Estructuras de Sistemas Operativos

UNIVERSIDAD DEL BÍO-BÍO

Agenda

1. Objetivos
2. Servicios del SO
3. Interfaz de usuario
4. Llamadas al Sistema
5. Tipos de Llamadas al Sistema
6. Programas del Sistema
7. Diseño e implementación del SO
8. Estructura del SO
9. Máquinas Virtuales

UNIVERSIDAD DEL BÍO-BÍO

Objetivos

- Describir los servicios que un sistema operativo proporciona a los usuarios, a los procesos y otros sistemas.
- Exponer las diversas formas de estructurar un SO.
- Explicar cómo se instalan, personalizan y arrancan los SO.

Servicios del SO

- Un conjunto de servicios del SO proporciona funciones que resultan útiles al usuario:
  - Interfaz de usuario: Los SO operativos disponen de una interfaz de usuario UI.
    - Línea de comandos (CLI), Interfaz gráfica de usuario (GUI).
  - Ejecución de programas: El sistema debe poder cargar un programa en memoria y ejecutarlo. Todo programa debe poder terminar su ejecución, de forma normal o anormal (indicando un error).
  - Operaciones de E/S: Un programa en ejecución puede requerir E/S dirigidas a un archivo o a un dispositivo de E/S.
  - Manipulación del Sistema de archivos: Los programas necesitan leer/escribir en archivos/directorios, crear/borrar archivos/directorios, búsquedas en un determinado archivo, gestión de permisos de archivos/directorios.

UNIVERSIDAD DEL BÍO-BÍO

Servicios del SO

- ...
  - Comunicaciones: procesos pueden intercambiar información, en el mismo computador o entre computadores en una red.
    - mediante memoria compartida o paso de mensajes (transferencia de paquetes de información entre procesos por el SO)
  - Detección de errores: SO necesita constantemente detectar posibles errores:
    - en HW del procesador y memoria (por fallo de alimentación), en dispositivo de E/S (error de paridad, fallo en conexión de red), en los programas de usuario (desbordamiento aritmético).
    - por cada error el SO debe tomar la acción apropiada para asegurar el correcto funcionamiento y la coherencia.
    - La depuración puede ayudar a usuarios/programadores para uso eficiente del sistema (GDB debugger[1], Ejemplos[2]).

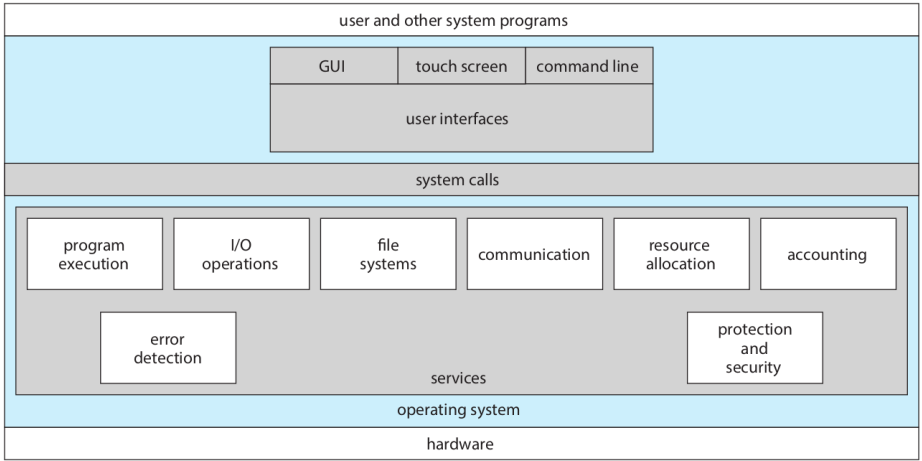
[1] GDB: The GNU Project Debugger. `https://www.sourceware.org/gdb/`
[2] Depurar un Programa en Lenguaje C con GDB. `http://ubb.cl/928822`

UNIVERSIDAD DEL BÍO-BÍO

Servicios del SO

- Otro conjunto de funciones del SO, pensadas en garantizar la eficiencia del propio sistema y no al usuario. Los sistemas de múltiples usuarios pueden ser más eficientes cuando se comparten los recursos:
  - Asignación de recursos: cuando hay varios usuarios o procesos, deben asignarseles los recursos necesarios (CPU, memoria,etc.).
  - Contabilidad: para mantener información de uso de recursos por parte de los usuarios.
  - Protección y seguridad: los propietarios de información almacenada en un sistema necesitan poder controlar el uso de esa información.
    - Protección: implica asegurar que todos los accesos a los recursos del sistema estén controlados.
    - Seguridad: garantizar seguridad del sistema frente a posibles intrusos - autenticación de usuarios.

Vista de Servicios del SO.



CLI

- Interfaz de línea de comandos (CLI) o intérprete de comandos permite la entrada directa de comandos:
  - algunos implementados en el kernel otros como programas de sistema (Windows, UNIX/Linux).
  - algunos ofrecen varios intérpretes de comandos, conocidos como shells: Bash (Bourne-Again SHell, shell C, etc.
  - la función principal del intérprete de comandos es obtener y ejecutar el siguiente comandos especificado por el usuario.

GUI

- Interfaz gráfica de usuario. Más amigable para el usuario:
  - Usualmente uso de mouse, teclado y monitor.
  - íconos representan archivos, programas, acciones, etc.
  - botones del mouse sobre algun objeto causan varias acciones (provee información, opciones, etc.)
  - Distribuciones de Linux vienen con entornos de escritorio (Fluxbox, GNOME, KDE, etc.)[3]

[3] `https://www.softzone.es/linux/programas/entornos-escritorio-linux/`

UNIVERSIDAD DEL BÍO-BÍO

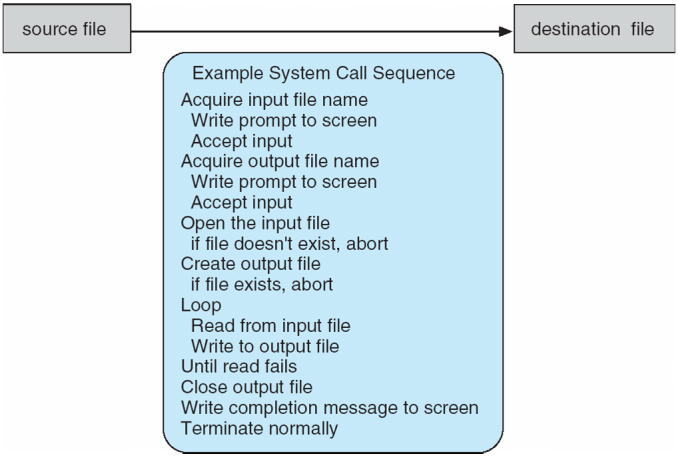
Touch screen

- Los sistemas móviles como teléfonos y tabletas proporcionan una interfaz de pantalla táctil, lo que permite a los usuarios deslizar los dedos por la pantalla o presionar botones en la pantalla para seleccionar opciones.

Llamadas al Sistema

- Proporcionan una interfaz para poder invocar los servicios que el SO ofrece.
- Típicamente disponibles como rutinas escritas en C y C++.
- Uso de API (Interfaz de Programación de Aplicaciones). Conjunto de funciones que el programador puede utilizar (paso de parámetros, retorno de valores).
- Tres APIs disponibles (Win32 - windows, POSIX (Portable Operating System Interface - X de UNIX) - unix,linux, JAVA - máquina virtual java - JVM)

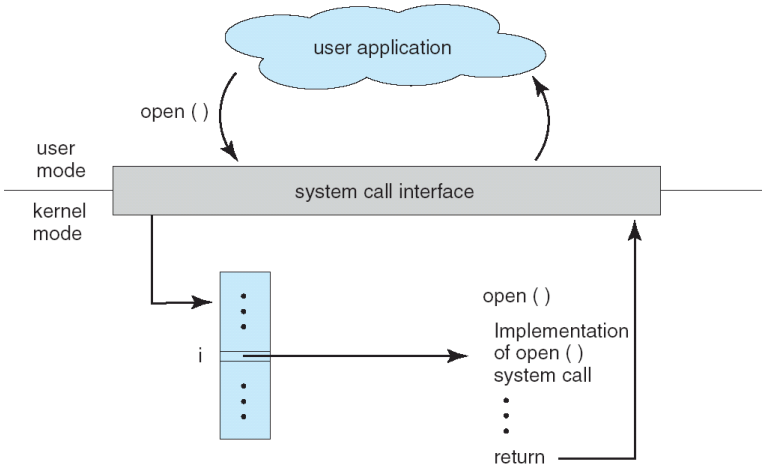
Ejemplo llamada al sistema



Implementación de llamada al sistema

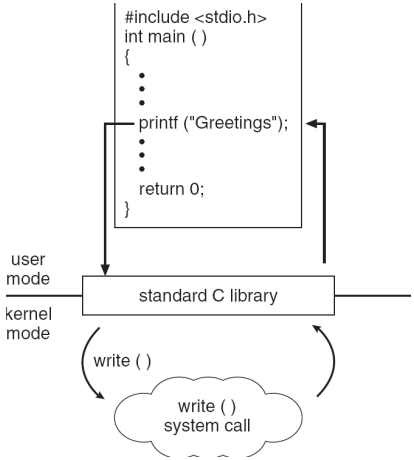
- Típicamente, cada llamada tiene asociado un número. La interfaz de llamadas mantiene tabla indexada según estos números.
- Interfaz de llamadas invoca a la llamada necesaria del kernel del SO, devolviendo el estado y los posibles valores de retorno.
- Quien realiza no necesita saber como se implementa dicha llamada o que ocurre durante su ejecución, solo debe ajustarse a la API específica y entender que hará el SO como resultado de la ejecución de la llamada.
- La API oculta la mayor parte de los detalles al programador.

Relación API - Interfaz llamada a Sistema - SO



Ejemplo Libreria Estandar de C

- Programa en C invocando la función printf(), la cual hace la llamada a sistema write()



Ejemplo Libreria Estandar de C. Llamada a sistema write()[4]

```
# hello.c
#include <stdio.h>


int main(int argc, char **argv)
{
    printf("Hola mundo!");

    return 0;
}
```

```
$ strace ./hello

execve("./hello", ["./hello"], ...
...
write(1,"Hola mundo!",12Hola mundo!)=12
exit_group(0) = ?
+++ exited with 0 +++
```

[4] https://man7.org/linux/man-pages/man2/write.2.html

Tipos de llamadas al sistema

- Pueden agruparse de forma muy general en cinco categorias:
  - Control de procesos.
    - terminar, abortar
    - cargar, ejecutar
    - crear y terminar procesos
    - obtener atributos del proceso, etc.
  - Manipulación de archivos.
    - crear y borrar
    - abrir, cerrar
    - leer, escribir, reposicionar
    - obtener atributos del archivo, etc.

Tipos de llamadas al sistema

- ...
  - Manipulación de dispositivos.
    - solicitar, liberar dispositivo
    - leer, escribir, reposicionar
    - obtener atributos del dispositivo,
    - conectar y desconectar dispositivos lógicamente, etc.
  - Mantenimiento de información.
    - obtener, definir hora y fecha
    - obtener datos del sistema, etc.
  - Comunicaciones.
    - crear, eliminar conexiones de comunicación
    - enviar, recibir mensajes
    - transferir información de estado
    - conectar, desconectar dispositivos remotos, etc.

Ejemplo llamadas a sistema en Windows y Unix

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

UNIVERSIDAD DEL BÍO-BÍO

Programas del Sistema

- Proporcionan un entorno cómodo para desarrollar y ejecutar programas. Pueden dividirse en las siguientes categorías:
  - Administración de archivos: Estos programas crean, borran, copian, cambian nombres, listan archivos/directorios.
  - Información de estado: Algunos solicitan cosas sencillas (la hora), otros mas complejos (información de rendimiento).
  - Modificación de archivos: editores de texto para crear, modificar archivos.
  - Soporte de lenguajes de programación: compiladores, ensambladores, depuradores, intérprete para lenguajes como C, C++, java, perl.
  - Carga y ejecución de programas: carga en memoria de programas compilados.
  - Comunicaciones: proporcionan mecanismos para crear conexiones virtuales entre procesos, usuarios y computadores. Usuarios pueden enviar mensajes a las pantallas de otros, explorar páginas web, enviar correo electrónico, etc.

UNIVERSIDAD DEL BÍO-BÍO

Diseño e implementación del SO

- No existen soluciones completas y únicas para enfrentar los problemas de diseño e implementación de SSOO. Si hay algunos métodos que han demostrado ser adecuados:
  - Objetivos del Diseño: Definir objetivos y especificaciones.
    - elección del hardware.
    - tipo de sistema (por lotes, tiempo compartido, monousuario, multiusuario, etc.)
    - objetivos de usuario (cómodo de utilizar, fácil de aprender y usar, etc.) y de sistema (fácil de diseñar, implementar y mantener).
  - Mecanismos (cómo hacer algo) y políticas (qué hacer).
    - ejemplo: el temporizador es un mecanismo para asegurar la protección de la CPU, datos de temporización para un usuario concreto es una decisión de política.
    - la separación de mecanismos y politicas es muy importante, permite flexibilidad si necesitan cambiarse a futuro las políticas.
  - Implementación.
    - una vez diseñado debe implementarse, tradicionalmente se escribían en lenguaje ensamblador, hoy se utiliza C o C++ (Linux y Windows).

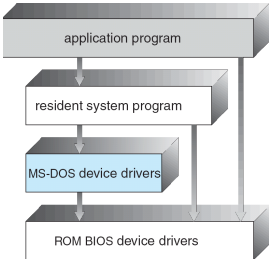
UNIVERSIDAD DEL BÍO-BÍO

Estructura del SO

- Ingeniería de un SO debe hacerse cuidadosamente para que funcione apropiadamente y sea fácil de modificar.
- Método habitual, dividir en componentes más pequeños en lugar de un sistema monolítico.
- Cada uno de estos componentes deben ser bien definidos con E/S y funciones cuidadosamente especificadas.

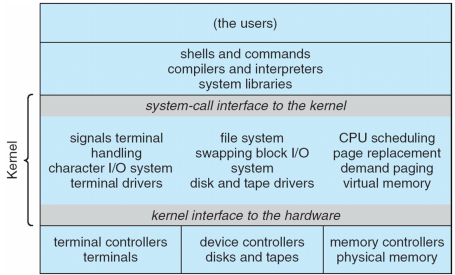
Estructura Simple

- No existe protección ni multiprogramación.
- Falla de programación puede hacer caer el Sistema.
- MS-DOS diseñado para proporcionar máxima funcionalidad en el menor espacio posible.
- MS-DOS diseñado en HW que no daba soporte de protección ni modo dual.



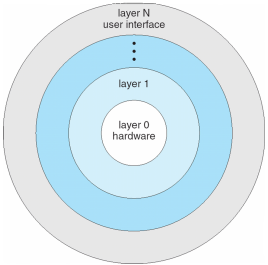
Sistema Monolíticos

- Permiten multiprogramación y múltiples usuarios.
- El SO es un conjunto de procedimientos que se agrupan en el núcleo.
- El núcleo está protegido (modo dual).
- Núcleo tiende a ser de gran tamaño, No modular.
- Frente a cualquier cambio se debe compilar el núcleo completo.



Estructura en niveles

- Permiten mejor modularización y protección de los componentes del sistema.
- Oculta detalles a niveles superiores, libertad a programadores en la implementación de rutinas de bajo nivel.
- Comunicación entre niveles introduce mucho costo en la operación (overhead).
- Dificultad, definir los niveles apropiadamente (cada nivel usa servicios de nivel inferior).

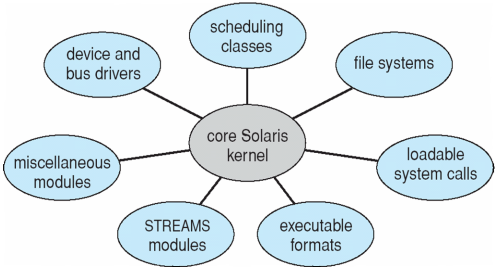


Microkernel

- Elimina componentes no esenciales del kernel y los implementa como programas de sistema y de nivel de usuario.
- Kernel más pequeño.
- Facilidad para ampliar el SO (servicios nuevos se añaden al espacio de usuario sin necesidad de modificar el kernel).
- Pueden tener un rendimiento peor que otras soluciones debido a la carga de procesamiento adicional.
- QNX, Windows NT (primeras versiones).

Módulos

- Los SSOO más modernos implementan kernel modulares.
- Kernel dispone de componentes fundamentales.
- Enlaza dinámicamente los servicios adicionales (módulos) (en el arranque o en tiempo de ejecución).
- Solaris, Linux, Mac OS X.



Módulos

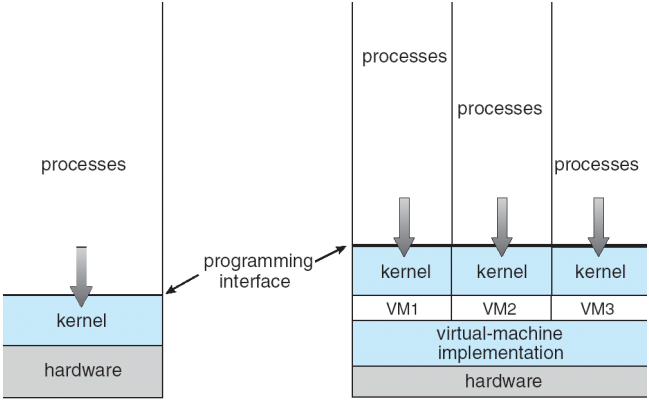
```
# ls /lib/modules/6.1.0-12-amd64/kernel/
9p              cachefiles  exfat     hfsplus     netfs
adfs            ceph        ext4      hpfs        nfs
affs            coda        f2fs      isofs       nfs_common
...

# lsmod | grep nfs

# modprobe -v nfs
insmod /lib/modules/6.1.0-11-amd64/kernel/fs/fscache/fscache.ko
insmod /lib/modules/6.1.0-11-amd64/kernel/fs/lockd/lockd.ko
insmod /lib/modules/6.1.0-11-amd64/kernel/fs/nfs/nfs.ko

# modprobe -v -r nfs
rmmod nfs
rmmod fscache
rmmod lockd
```

Máquinas Virtuales

- Permite crear sobre una máquina varias máquinas virtuales.
- Cada máquina virtual puede ejecutar un SO diferentes.
- Permite compartir el mismo HW por diferentes SO de forma concurrente.
- VMware, VirtualBox, KVM, etc.

Máquinas Virtuales

- (a) Máquina No virtual - (b) Máquina Virtual.



Máquinas Virtuales