

# **Introducción a los sistemas operativos**

Adaptación de múltiples referencias bibliográficas

Juan Felipe Muñoz Fernández

Do you pine for the days  
when men were men and  
wrote their own device  
drivers?

Torvalds, Linus (Oct 5, 1991)



# Programar un controlador de Floppy Disk: PD765 (en palabras)

- Comandos
  - Leer datos
  - Escribir datos
  - Desplazar brazo del disco
  - Formatear pistas
  - Detectar dispositivo y unidades
  - Inicializar dispositivo y unidades
  - Recalibrar dispositivo y unidades
  - Otros
- En total 16 comandos para programar



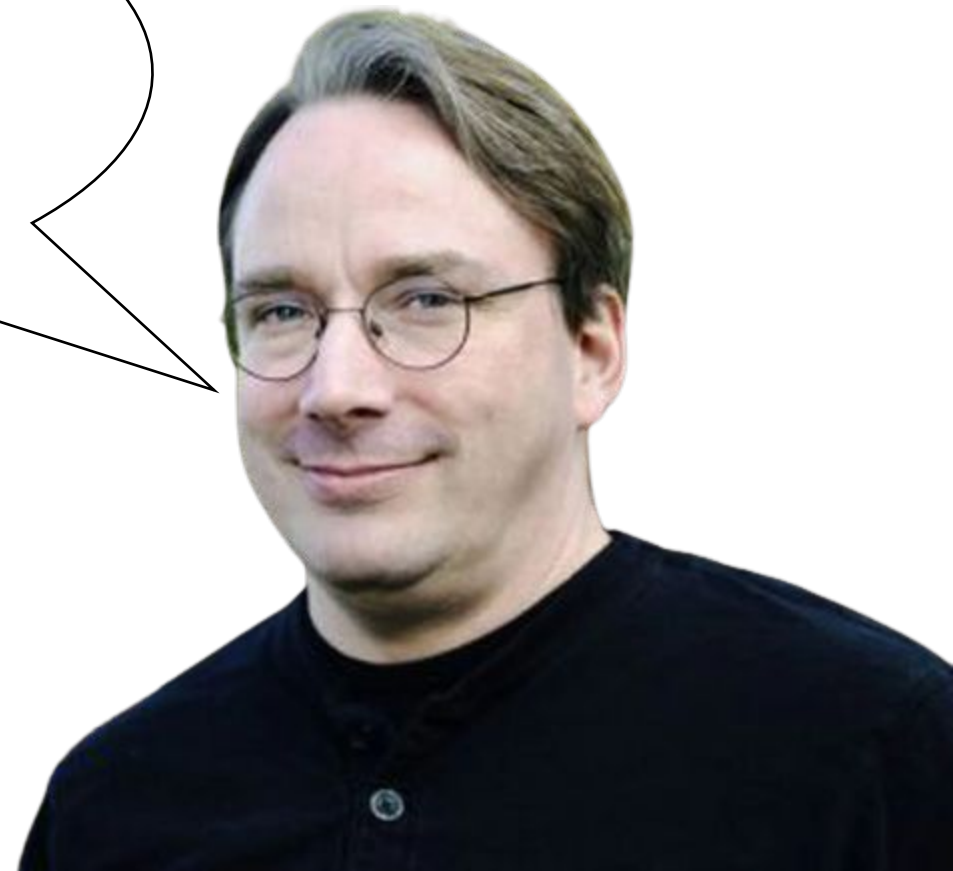
# Programar un controlador de Floppy Disk: PD765 (en palabras)

- 16 Comandos
  - Cada comando carga de 1 a 9 bytes en el registro del dispositivo.
- Comando básicos: **read** y **write**
  - Cada uno requiere 13 parámetros
  - Los 13 parámetros deben ir empaquetados en 9 bytes.
- ¿Qué especifican **read** y **write**?
  - Dirección de bloque a leer
  - Número de sectores por pistas
  - Modo de grabación
  - Espacio separación entre sectores
  - Decisión con sectores disponibles (datos eliminados)

# Programar un controlador de Floppy Disk: PD765 (en palabras)

- Operación completada chip controlador de dispositivo devuelve:
  - 23 campos de estado y error
  - Empaquetados en 7 bytes
- Otros para programar
  - ¿Motor encendido o apagado?
  - Encender motor
  - Apagar motor para evitar desgaste y daños en los medios
  - Y otros...
- ¿Sería usted capaz de programar un controlador de *floppy disk*?

Talk is cheap.  
Show me the  
code.



# Dos funcione básicas (S.O)

1. Proporcionar a los programadores un conjunto **abstracto** de recursos simples
2. Administrar los recursos de **hardware**

# Dos funciones básicas

1. Proporcionar a los programadores un conjunto **abstracto** de recursos simples
  - **Abstracción:** ocultar la complejidad y los detalles de hardware
  - Abstracción **simple** de lidiar con los detalles de leer y escribir en disco
  - P. Ej.: ver el **diskette** como una colección de archivos con nombre
  - Abrir archivo, leer/escribir, cerrar archivo

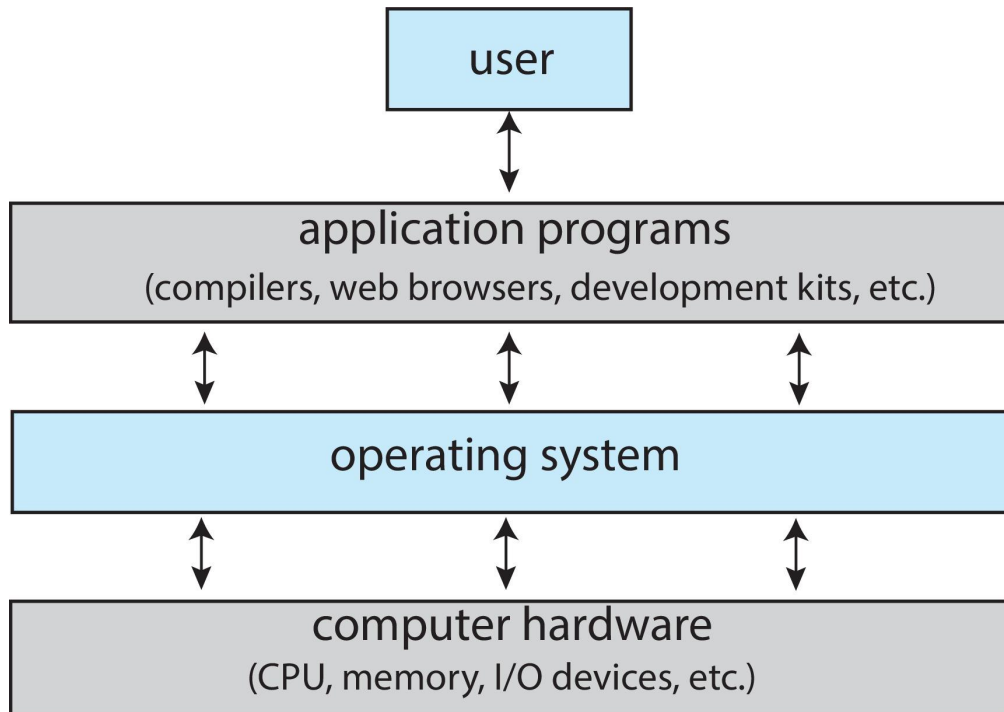


# Dos funciones básicas

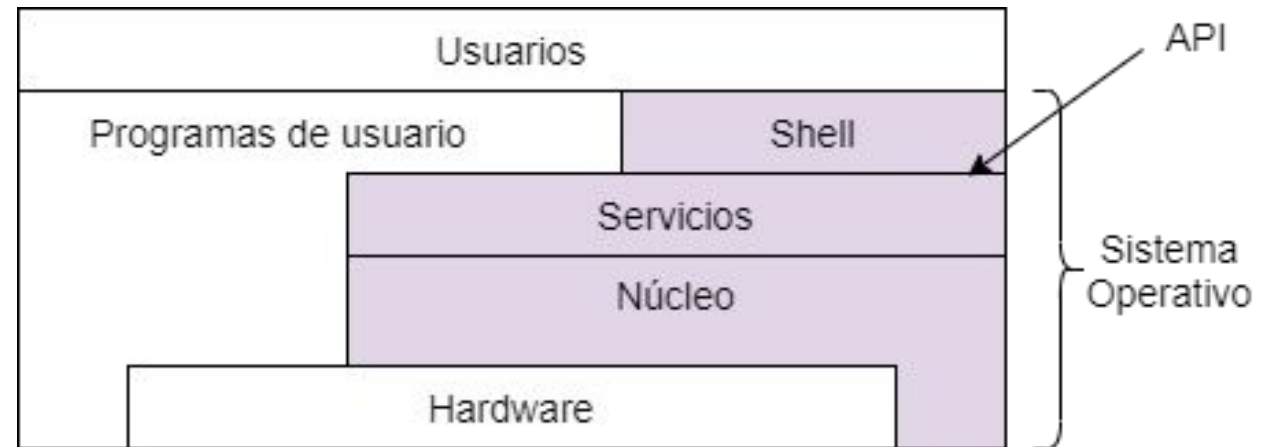
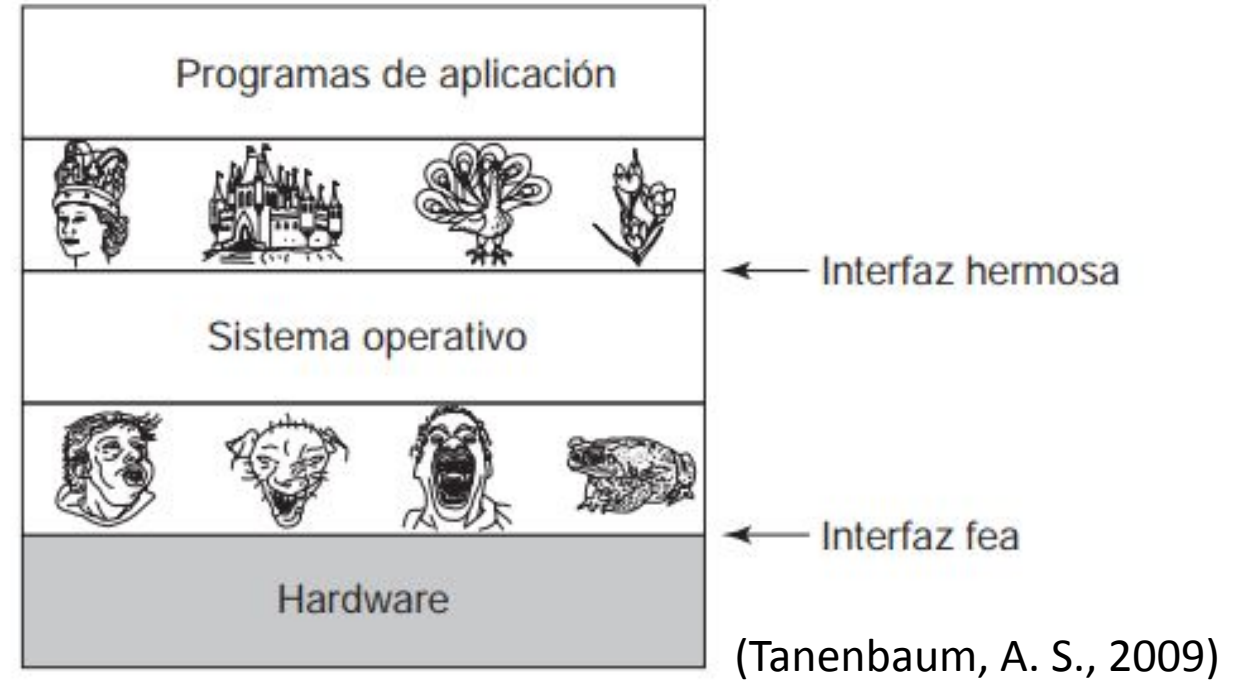
## 2. Administrar los recursos de **hardware**

- Asignación ordenada y controlada de recursos
- Programas compiten por los recursos
- P. Ej.: tres programas imprimiendo de manera simultánea.
- S.O impone **orden** al uso de los recursos
  - Orden de llegada. P. Ej.: la cola de impresión.
  - Uso de búferes para que los programas pongan allí lo que vana imprimir
- **Multiplexar** (compartir) los recursos
  - **En el tiempo** (por turnos): CPU, impresora
  - **En espacio** (una parte del recurso): Disco duro, RAM
- ¿Cómo se multiplexa? ¿Quién sigue? ¿Cuánto espacio? ¿Cuánto tiempo? Etc.

# Definición



(Silberschatz, A., et. al., 2018)



Con modificaciones tomada de:  
(Carretero Pérez, J. et. al., 2001)

# Definición

- ¿Qué es un SO?
  - Pieza de software (programa de computadora)
- ¿Por qué se necesita?
  - La CPU solo está ejecutando repetidamente un bucle infinito de tres pasos.
  - Complejidad del hardware

Si no hay nada para ejecutar, ¿qué sentido tiene tener una CPU?



1. Lectura instrucción IP
2. Incremento del IP
3. Ejecución instrucción

- ¿Cuál es su objetivo?
  - Simplificar el manejo y la utilización del computador
  - De manera eficiente
  - De manera segura
- ¿Cuáles son sus funciones?
  1. Gestión de los recursos (hardware)
  2. Ejecución de servicios
  3. Ejecución de órdenes de los usuarios

# Componentes básicos

- **Núcleo o Kernel**

- Gestiona los recursos de hardware
- Ofrece funcionalidad básica

- **Shell y/o GUIs\***

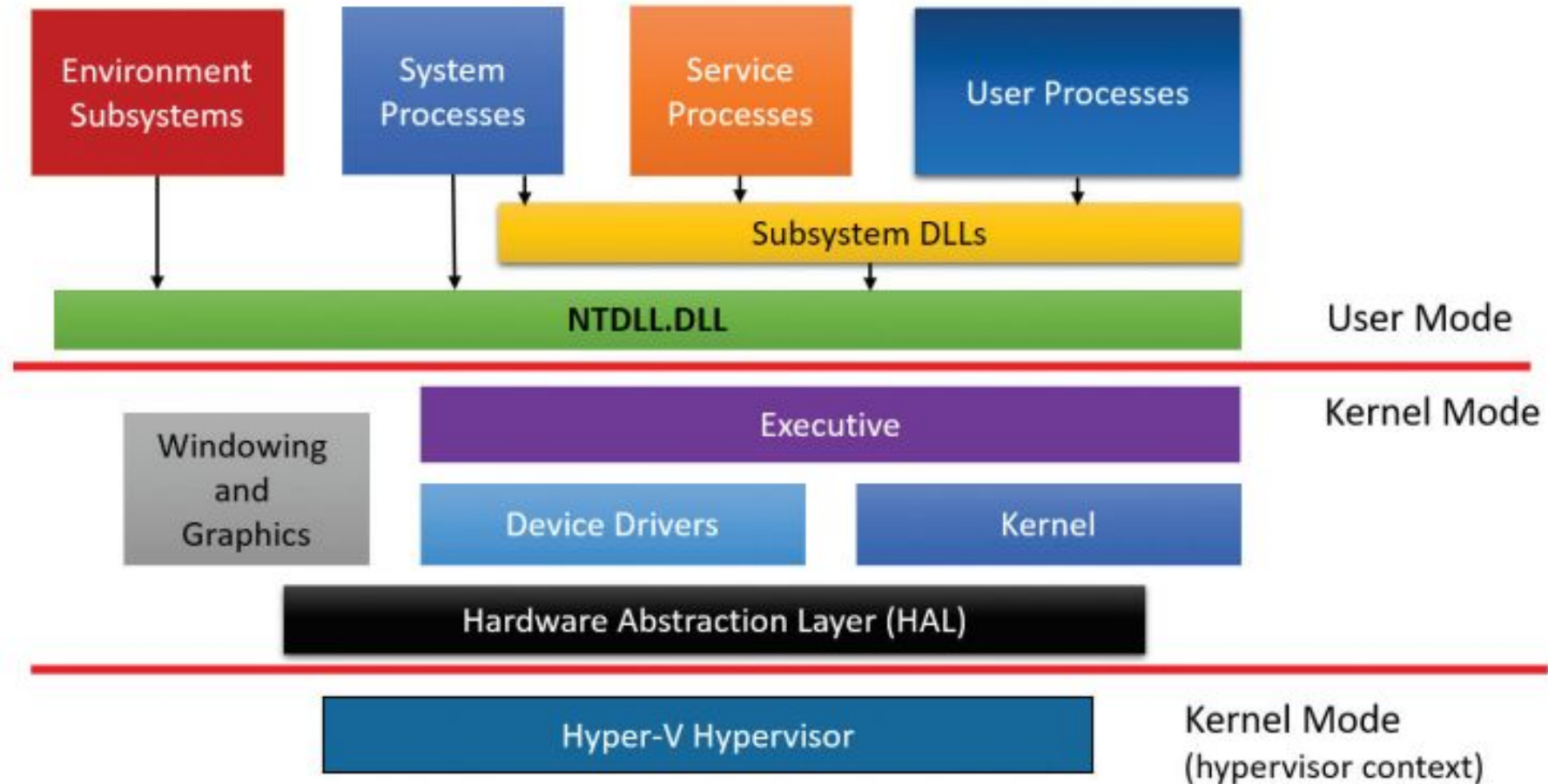
- Línea de comandos
- Intérprete de comandos
- Interfaz de texto o gráfica con la que el usuario dialoga con el SO

- **Servicios**

- API (Application Programming Interface)
- Proporciona servicios a los programas de usuario
- Windows API (Windows)
- System Calls (Linux)
- **Punto de entrada a los programadores**
  - Abstracciones simples del hardware

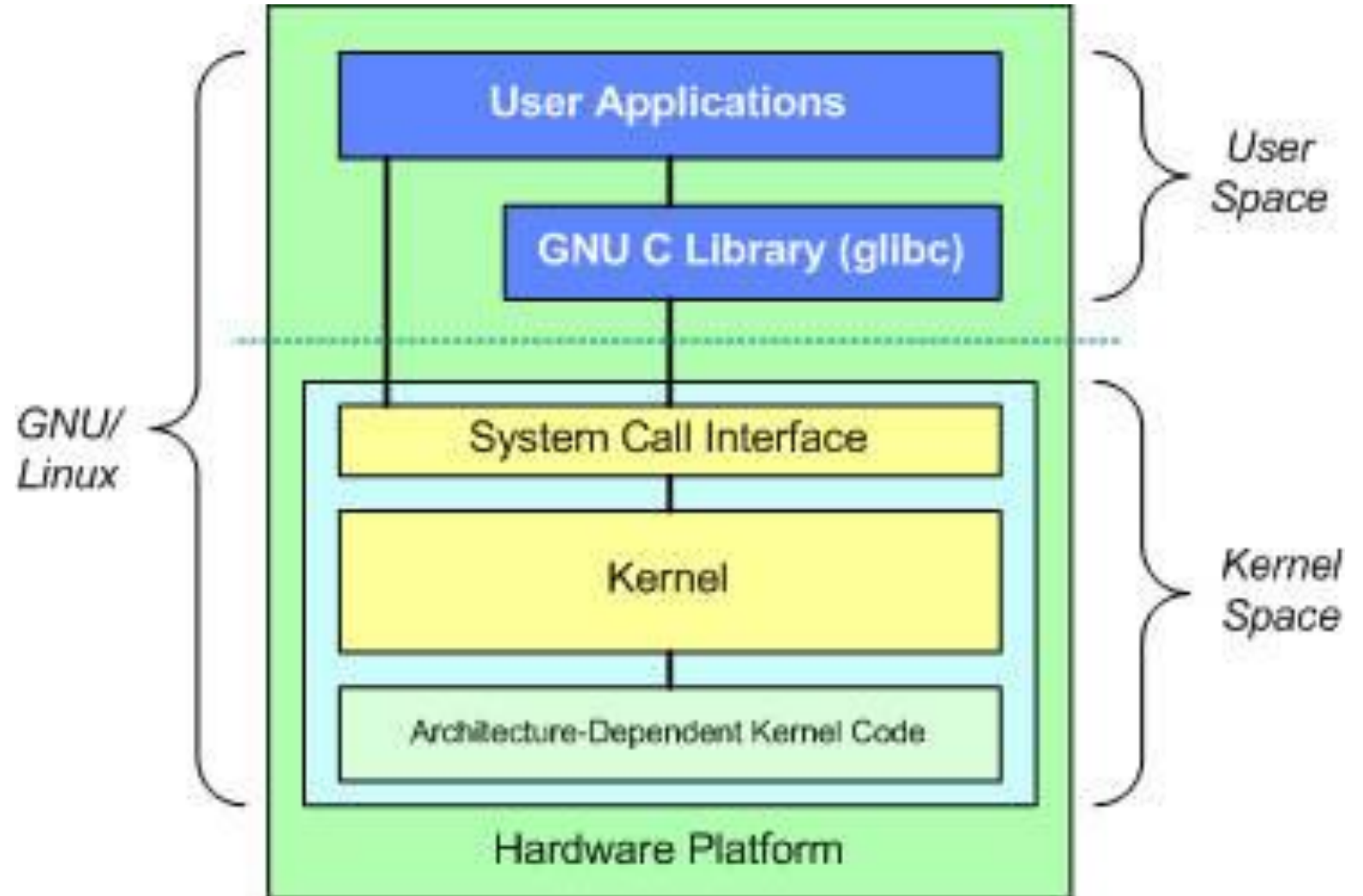
\*Algunos consideran que no hace parte el S.O

# Arquitecturas de diseño: Windows



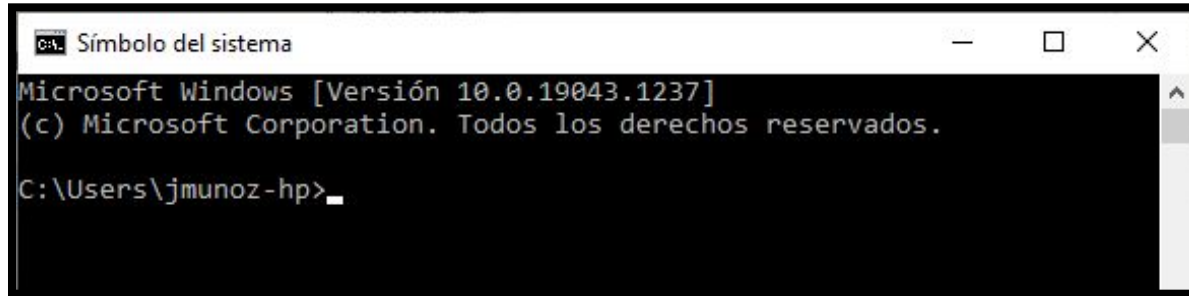
(Yosifovich, P., et. al., 2017)

# Arquitecturas de diseño: Linux

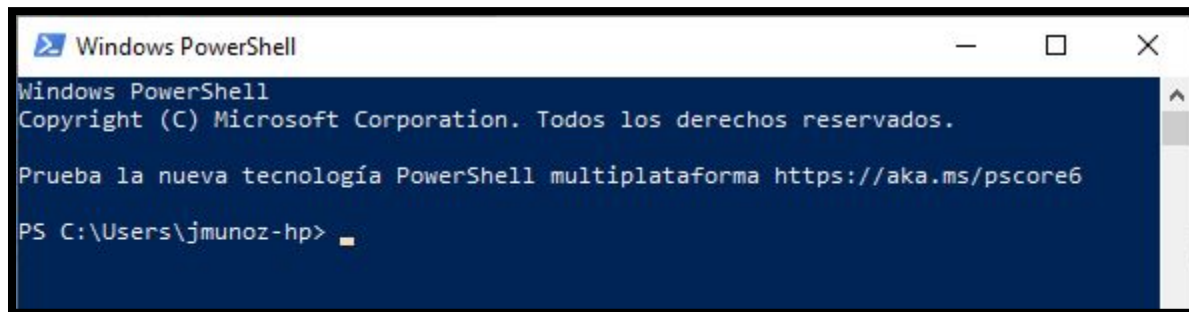


Fuente: <https://developer.ibm.com/articles/l-linux-kernel/>

# Shells e intérpretes de comandos



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19043.1237]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\jmunoz-hp>
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6
PS C:\Users\jmunoz-hp>
```



```
root : bash
File Edit View Scrollback Bookmarks Settings Help
[root@correo ~]#
```

# Windows API

- Documentación de Windows API
  - <https://docs.microsoft.com/en-us/windows/win32/apiindex/windows-api-list>
- Programa que usa las funciones `CreateFile()` y `CloseHandle()` definidas en el API de Windows.
- Programa crea un archivo en blanco llamado `archivo.txt`

```
#include <windows.h>
#include <stdio.h>

int main() {
    LPCWSTR filename = L"archivo.txt";

    HANDLE hFile;
    hFile = CreateFile(filename,
        GENERIC_WRITE,
        0,
        NULL,
        CREATE_ALWAYS,
        FILE_ATTRIBUTE_NORMAL,
        NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        return 2;
    }
    CloseHandle(hFile);
}
```



# Linux System Calls

- Programa que usa las funciones `open()` y `close()` definidas en las System Calls de Linux
- Programa crea un archivo en blanco llamado `archivo.txt`
- Documentación de estas funciones
  - <https://man7.org/linux/man-pages/man2/open.2.html>

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main()
{
    int hFile;
    const char filename[] = "archivo.txt";
    hFile = open(filename, O_CREAT);
    if (hFile < 0)
    {
        return 2;
    }
    close(hFile);
}
```

# Solo librería estándar de lenguaje C

- El mismo código funciona para ambos sistemas operativos.
- ¿**Cuándo** usar entonces la capa de servicios?
  - Funcionalidades **específicas** del S.O
  - P. Ej.: Registro de Windows (Windows)
  - P. Ej.: Colas de mensajes System V (Linux)
  - P. Ej.: Manipulación de usuarios y grupos (cada S.O)
  - Drivers

```
#include <stdio.h>

int main()
{
    FILE* hFile = NULL;
    const char filename[] = "archivo.txt";

    hFile = fopen(filename, "a");
    if (hFile == NULL)
    {
        return 2;
    }

    fclose(hFile);
}
```

El mismo código para Windows y Linux

# Funciones del SO

1. El S.O como gestor de recursos
2. El S.O como maquina extendida
3. El S.O como interfaz de usuario

# El S.O como gestor de recursos

- En un computador se ejecutan varios programas al tiempo
- Programas compiten por recursos
  - CPU, Memoria, E/S

1. **Asignación** □ Disponibilidad, prioridad, resolución de conflictos, recuperación luego del uso.
2. **Protección** □ Unos programas no puedan usar los recursos de otros, protección de la información.
3. **Contabilidad** □ Medir cantidad de recursos usados, monitoreo.

# El S.O como máquina extendida

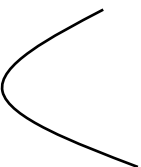
Extiende el modelo de programación de la computadora.

1. **Ejecución de programas (procesos)** □ lanzar ejecución, modificar condiciones ejecución, comunicar, sincronizar, administrar procesos
  - `exec()` □ Linux System Call
  - `CreateProcess()` Windows API
2. **Ordenes de E/S** □ operaciones de lectura/escritura, estado periféricos, dependiente del hardware
3. **Operaciones sobre archivos** □ creación, borrado, renombrado, apertura, escritura, lectura de archivos
4. **Tratamiento de errores** □ detección y tratamiento de errores
  - Operaciones inválidas

# El S.O como interfaz de usuario

- **A través del shell □ línea de comandos**

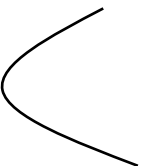
- Bucle infinito

- 
- Espera la orden del usuario
      - Interpreta las órdenes escritas del usuario
      - Ejecuta la orden

- Archivos de scripting (programación)

- **GUI (*Graphical User Interface*)**

- Bucle infinito (P. ej. En Windows Message Loop, Events Loops)

- 
- Lee los eventos producidos por el mouse, teclado
      - Interpreta los eventos
      - Ejecuta las acciones que indican los eventos

- Desarrollo de aplicaciones gráficas

# Ejemplo Message Loop en Windows

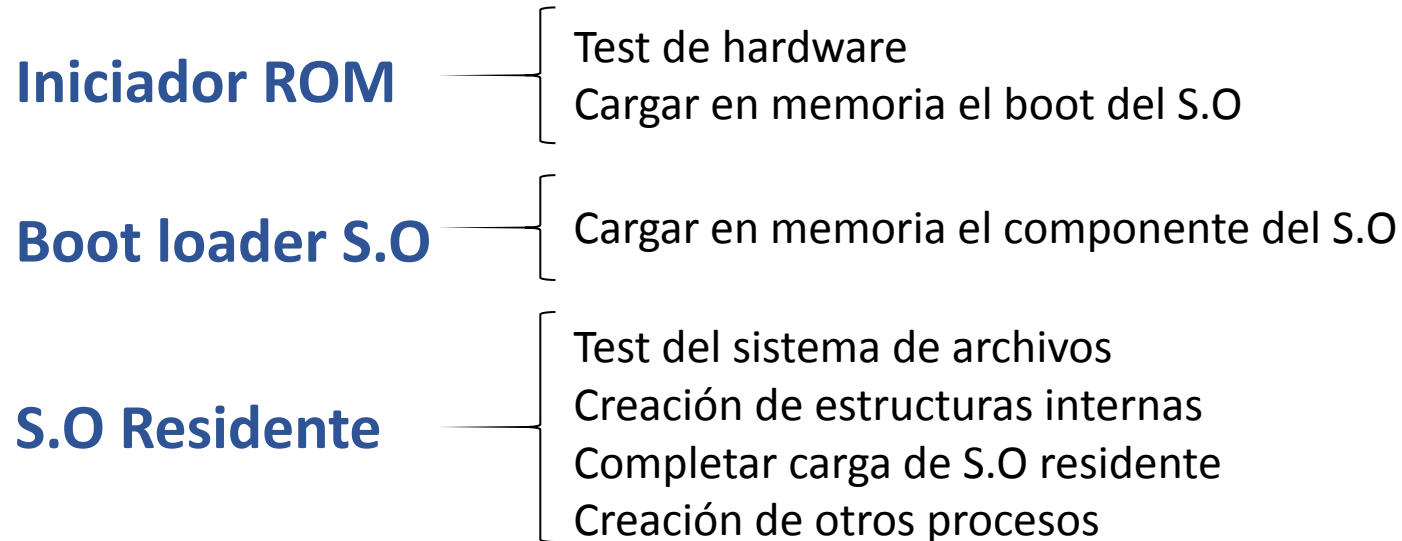
```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    MSG msg;
    BOOL bRet;

    while (1)
    {
        bRet = GetMessage(&msg, NULL, 0, 0);

        if (bRet > 0)  // (bRet > 0 indicates a message that must be processed.)
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        else if (bRet < 0)  // (bRet == -1 indicates an error.)
        {
            // Handle or log the error; possibly exit.
            // ...
        }
        else  // (bRet == 0 indicates "exit program".)
        {
            break;
        }
    }
    return msg.wParam;
}
```

# Arranque del computador

- Dos fases
  - Arranque del hardware
  - Arranque del S.O
- Actividades y responsabilidades



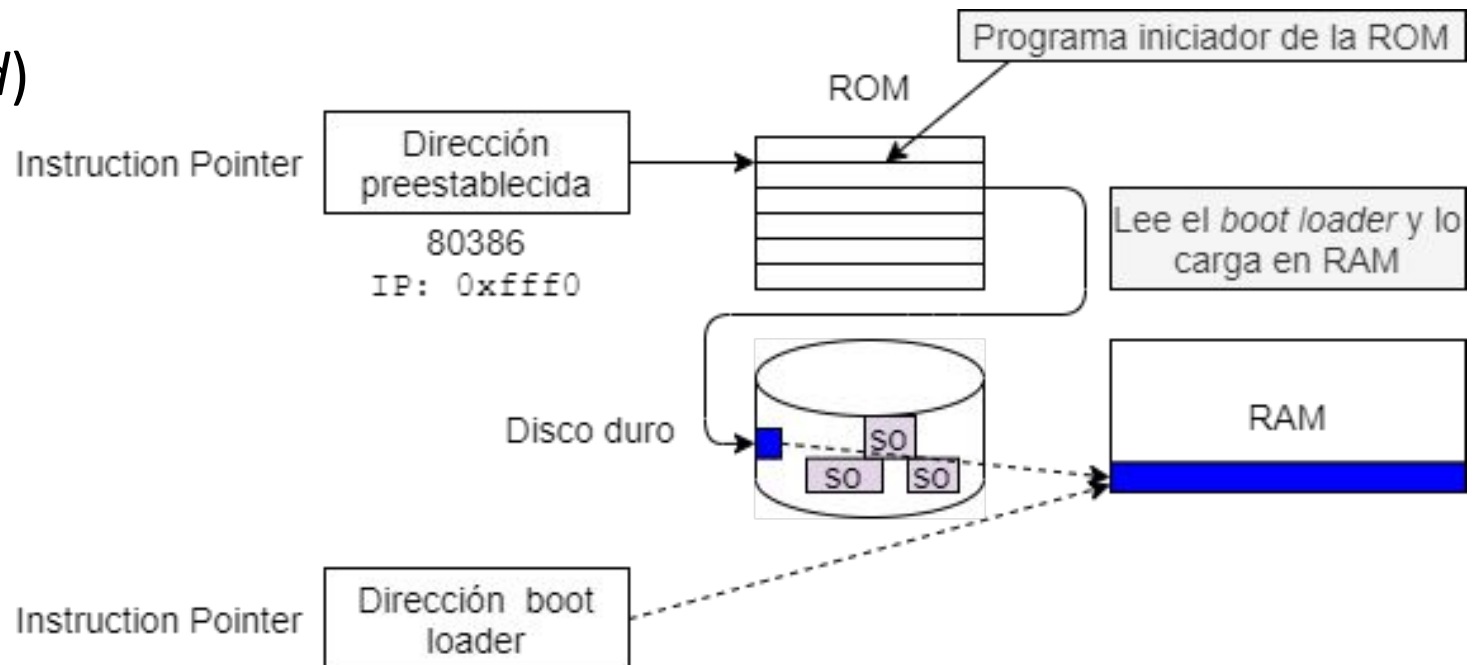


# Arranque del computador

- Arranque del hardware
  - Encendido o RESET
    - Cargar valores **predefinidos** en los registros
    - Registro IP con la dirección de comienzo del iniciador ROM (programa)
  - Iniciador ROM
    - Comprobación del sistema
    - Lectura y carga del *boot loader* del sistema operativo en memoria
    - Entregar control al *boot loader*: IP apunta a la primera instrucción del *boot loader*

# Arranque del computador

- Ubicación del sistema operativo
  - *Boot loaders* están siempre almacenados en zonas predefinidas del disco
  - P. Ej.: los cuatro primeros sectores del disco.
  - Tamaño fijo
  - MBR (*Master Boot Record*)



# Arranque del computador

- Arranque del sistema operativo
  - Boot loader lleva a memoria algunos componentes del S.O
  - Se inician operaciones tales como
    - Comprobación del sistema: hardware, sistema de archivos
    - Establecimiento de estructuras propias del S.O: procesos, memoria, E/S
    - Cargar en memoria S.O residente (siempre está en memoria)
    - Iniciar otros programas: login, demonios, procesos auxiliares
- Lectura recomendada: Kernel booting process. Part 1
  - <https://0xax.gitbooks.io/linux-insides/content/Bootting/linux-bootstrap-1.html>

# Referencias

- Carretero Pérez, J., García Carballeira, F., De Miguel Anasagasti, P., & Pérez Costoya, F. (2001). Introducción a los sistemas operativos. In *Sistemas operativos. Una Visión Aplicada* (pp. 33–74). McGraw Hill.
- Tanenbaum, A. S. (2009). Introducción. In *Sistemas Operativos Modernos* (3rd ed., pp. 3–75). Pearson Educación.
- Yosifovich, P., Ionescu, A., Russinovich, M. E., & Solomon, D. A. (2017). *Windows Internals Part 1. System architecture, processes, threads, memory management, and more* (7th ed., Ser. Windows Internals). Microsoft Press.