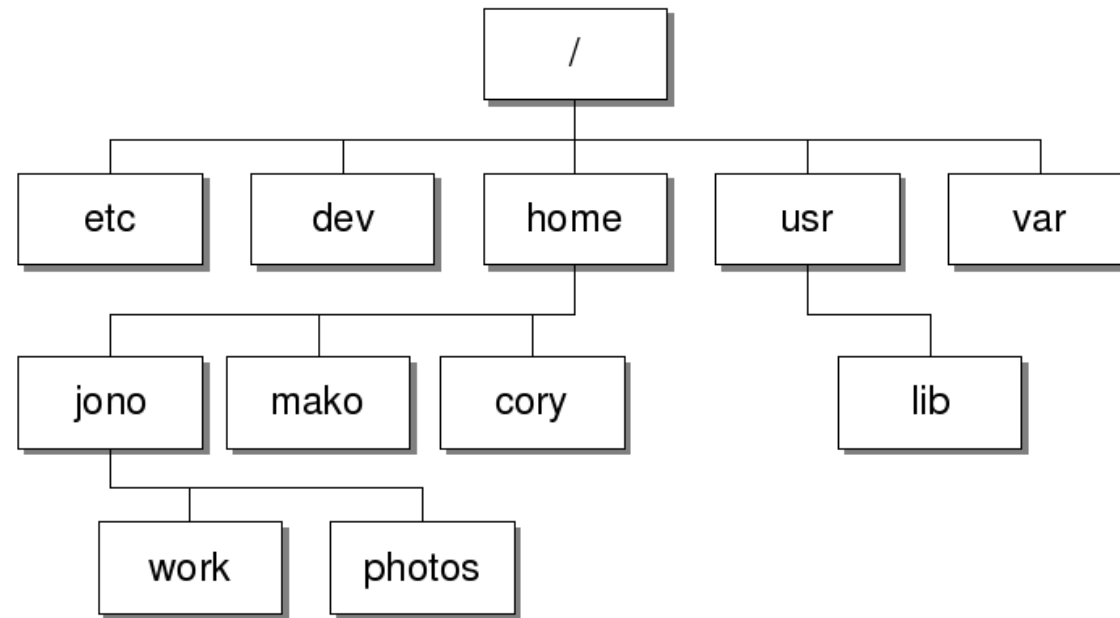
Kernel: Núcleo del sistema operativo, controla todo el funcionamiento del sistema.

Shell: Interface entre el Kernel y las aplicaciones de usuario. Esconde la complejidad del Kernel, ejecuta comandos de usuarios y aplicaciones.

Introducción a la arquitectura de Linux/Unix

Sistema de archivos

- Es monolítico, empieza desde la raíz o “root” representada por “/”



Para la siguiente sesión ...

Introducción al Shell de Linux:

- Lectura: The Linux Command Line (TLCL) capítulos 1 – 4
- Opcional pero recomendado: Instalar Linux usando VirtualBox o directamente en un PC.