Programación Sobre Redes

T0: Introducción

Nicolás Mastropasqua

March 10, 2020

Instituto Industrial Luis A. Huergo

Contenidos

1. Introducción a la materia
2. Sistemas operativos: Primeras nociones
3. Sistemas operativos: Evolución

Introducción a la materia

Bienvenidos

¿De qué se trata esto?

Sistemas operativos: Primeras nociones

¿Qué son?

- Uno de los objetivos es **hacerle la vida más fácil** al usuario.
- Los programas no necesitan conocer los detalles técnicos del hardware subyacente.
- Desde el lado del fabricante de Hardware, se asegura el correcto uso del mismo.
- Gestiona recursos y brinda servicios para los programas

¿Que compone a un Sistema Operativo?

Definición simplista

Es todo lo que viene cuando lo adquiero

Figure 1: IE como parte de Windows

¿Qué compone a un Sistema Operativo?

Otra definición

Es únicamente ”el programa que está corriendo todo el tiempo” (**Kernel**), siendo todo lo demás programas del sistema o programas de aplicación

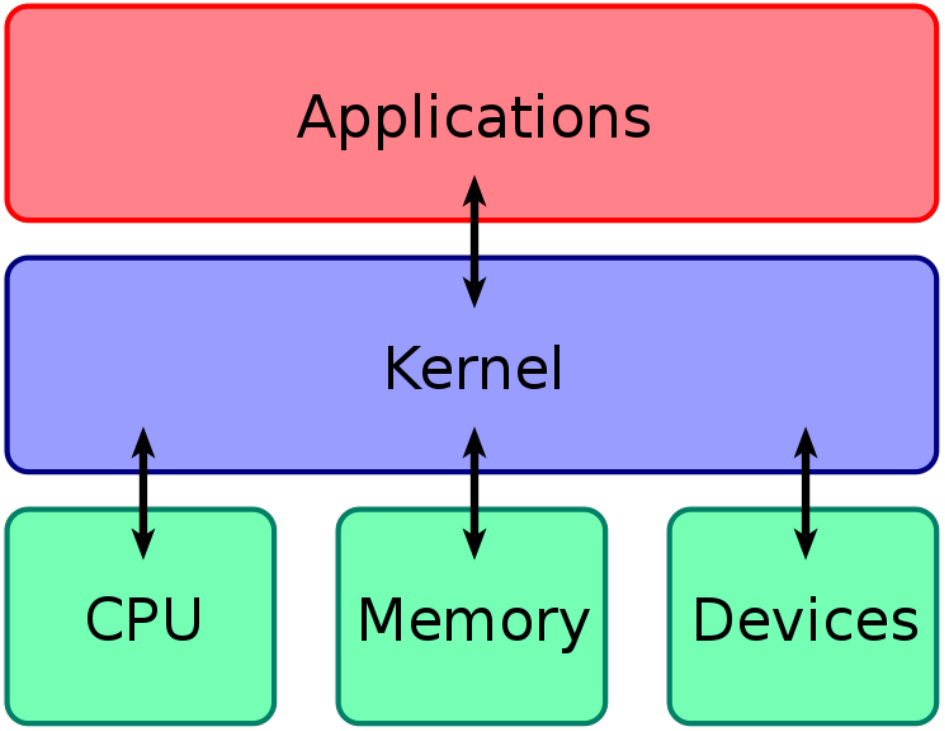


Figure 2: Esquema de interacción del Kernel

¿Y qué es parte del Kernel?

Distintos enfoques

- Monolito
- Capas
- Micro Kernel
- Modulos

Ejemplos: Kernel Monolito

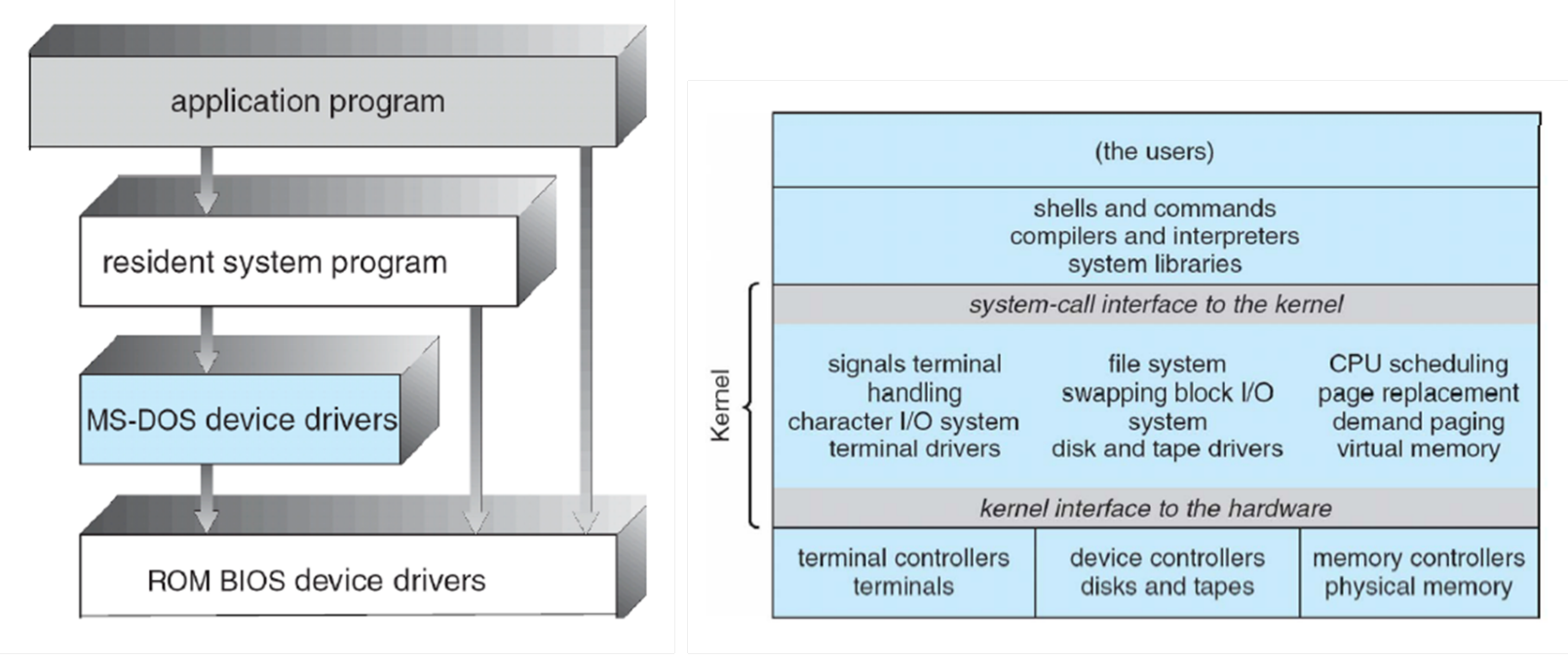


Figure 3: Esquemas de kernel MSDOS (izquierda) y UNIX(derecha)

Ejemplo: Kernel modular

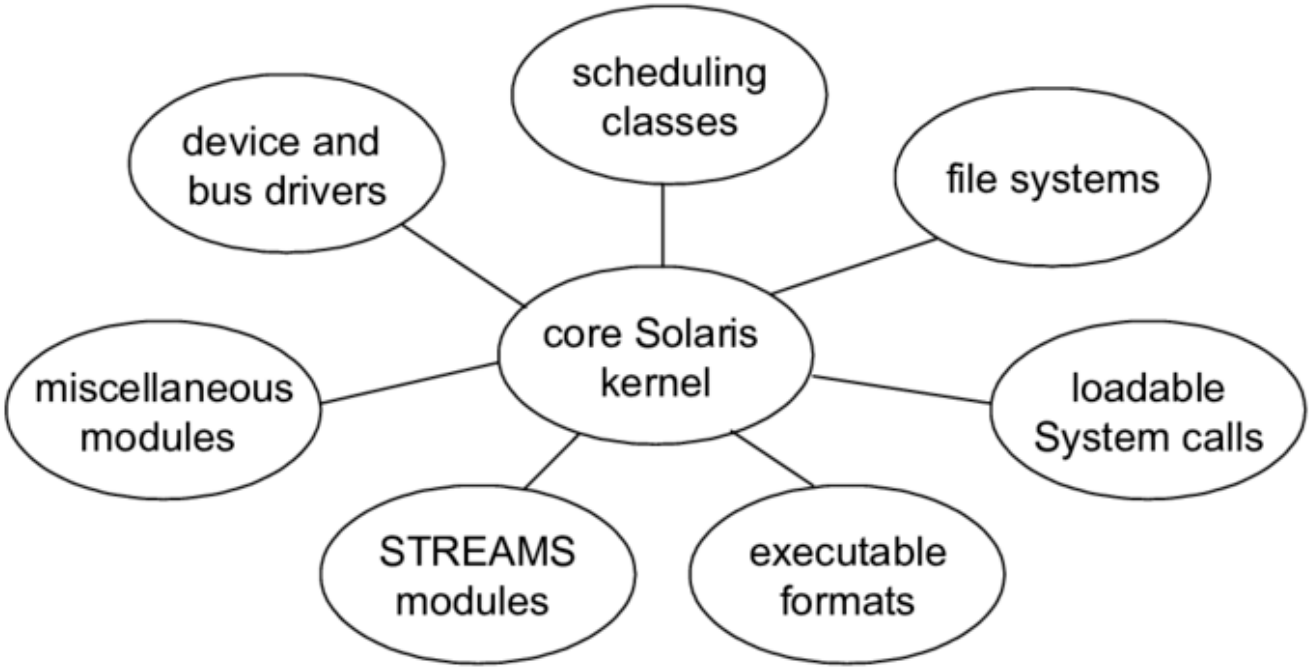


Figure 4: Esquema módulos del kernel Solaris

Sistemas operativos: Evolución


UN MEME
UNA PERSONA FELIZ

Evolución...

Figure 5: Richard Stallman, impulsor del proyecto GNU, a través de los años

Sistemas Batch

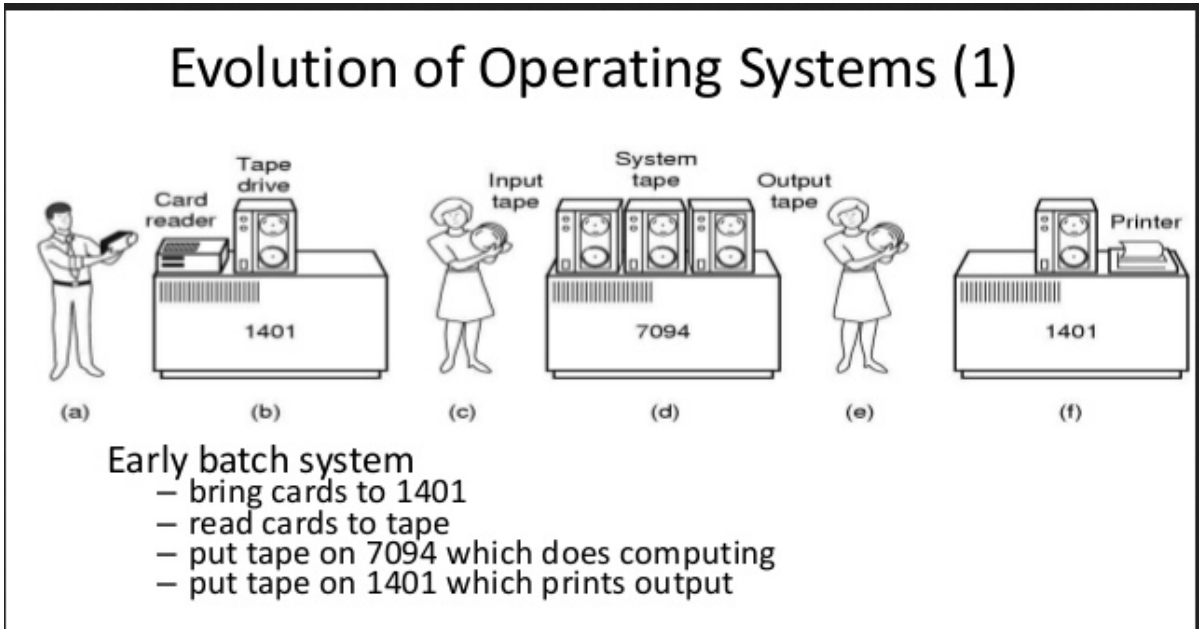


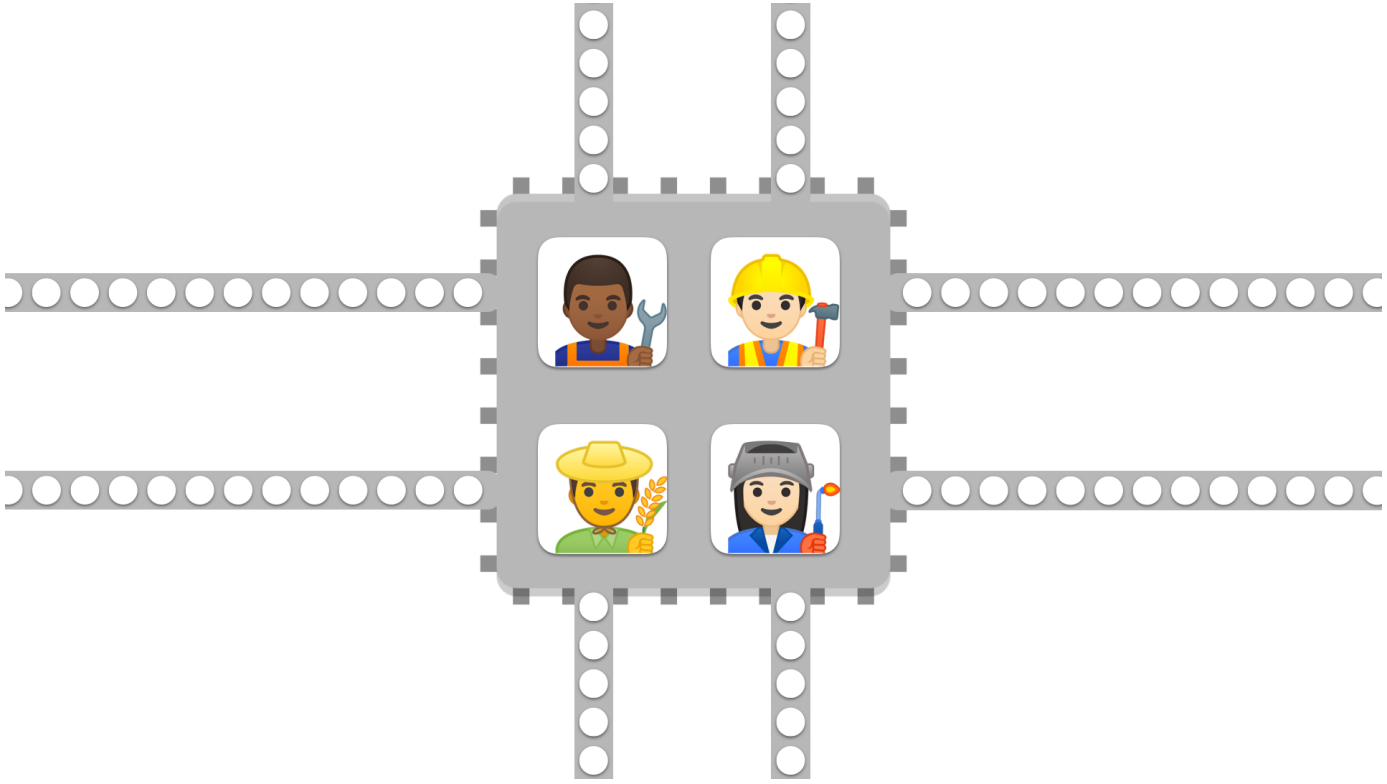
Figure 6: Sistema Batch

Sistemas batch

- Conocidos así porque no se los manejaba interactivamente como hoy, sino que los programas se cargaban en tandas (cada cinta).
- Quién o qué hacía de intermediario entre usuario/HW? El operador.

Problema

- ¿Sería posible ejecutar más de una tarea al mismo tiempo con un **único procesador**?
- ¿Y para el caso de un **multicore**?



Multitasking? Interactividad?

Figure 7: Caption

Problema

Cómo lograr la noción de simultaneidad de ejecución de tareas en una sistema single core

Multiprogramación: Camino al Multitasking

Concepto utilizado en sistemas operativos más viejos. Inicialmente pensado para sistemas multiusuario.

Esquema general de funcionamiento

- Varios trabajos son cargados en memoria.
- El procesador toma un trabajo de la memoria.
- Comienza la ejecución del mismo hasta que este necesite esperar por una tarea (por ejemplo, una operación de E/S).
- En ese momento, se intercambia el trabajo con otro de la memoria.
- El proceso continúa. Mientras exista un trabajo el procesador no queda IDLE.

Observaciones

- Aumenta el throughput.
  - El trabajo j_1 toma el mismo tiempo que antes, **o incluso un poco mas**
  - Pero, ahora $j_1 + j_2$ tarda menos
- Sigue sin existir interacción entre el usuario y el sistema
- Surgen problemas de contención de recursos y de protección.

Ejemplos

Pizarrón

Preemptive Multitasking

- Es más general que multiprogramación.
- Genera la ilusión de que más de una tarea se ejecuta **al mismo tiempo**.
- Un sistema que es **preemptive** puede interrumpir la tarea actual, suspendiéndola y luego tomando **la decisión de cual es la próxima a ejecutar**.
- Surge la necesidad de **Cpu Scheduling** (próximamente)

Preemptive Multitasking

Un esquema general

- El CPU recibe una tarea y la ejecuta durante un tiempo determinado, llamado **quantum**.
- Cuando el **quantum** de una tarea se termina, aunque esta no haya finalizado, y si existen tareas esperando, se le asigna al CPU **la próxima tarea** a ejecutar durante otro **quantum**.
- Una forma de definir la próxima es acomodando las tareas por orden de llegada
- ¿Cómo se puede implementar el quantum?

Pizarrón

Multiprogramación vs Multitasking

- Con multiprogramación las tareas se ejecutan hasta bloquearse (por una E/S por ejemplo).
- Si la tarea es **cpu bounded**, entonces la misma acaparará el CPU durante toda su duración invalidando al resto. Esto no ocurre si tenemos un sistema preemptive que pueda desalojarla.
- Si hay preemtion, podemos lograr multitasking, dándonos la noción de **simultaneidad** en la ejecución de tareas, incluso en sistemas single core.
- En el caso anterior, se dice que hay concurrencia.
  No confundir con paralelismo!

Bibliografía

- *Abraham Silberschatz, Peter Baer Galvin, Greg Gagne.* ***Operating System Concepts Seventh Edition***
  - Multiprocesador: Secciones 1.3 a 1.3.2
  - Multiprogramación: Sección 1.4
  - Kernels: Secciónes 2.7 a 2.7.4 incluídas

Dudas?