Sistemas Operativos

Introducción

2021-2022

Bibliografía:

Sistemas operativos. J. Carretero el at. Tema 2.

Operating Systems: Three Easy Pieces (https://pages.cs.wisc.edu/~remzi/OSTEP/). Introduction

Indice

1. ¿Qué es un SO? (Carretero 2.1 y 2.6)
2. Componentes del SO (Carretero 2.1)
3. Concepto de llamada al sistema (Carretero 2.5)
4. Arranque del SO (Carretero 2.4)
5. El shell Bash. Introducción a Bash scripting (Carretero 2.7)

   LABORATORIO.

   Machtelt Garrels. Bash Guide for Beginners. 2008.
   http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf

0. Conocimientos previos

- Aspectos arquitectónicos y estructurales de los computadores que afectan al sistema operativo:
  - Arquitectura von Newmann y ejecución de instrucciones.
  - Mecanismos de protección.
  - Modos de ejecución.
  - Interrupciones y excepciones. Temporizadores.
  - Jerarquía de memoria.
  - Entrada/salida. Dispositivos de bloques y caracteres.
  - Mecanismos de E/S: programada, mediante interrupciones y por DMA.

Modos de ejecución del procesador

- El **procesador** ofrece un **conjunto de modos de ejecución** para proporcionar **distintos niveles de acceso a los recursos** de la máquina
- Cada modo de ejecución está caracterizado por:
  - Subconjunto de instrucciones del repertorio disponibles
  - Acceso al mapa de E/S
  - Acceso a los registros de soporte de gestión de memoria (config. MMU)
  - Permiso de cambio de modo (Bits del registro de estado)
- En GNU/Linux, **el núcleo del SO se ejecuta en el modo menos restrictivo** (“modo kernel”) y los **programas de usuario en el más restrictivo** ("modo usuario")
- Las arquitecturas x86 (Intel y AMD) ofrecen 4 niveles de privilegio (*ring levels*): 0, 1, 2 y 3.
  - Modo usuario: ring level 3
  - Modo kernel: ring level 0

1. ¿Qué es un sistema operativo?

- Un software (normalmente en C, pero algunos en C++).
- Tiene dos objetivos:
  - Simplificar: **Interfaz entre el usuario y el computador**.
    - VIRTUALIZA CPU, memoria y dispositivos.
    - Conceptos clave: proceso, fichero.
  - Optimizar: **Gestor de los recursos del computador**.
    - Uso eficiente.
    - Protección.

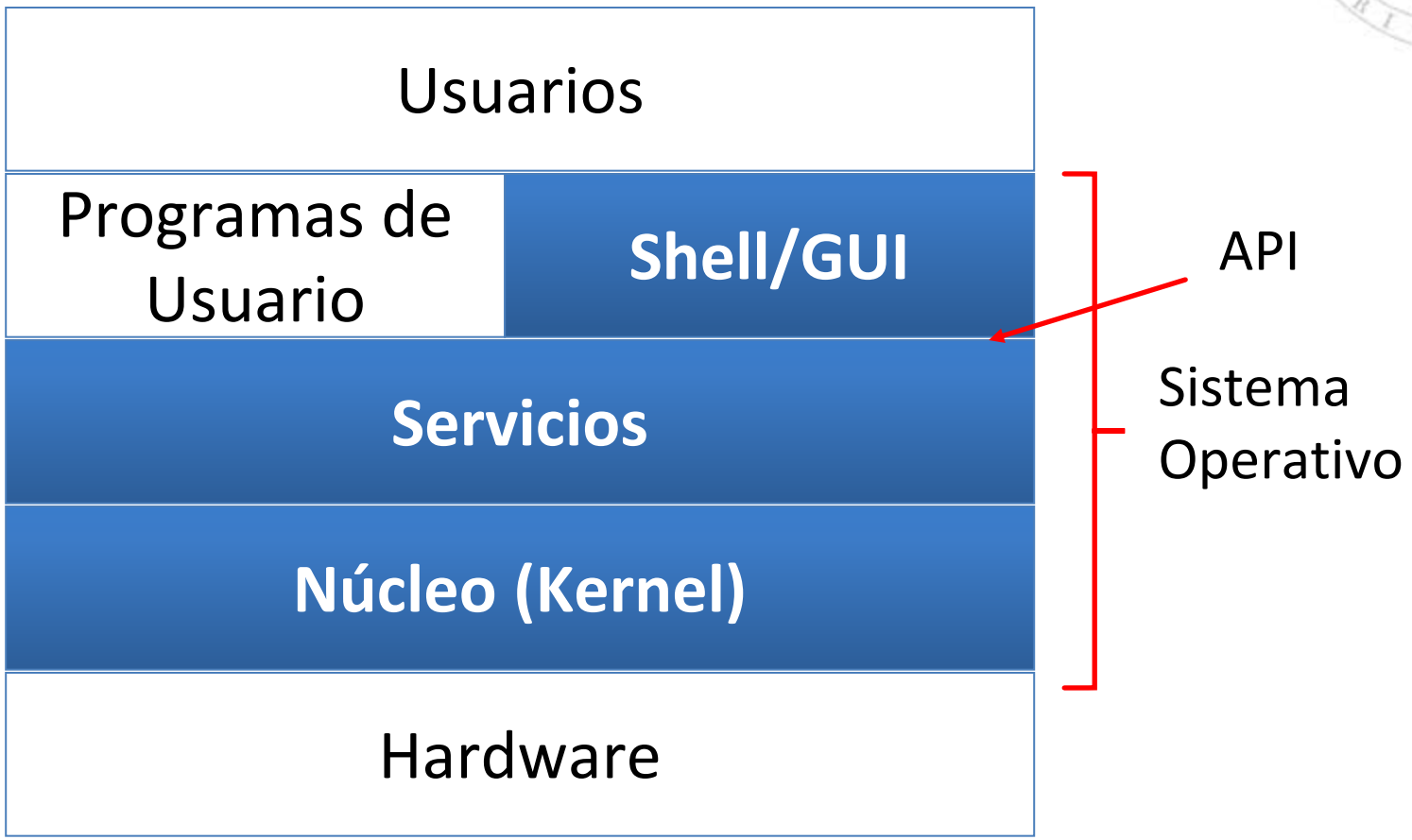
Estándar POSIX

- Interfaz estándar de sistemas operativos de IEEE.
- Objetivo: portabilidad de las aplicaciones entre diferentes plataformas y sistemas operativos.
  - POSIX: *Portable Operating System Interface for uniX*
- NO es una implementación. Sólo define una interfaz
- Diferentes estándares
  - 1003.1 Servicios básicos del SO
  - 1003.1a Extensiones a los servicios básicos
  - 1003.1b Extensiones de tiempo real
  - 1003.1c Extensiones de procesos ligeros
  - 1003.2 Shell y utilidades
  - 1003.2b Utilidades adicionales

Características de POSIX

- Nombres de funciones cortos y en letras minúsculas
  - fork
  - read
  - close
- Las funciones normalmente devuelven 0 en caso de éxito o –1 en caso de error
  - Variable errno
- Recursos gestionados por el sistema operativo se referencian mediante descriptores

2. Componentes del S.O.



Gestión de procesos

- El sistema operativo ofrece la noción de ***proceso***
  - Un *proceso* es un programa **en ejecución**
  - Con su estado (contexto de ejecución) y recursos asociados (variables de entorno, buffers, mapa de memoria,…)
- El sistema operativo nos proporciona servicios para…
  - Ejecución de programas, mediante la creación de procesos
  - Repartir el uso de la CPU y la memoria entre procesos
  - Establecer mecanismos de comunicación y sincronización entre procesos (e hilos)

Gestión de memoria

- El sistema operativo se hace cargo de…
  - Gestionar el espacio libre
  - Asignar memoria a los nuevos procesos
  - Evitar accesos a zonas de memoria que pertenecen a un proceso
  - Gestionar la compartición de memoria entre procesos
- Además, gestiona las transferencias de información entre memoria y almacenamiento secundario

Gestión de ficheros

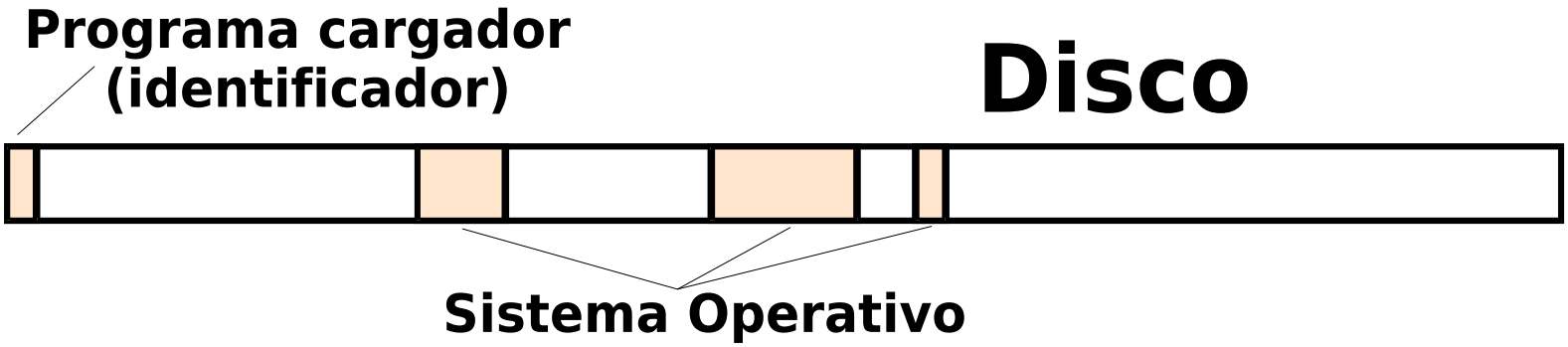
- El disco es almacenamiento no volátil.
  - Fichero: unidad lógica de almacenamiento.
- Mismos problemas que para la gestión de memoria:
  - ¿Qué zonas están libres/ocupadas?
  - Al crear un nuevo fichero, ¿dónde lo coloco?
  - Dado un fichero, ¿cómo lo encuentro en el disco?
  - Si un fichero es muy grande, ¿debe estar consecutivo en el disco?
  - Ese fichero me pertenece; mecanismos de privacidad/protección

Gestión de E/S

- ¿Y si usar un dispositivo de E/S supusiera revivir el laboratorio del ARM día tras día?
  - Programar dispositivos de E/S puede ser un poco tedioso: interrupciones, DMA….
- Definitivamente, que lo haga el sistema operativo.
  - Proporciona interfaces *friendly* para el acceso a los dispositivos
  - La funcionalidad ardua la realizar el *driver*
  - ¿Y cómo comunicar el *driver* con mi código?

3. Arranque del SO

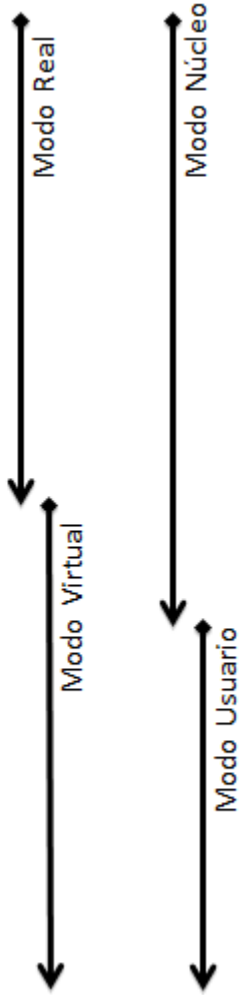
- El SO está almacenado en el disco



- Primera fase: Arranque del hardware (**Reset**)
  - El computador incluye una memoria ROM (no volátil) que contiene un programa encargado de transferir a memoria el cargador del SO y ejecutarlo: *Iniciador ROM*
- *Segunda fase: Arranque del Sistema operativo*
  - *El cargador del sistema operativo* carga en memoria los componentes del mismo, crea las estructuras de datos necesarias,…

Arranque del computador

- Bajo control del iniciador ROM
  - •Test del HW
  - •Carga en memoria el cargador del SO
- Bajo el control del cargador (*boot*) del SO
  - •Carga en memoria componentes del SO
- Inicialización bajo el control de la parte residente del SO
  - •Test del sistema de ficheros
  - •Creación de estructuras de datos internas
  - •Paso, si procede a modo Memoria Virtual
  - •Completa la carga del SO residente
  - •Habilita las interrupciones
  - •Crea el proceso de *login*
- Se entra en la fase normal de funcionamiento del SO. El SO se queda a la espera de que se produzca alguna interrupción



Invocación del sistema operativo

- ¿Cómo puede ejecutarse código del sistema operativo?
- Desde una aplicación en ejecución:
  - Por invocación directa desde el código del programa: llamada al sistema
  - Por la llegada de una interrupción (un timer, por ejemplo)
  - Porque se produzca una excepción (el reproductor incurre en una *violación de segmento*, por ejemplo)
- De manera interactiva: shell.

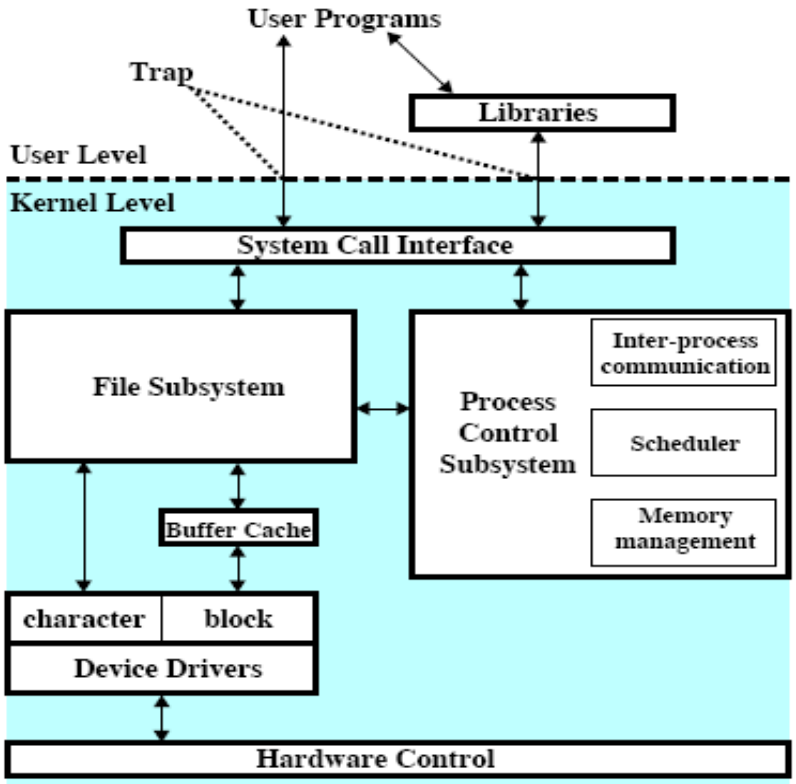
Activación de los servicios del sistema operativo (SO)

- El sistema operativo se activa también periódicamente → interrupciones generadas por el temporizador del sistema (cambio de modo)
  - Planificación: CPU accounting, expropiaciones de procesos/hilos, equilibrado de carga, …
  - Gestión del almacenamiento intermedio (escrituras a disco desde la cache de bloques)
  - Tareas periódicas asociadas a los protocolos de red

4. Llamadas al sistema

- Es fácil confundir una llamada al sistema con una función cualquiera….
  - Desde el punto de vista del programador, son indistinguibles
- Pero no son iguales:
  - Dicha función no está en el espacio de direcciones del proceso sino del kernel
  - En SOs monolíticos (como GNU/Linux), la función debe ejecutarse completamente en modo kernel (modo privilegiado)
- Para invocar una llamada al sistema desde un proceso de usuario es preciso forzar un cambio de modo usuario a modo kernel (instrucción trap en x86 o swi en ARM)

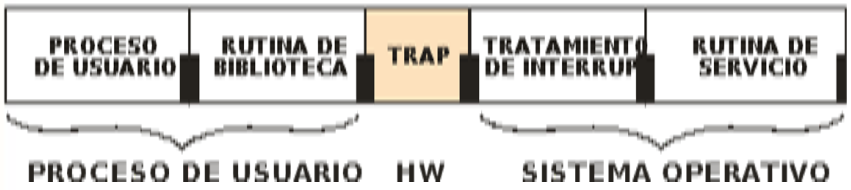
Llamadas al sistema: ejemplo




```
int main(void) {
  …
  v=read(fd,buf,nbytes);
  …
}
```

PASOS

1. Apilar* argumentos (fd,buf,nbytes)
2. Apilar* número de llamada al sistema (0 para read)
3. Ejecutar una instrucción *trap* que genera una interrupcion SW especial para cambiar de modo usuario a modo núcleo
4. El kernel accede a la tabla de llamadas al sistema para acceder a la función que la implementa y la invoca
5. Una vez finalizada la ejecución de la función, el kernel apila* el valor de retorno y vuelve a modo usuario



(*) Estos valores pueden pasarse usando registros, memoria o a través de la pila del programa