Centro de
Investigación en
Computación

Architecture / Kernel

Arquitectura / Nucleo

Modelo

Componentes

Course

Operating System (with focus on Security)

Instructor

Acosta Bermejo Raúl

Lecture notes

2024-B
28 de agosto del 2024
Última actualizacioón

Instituto
Politécnico
Nacional

Table of contents (outline)

Tabla de contenido

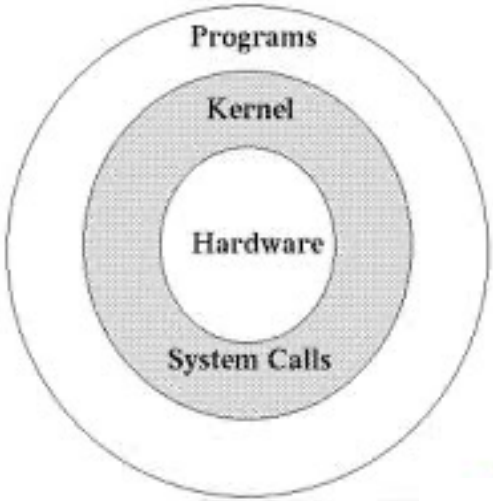
1. Introducción
2. Linux
    1. Arquitectura y Módulos
    2. ABI, LSB
    3. Librerías
    4. Demonios y servicios
3. Mac OS X
4. Windows
5. Standares

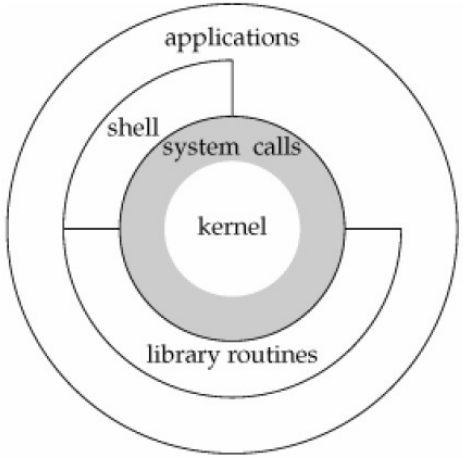
Introducción

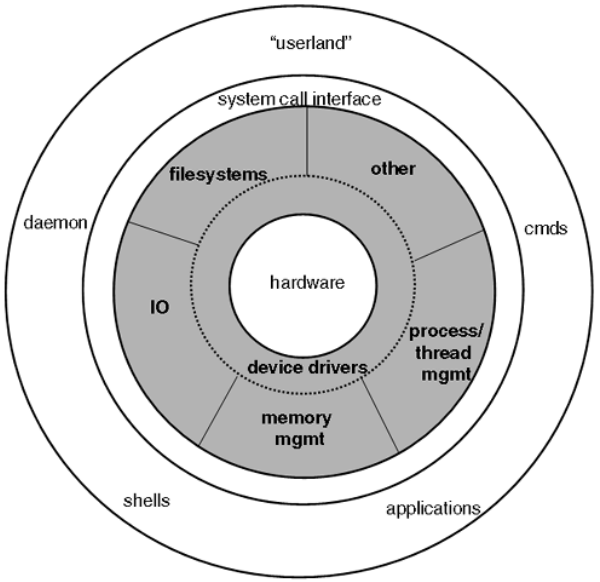
Arquitectura del SO

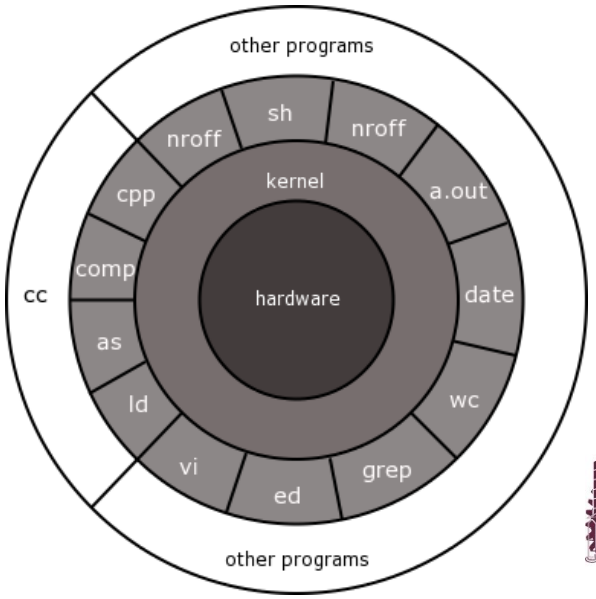
Modelo a capas (Layers)

Diseño básico





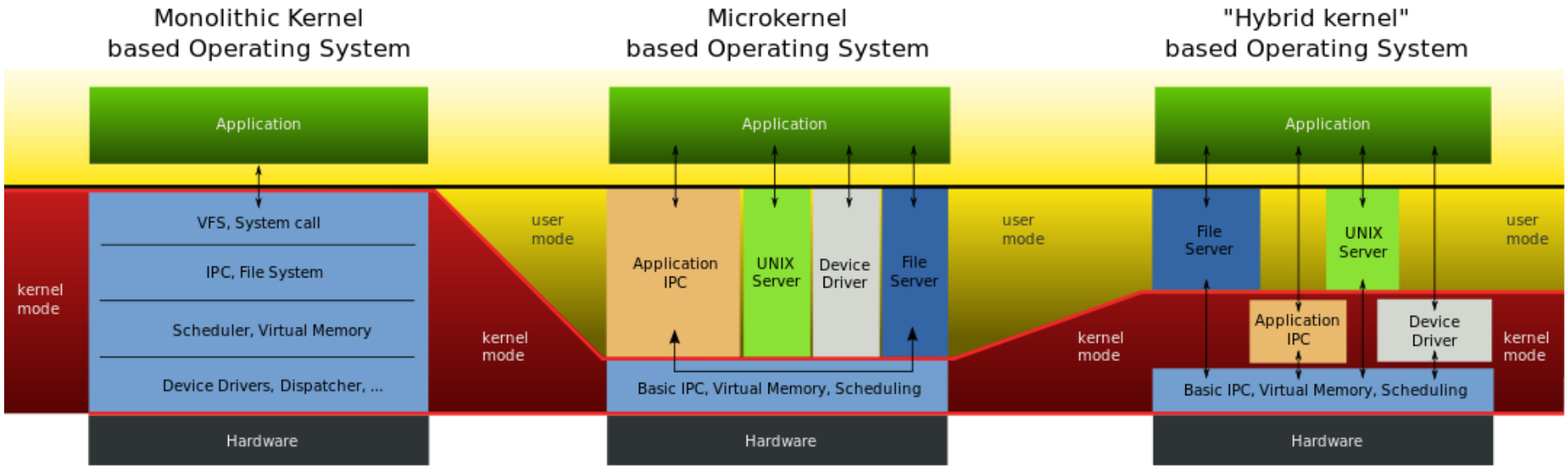




Introducción

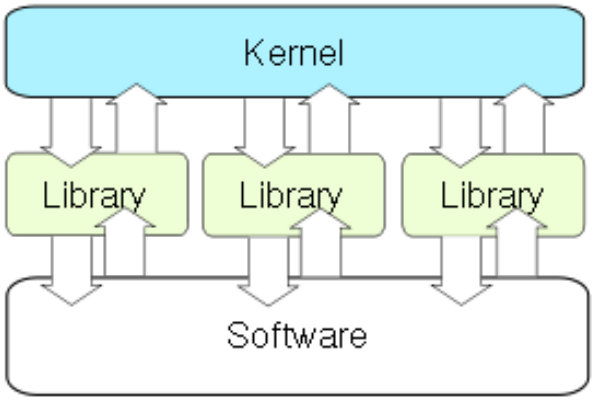
Arquitecturas - Diseño

En la evolución de los SO se creado varios diseños/arquitecturas. El elemento principal que define la arquitectura del SO es el núcleo (**kernel**) el cual puede tener las siguientes formas:



Introducción

Arquitecturas - Diseño



Exokernel

- El término se refiere a un sistema creado con fines de investigación en el Instituto Tecnológico de Massachusetts sobre OpenBSD y otros sistemas operativos similares.
- Su propósito es crear una especie de capa de software para otros sistemas virtuales.
- Tradicionalmente los núcleos intentaban hacer invisibles los recursos de hardware a las aplicaciones requiriendo que interactúen con el hardware de alguna manera conceptual.
- Estos modelos incluyen, por ejemplo, sistemas de archivos para almacenamiento en disco, espacio de direcciones virtual.
- Estas abstracciones del hardware hacen más fácil escribir programas en general, pero limitaban el rendimiento y reprimían la experimentación de nuevas abstracciones.
- Una aplicación orientada a la seguridad requiere un sistema de archivos que no deje datos viejos en el disco, mientras que una orientada a la fiabilidad necesita un sistema que almacene esos datos para recuperarlos en caso de fallos.

Introducción

Arquitecturas - Diseño

Núcleo Monolítico

- El núcleo se comporta como un solo programa.
- Linux tiene un núcleo monolítico de decenas de megas y cientos de componentes encerrados dentro de un solo modulo.
- Desventajas (problemas mencionados por Linus Torvalds)
  - Cuando un componente de hardware requiere un parche para su funcionamiento correcto, este puede traer inconvenientes a la estructura del resto del kernel monolítico.
  - Por este motivo se prefiere (Linus) no dar soporte a determinados componentes de Hardware con el fin de conservar el funcionamiento del kernel Linux.
- Ventajas
  - No hay problemas de no encontrar algún componente y todo se accesa rápido.

Introducción

Arquitecturas - Diseño

Micro-kernel

- El microkernel se comporta como un programa compuesto por mini-programas o módulos que gestionan de forma independiente los componentes externos e internos con los cuales interactúa el núcleo.
- Ventajas
  - Reducción de la complejidad.
  - Descentralización de los fallos (un fallo en una parte del sistema no se propagaría al sistema entero).
    - La ejecución de una aplicación no se verá afectada por la ejecución de un componente del hardware como puede ser un mouse, audio, etc.
    - El SO trabaja e interactúa de manera independiente con cada una de los componentes o servicios.
  - Facilidad para crear y depurar controladores de dispositivos.
- Hurd y Minix utilizan el modelo de microkernel. Hurd deriva de Mach.

Introducción

Arquitecturas - Diseño

Micro-kernel

- Desventajas
  - Las funcionalidades básicas como el manejo de archivos requieren de mucha comunicación externa con otros componentes.
- Ejemplos de SO con micro-nucelo
  - Mach, Amoeba, Chorus
  - Hurd y Minix utilizan el modelo de microkernel. Hurd deriva de Mach.
  - Qnx, Symbian, etc.

Introducción

Arquitecturas - Diseño

Nuevos componentes en el kernel

Si un sistema operativo monolítico requiere de nuevas funcionalidades, este no podría ofrecerlas sin un mecanismo que le permita al kernel agregar componentes con ciertas reglas.

- Linux ofrece el concepto de módulos.
- Mac OS X ofrece el concepto de extensiones (kexts).
- Windows trabaja con librerías.

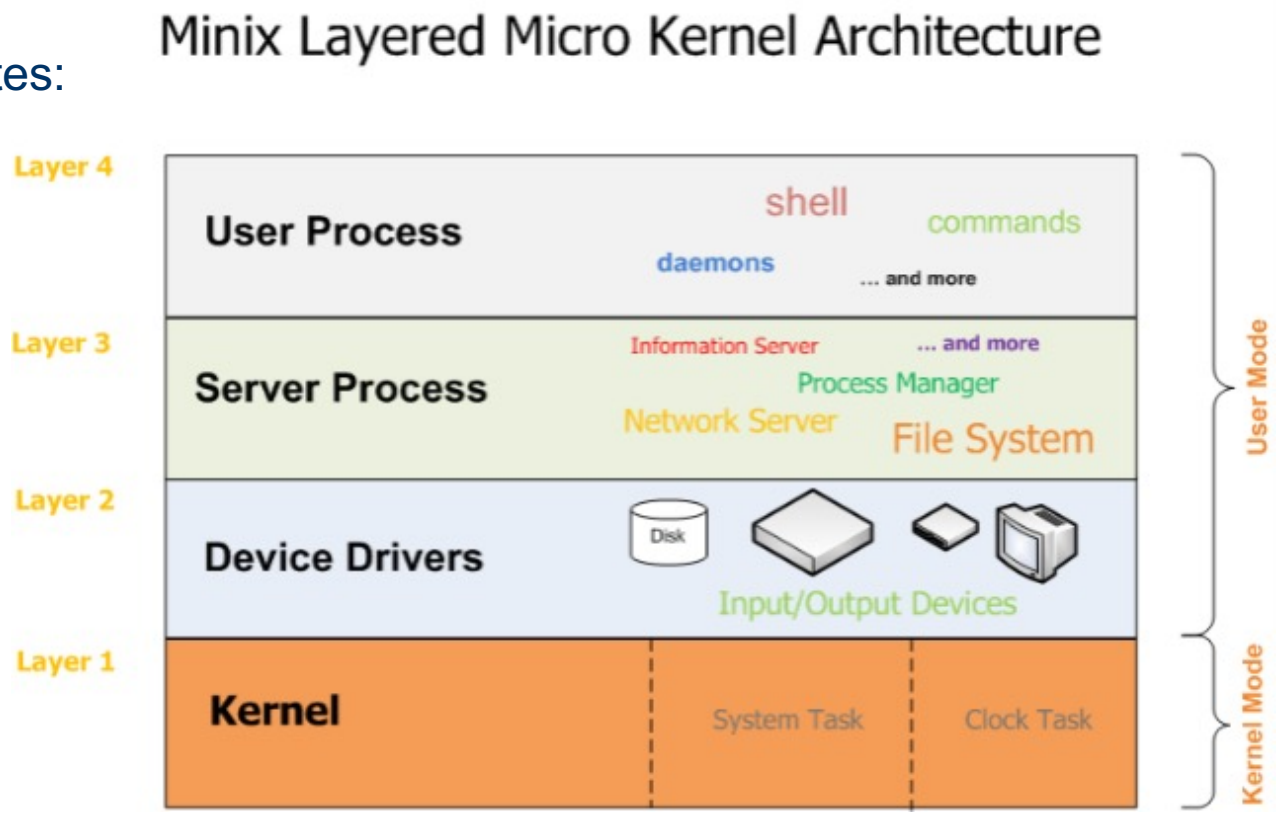
Introducción

Arquitectura del SO

Ejemplos Minix

Elementos importantes:

- Capa 2 y 3
- Permisos especiales.

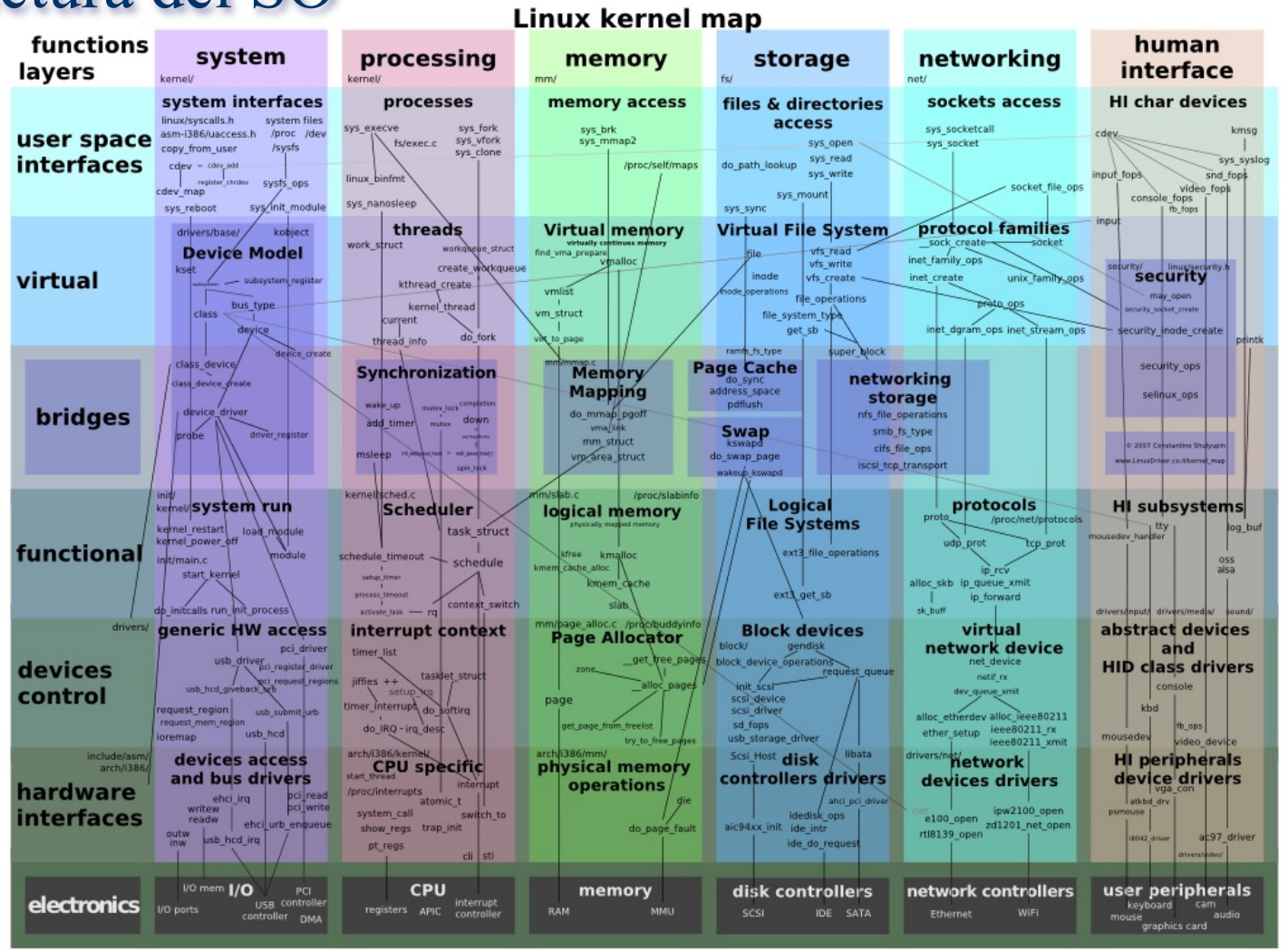


Introducción

Arquitectura del SO

Ejemplos Linux

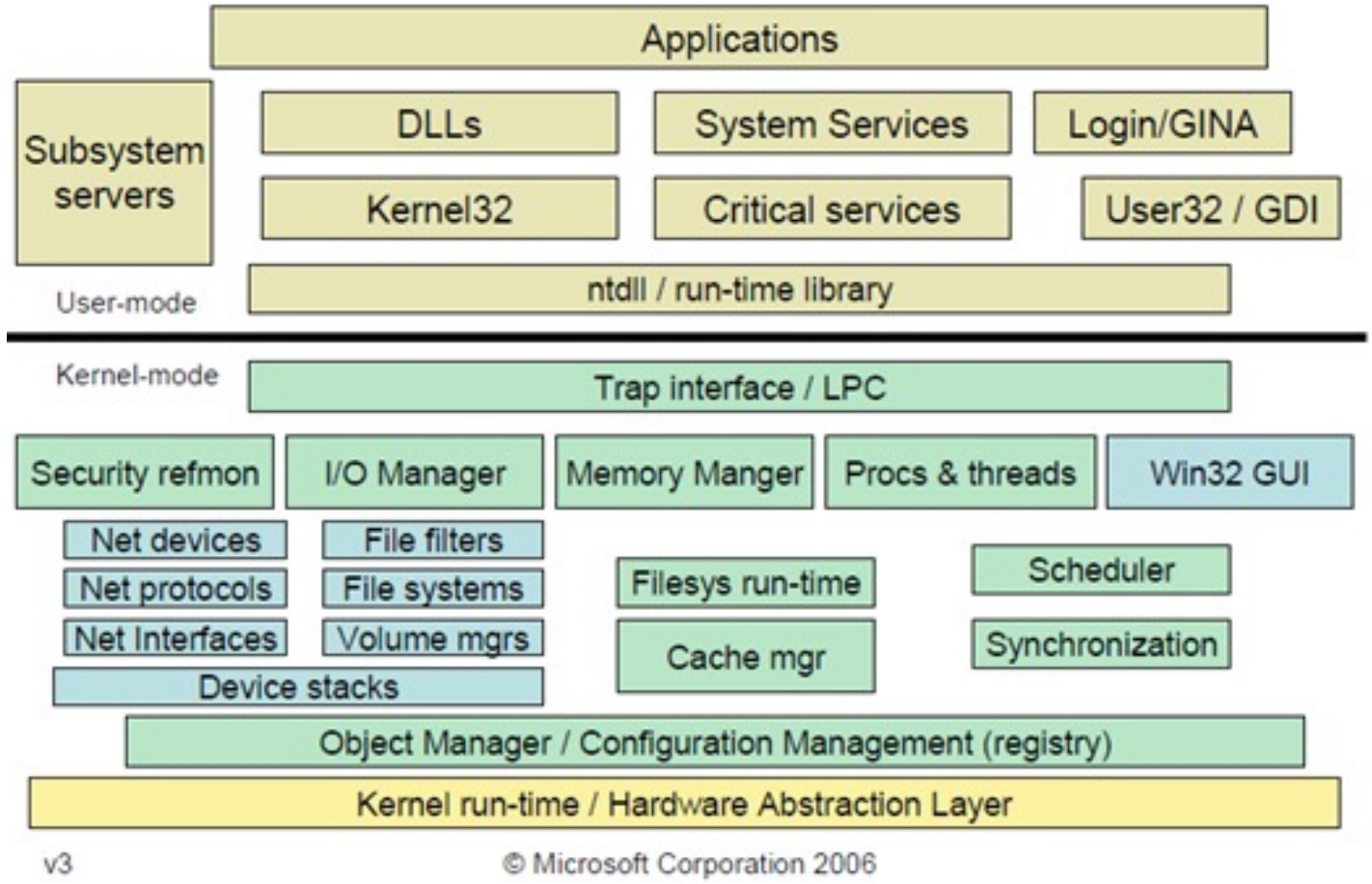
http://www.makelinux.net/kernel_map/.



Introducción

Arquitectura del SO

Ejemplos Windows



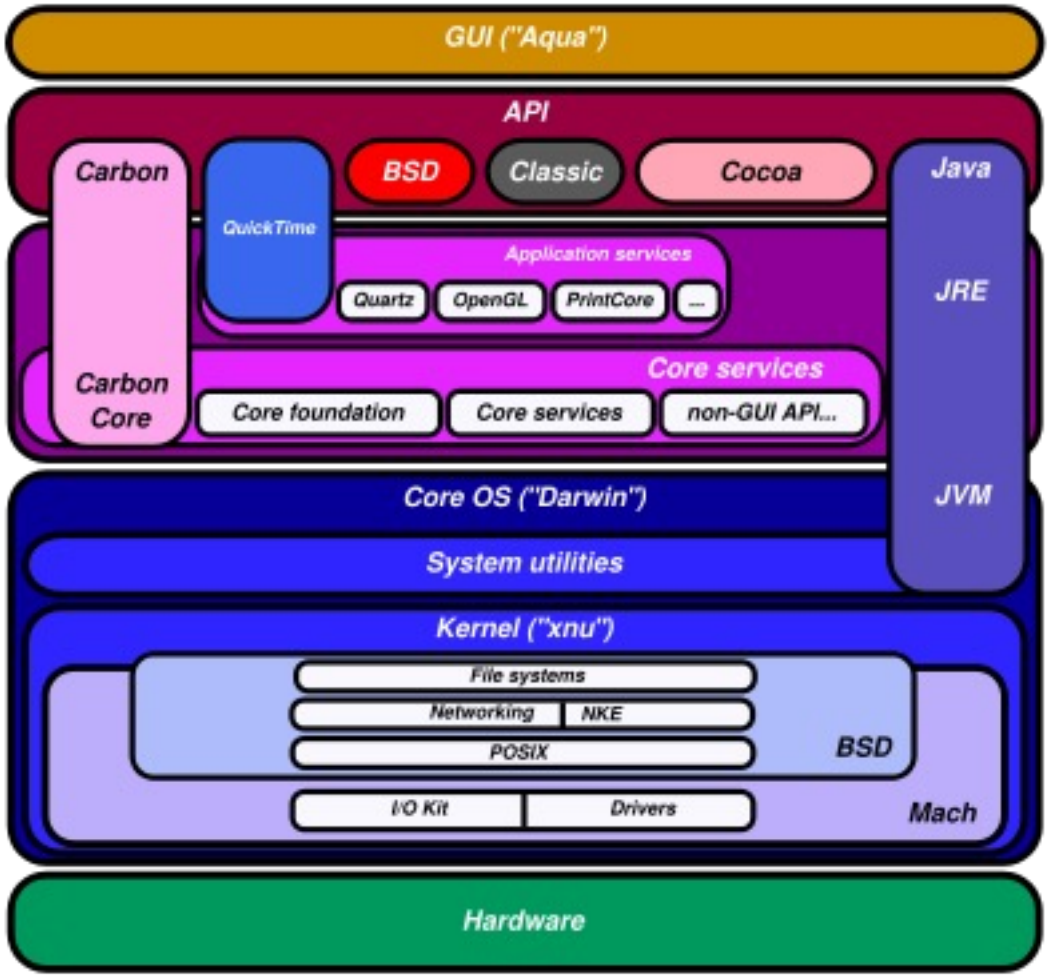
Introducción

Arquitectura del SO

Ejemplos Mac OX

Elementos importantes:

- Mach / BSD
- Darwing
- XNU
- Carbon
- Aqua



Introducción

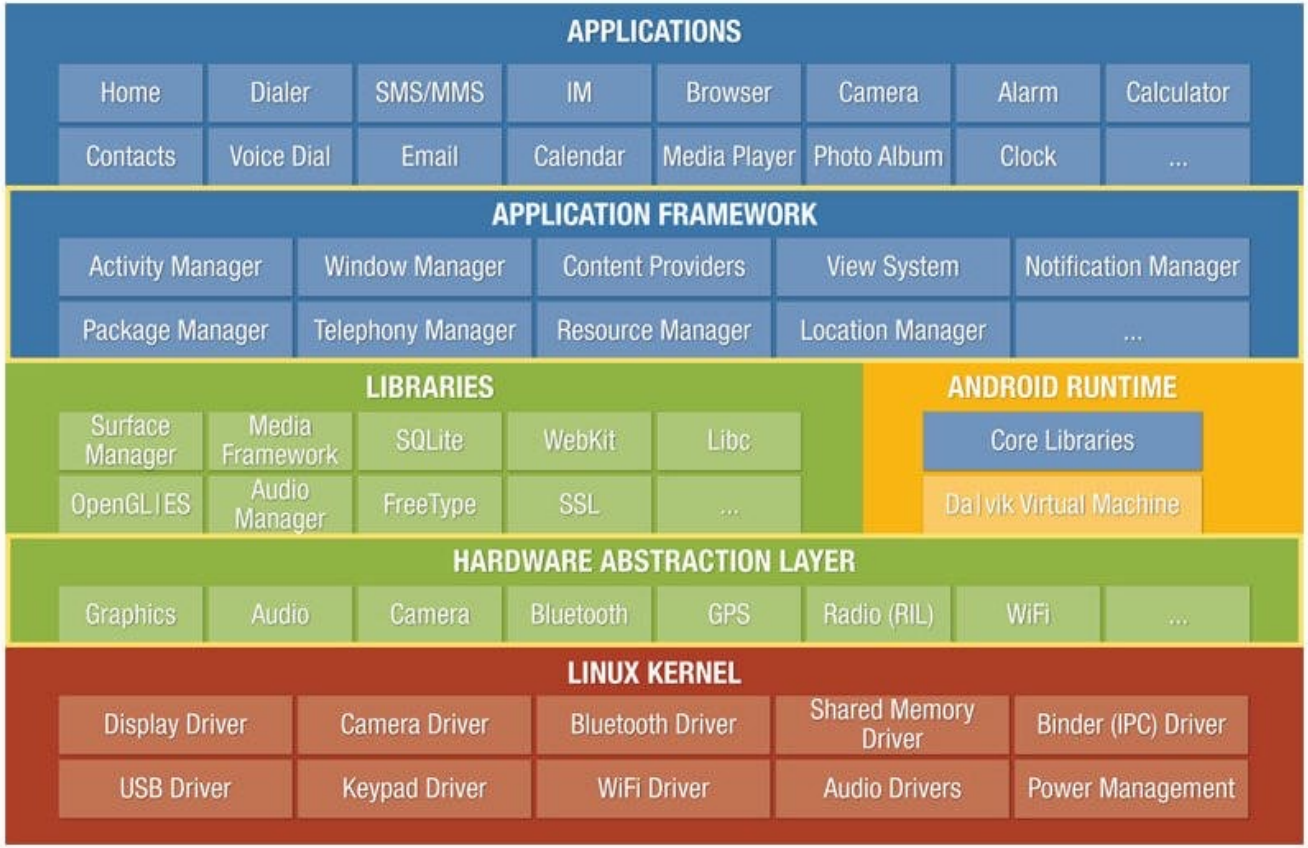
Arquitectura del SO

Ejemplos Android

Elementos importantes:

Otra máquina virtual.

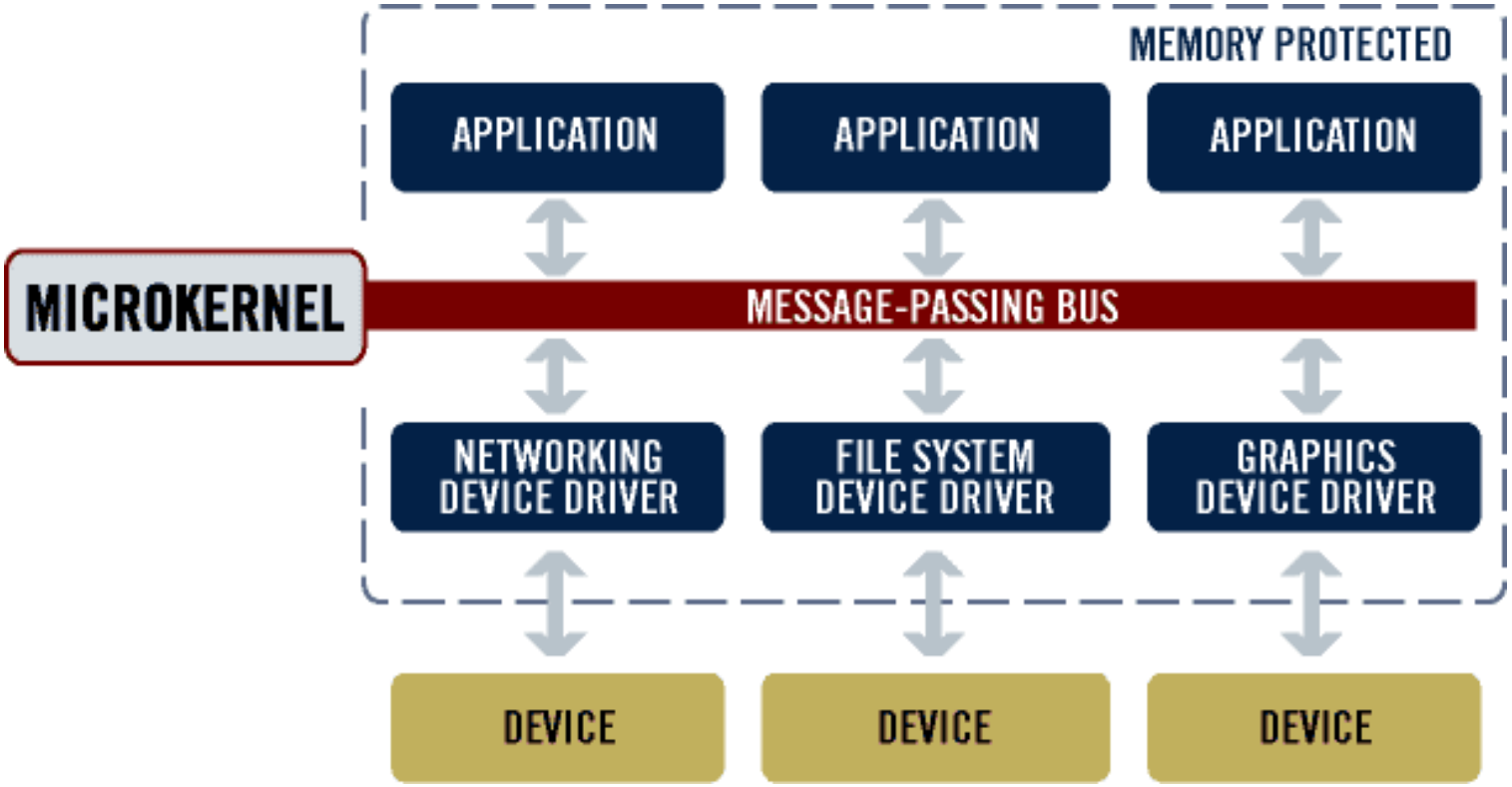
Otro ambiente gráfico (no es X).



Introducción

Arquitectura del SO

Ejemplos Qnx (real-time like-UNIX)



Introducción

Modelo a capas

Aplicaciones

Herramientas de Programación

Librerias, Compiladores, Comandos, etc.

Sistema Operativo

Adminitrador de Memoria, de Procesos

Sistema de Archivos y de Entrada/Salida

Hardware

CPU (procesador,Memoria, DD, Drives)

Perifericos(mouse, teclado), Monitor

Programación

Funciones en Librerias

`system()`

⇩

Llamadas al Sistema *(UNIX)*

Interrupciones *(DOS)*

`execl()`

Un modelo debe ser capaz de ocultar los detalles que no son relevantes para describir el comportamiento deseado.

Linux

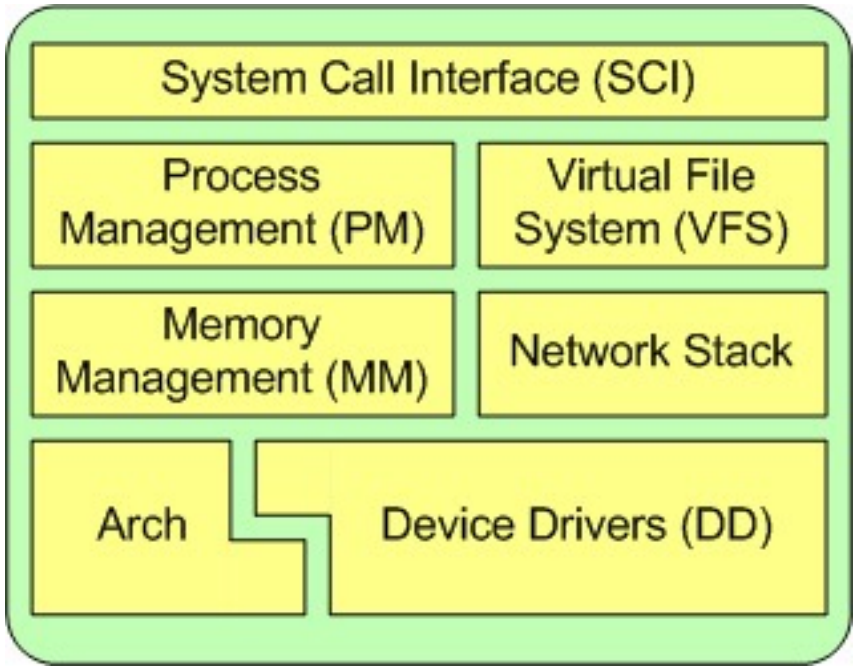
Arquitectura

Capas

Linux

Arquitectura

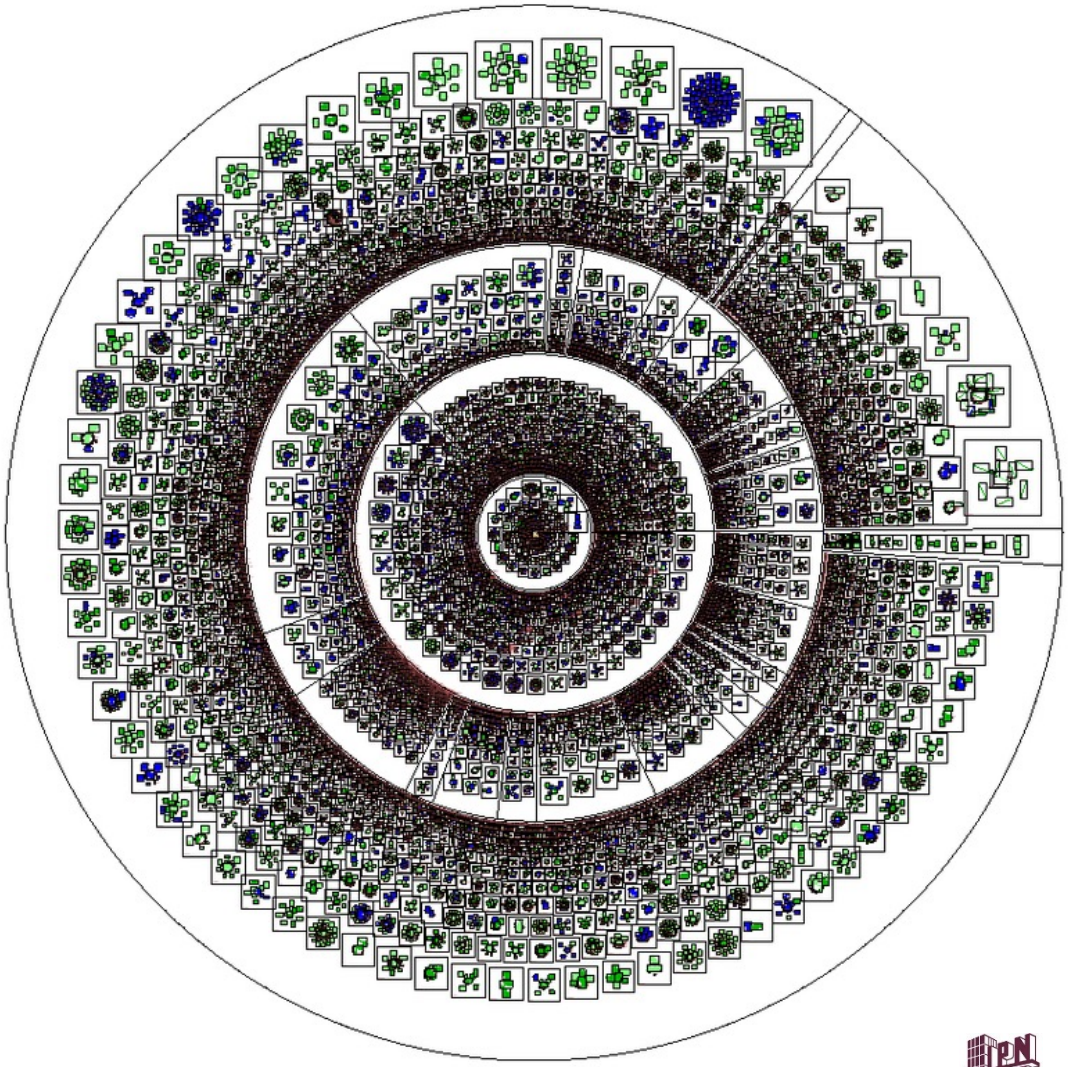
Imagen simplificada del kernel.



Linux

Modelo de cebolla

- The Linux kernel is 50% device drivers, and 25% CPU-specific code.
- The two inner layers are very generic.



Linux

Estructura de los archivos

FHS (*Filesystem Hierarchy Standard*)
Define como un estándar que detalla los nombres, ubicaciones, contenidos y permisos de los archivos y directorios, es decir, un conjunto de reglas que especifican una distribución común de los directorios y archivos en sistemas Linux.

/

- /bin (código binario o compilado de los programas/comandos que todos los usuarios puede usar)
- /boot (contiene lo necesario para que funcione el arranque del sistema)
- /dev (almacena las definiciones de todos los dispositivos)
- /etc (archivos necesarios para la configuración del sistema)
- /home (ubicación de las cuentas de los usuarios)
- /lib (contiene las librerias estáticas y dinámicas que usan los ejecutables)
- /mnt
- /sbin (igual que bin pero para ejecutables del administrador)

/root
/tmp
/opt
/usr
/usr/bin
/usr/lib
/usr/include
/var/…

Linux

Parámetros para el kernel

Como todo programa, el kernel tiene definidas varias **variables** y algunas de ellas deben tomar valores por default o ser inicializadas al arranque del sistema.

El kernel de Linux es capaz de recibir valores de estas variables como **parámetros** al arrancar (GRUB) y desde luego una vez en ejecución también se pueden modificar por comando y/o archivo de configuración.

Por ejemplo, la variable smt (*set maximum nimbre of threads*) es una de ellas.

Referencias

- https://www.kernel.org/doc/Documentation/kernel-parameters.txt

A continuación, se verá la forma de modificar las variables en tiempo de ejecución.

Linux

Sistema de Archivos Virtual

La carpeta **/proc**

OLD Inspección

Estructura básica:

- Varios archivos que en realidad son archivos virtuales (no existen en el DD)
  - version, cpuinfo
  - kallsyms
  - softirqs, interrupts
  - devices
  - ioports, iomem
  - Meminfo, pagetypeinfo
  - diskstats
  - cgroups
  - vmstat
  - …
- Varias carpetas cuyos nombres son números y representan los procesos.

El archivo **kcore** representa el kernel mismo.
Su tamaño de 128T no es real.

Linux

Sistema de Archivos Virtual

La carpeta **/sys**

Estructura básica:

- block
- bus
- class
- dev
- devices
- Firmware
- Fs
- hypervisor
- **kernel**
- module
- power

NEW: Introspección

- Do the same but more structured.
- But some information of proc is not here.
- Most operations of files can not be done on sys.

Linux

GRUB parámetros

Arranque con GRUB

Más adelante se verá a detalle el proceso de arranque y se verá el uso de GRUB pero por ahora nos limitaremos a decir que dado que GRUB es el programa que arranca al kernel de Linux es este el que le pasa **parámetros**.

Si no se quieren dar parámetros en su línea de comandos se pueden colocar en sus archivos de configuración que se encuentran en:

/etc/default/grub

Linux

Linux Standard Base (LSB)

Distribución de archivos

- The LSB was created to lower the overall costs of supporting the Linux platform.
- By reducing the differences between individual Linux distributions, the LSB greatly reduces the costs involved with porting applications to different distributions, as well as lowers the cost and effort involved in after-market support of those applications.
- Firts version (1.0) at 2001 and last version (**5.0**) at 2015.
- The LSB is registered as an official ISO standard. ISO/IEC 23360.

Links

- https://wiki.linuxfoundation.org/lsb/start
- https://en.wikipedia.org/wiki/Linux_Standard_Base
- http://refspecs.linuxfoundation.org/lsb.shtml

Algunas distribuciones no han implementado todo el LSB (debian)

Sistema Informático

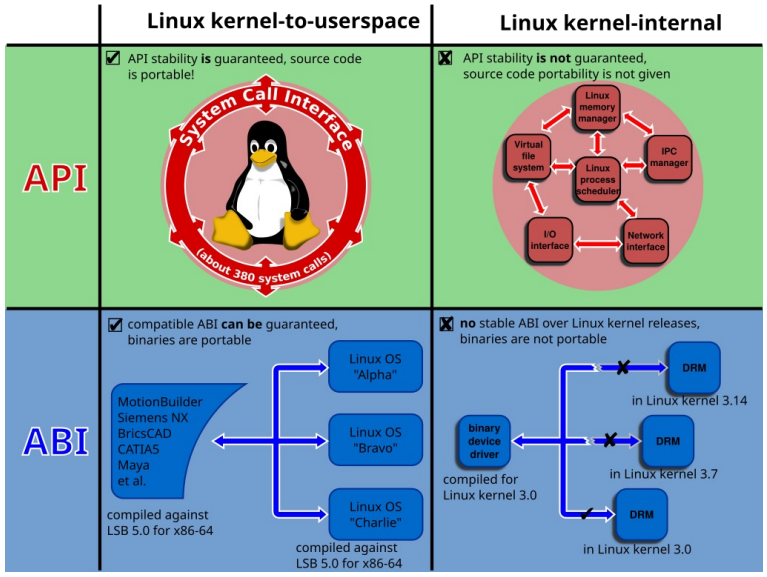
Interfaces

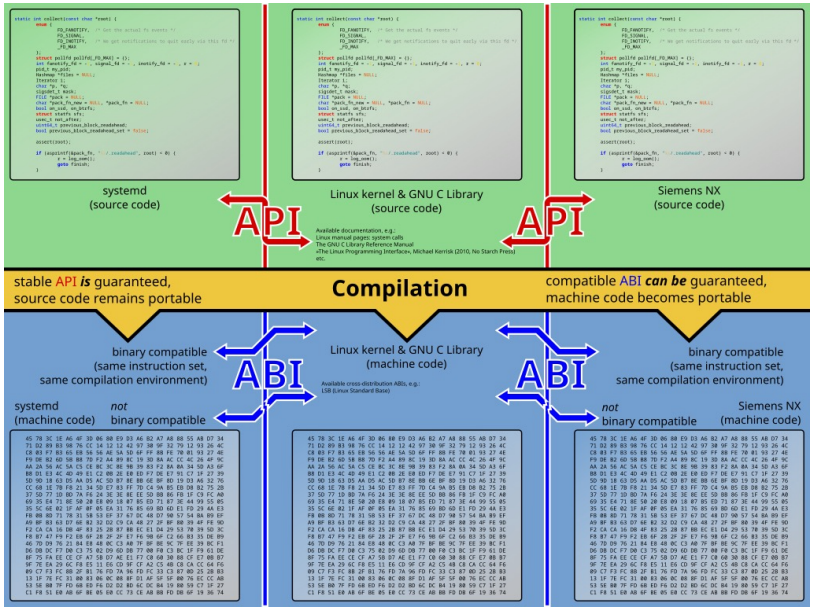
1. **GUI** (del inglés ***Graphical User Interface***)
   Es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en el sistema.
2. **CLI** (del inglés, ***Command-line interface***)
   Es un método que permite a los usuarios dar instrucciones a algún programa informático por medio de una línea de texto simple. Las CLI pueden emplearse interactivamente, escribiendo instrucciones en alguna especie de entrada de texto, o pueden utilizarse de una forma mucho más automatizada (archivo *batch*), leyendo órdenes desde un archivo de *scripts*.
3. **API** (del inglés ***Application Programming Interface***)
   Es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Linux

Application Binary Interface (ABI)

- ABI defines how data structures or computational routines are accessed in machine code, which is a low-level, hardware-dependent format.
- API, in contrast, defines this access in source code, which is a relatively high-level, hardware-independent, often human-readable format.





Linux

Componentes

Los principales

Linux

Componentes

Administrador de Procesos

- API

  Se programan mediante los syscalls:
  fork = > Procesos
  pthreads => Hilos

- CLI

  Algunos comandos principales:
  $ ps
  $ top

Linux

Componentes

Administrador de Memoria

- API

  Se programan mediante los syscalls:
  malloc => C
  new => C++
  free
- CLI

  Algunos comandos principales:
  $ top

Linux

Componentes

Network

- API

  Se programan mediante sockets.
- CLI

  Algunos comandos principales:

  $ netstat

  $ tcpdump

  $ ifconfig

Linux

Componentes

InterProcess Communication (IPC)

- API

  Se programan mediante:

  1. Pipes (tuberias).
  2. Segmentos de Memoria Compartida (Shared memory).
  3. Semáforos (mutex).
  4. Señales (signals).

- CLI

  Comando(s) principal(es)

  $ ipcs

Linux

Componentes

Storage

- API

  Se programan mediante syscalls:

  open, close, read, write, etc.

  fopen, flcose, fprintf, fscanf, etc.

- CLI

  Comando(s) principal(es)

  $ fdisk

  $ mount, umount

  $ dd

  $ lsblk

Libraries

Librerías

GNU

Linux

Librerías

Glibc

- Any Unix-like operating system needs a C library: the library which defines the ``system calls'' and other basic facilities such as open, malloc, printf, exit, ...
- The GNU C Library is used as *the* C library in the GNU system and in GNU/Linux systems, as well as many other systems that use Linux as the kernel.
- Current version 2.4 release date 2024-julio-22.

Links

- https://sourceware.org/glibc/
- https://www.gnu.org/software/libc/
- Manual: https://www.gnu.org/software/libc/manual/pdf/libc.pdf

Linux

Librerías

Glibc

- El manual tiene 1,295 páginas.
- Tiene varias secciones donde describe el API con elementos como:

  - Virtual Memory Allocation
  - String and array utilities
  - Search and sorting
  - Input/ouput (low-level, streams)
  - Job control
  - POSIX Threads
  - Encryption (DES)
  - Error reporting
  - File System
  - Pipes and FIFOs
  - Sockets
  - Mathematics
  - Date and time
  - Signal handling
  - Inter-Process Communication
  - Syslog

Daremos un vistazo al documento ahora …

Linux

Niveles, Demonios y Servicios

Linux

Niveles

- A *run level* is a state of **init** and the whole system that defines what system services are operating.
- Run levels are identified by numbers.
- Some system administrators use run levels to define which subsystems are working (e.g., X), whether the network is operational, and so on.
- Others have all subsystems always running or start and stop them individually, without changing run levels, since run levels are too coarse for controlling their systems.
- Configuration is in:
- /etc/inittab (information for each run level)
- /etc/init.d/rcN.d or /etc/rcN.d (scripts)
- Each distribution has its own command to configure the run level: chkconfig.
- Othe way to change level is using: # init number

Linux

Servicios/Demonios

System services are demons with special privileges.

Comandos

chkconfig –list
Show all the services
chkconfig –list
Show the information of a particular service
chkconfig *service_name* on|off
Activate/deactivate a particular service
chkconfig *service_name* on --level *runlevels*
Activate a service in a particular level
chkconfig –add *service_name*
chkconfig –del *service_name*

En varias distribuciones como RedHat (Fedora) existe el comando
service
con el mismo objetivo en otras existe
systemctl

Linux

Servicios

Common system services

- Systemd (demons for managing services in some distributions)
- Login (getty/local, rlogin/remote)
- Syslog
- Cron
- Mail
- Printing
- Network (Firewall)

Linux

Servicios

Comando service
It runs a System V init script.

```
# service --status-all
# service command [start, stop, restart]
# service httpd status
```

Linux

Servicios

Comando chkconfig

It updates and queries runlevel information for system services.

```
# chkconfig –list
…
network  0:off   1:off   2:off   3:on   4:off   5:off   6:off
...
# chkconfig name-service on
```

Linux

Servicios

Comando systemctl

Redhat/Centos

```
# systemctl
```

Comando systemd-cgtop

To view in real time the process associated with a particular service (cgroup).

```
# systemctl
```

Hardware Abstraction Layer (HAL)

Capa de Abstracción del Hardware

Obsoleto en general en Linux

pero ...

Se retomo en Android

HAL

Aun se usa en algunos SO

- It is a software subsystem for UNIX-like operating systems providing hardware abstraction.
- HAL is now **deprecated** on most Linux distributions and on FreeBSD. Functionality is being merged into udev on Linux as of 2008–2010 and devd on FreeBSD. Previously, HAL was built on top of udev.
- Some other OS-es which don't have an alternative like udev or devd, still use HAL.
- The purpose of the hardware abstraction layer was to allow **desktop applications to discover and use the hardware** of the host system through a simple, portable and abstract API, regardless of the type of the underlying hardware.

Linux kernel integrity

Herramientas

Linux kernel integrity

Herramientas

- Kernel rootkits that modify operating system state to avoid detection are a dangerous threat to system security.

Links

Papers

- Ensuring Operating System Kernel Integrity with OSck
  https://www.cs.utexas.edu/~sangmank/pubs/osck.pdf

Mac OS X

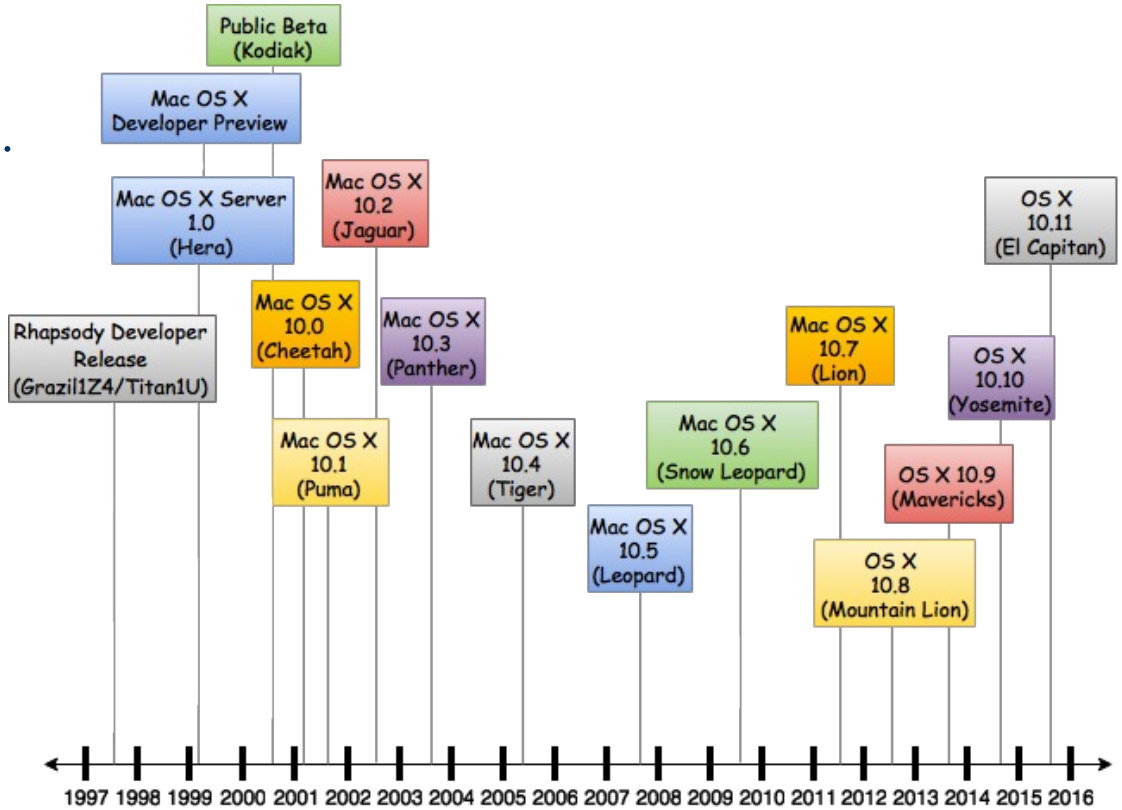
Macintosh Operating System Version X

Mac OS X

Versiones

Hay 2 grandes grupos de versiones:

- Classic Mac OS
  Que corre en 68k y PowerPC.
- Mac OS X



- **macOS** 10.12 "Sierra" was announced at WWDC 2016.
- OS X would be renamed to stylistically match Apple's other operating systems, such as iOS, watchOS, and tvOS.

Kernel de Mac OS X

Extensiones

Para ver las extensiones se usa el comando **kextstat** el cual despliega la siguiente información:

- Index: The load index of the kext (used to track linkage references). Gaps in the list indicate kexts that have been unloaded.
- Refs: The number of references to this kext by others. If nonzero, the kext cannot be unloaded.
- Address: The address in kernel space where the kext has been loaded.
- Size: The number of bytes of kernel memory that the kext occupies. If this is zero, the kext is a built-in part of the kernel that has a record as a kext for resolving dependencies among kexts.
- Wired: The number of wired bytes of kernel memory that the kext occupies.
- Architecture (if the -a option is used) The architecture of the kext.
- Name: The CFBundleIdentifier of the kext.
- Version: The CFBundleVersion of the kext.
- Linked Against: The index numbers of all other kexts that this kext has a reference to.

Las extensiones se encuentran en /System/Library/Extensions.

Kernel de Mac OS X

Extensiones

- El kernel del sistema operativo tiene definida una interfaz denominada **KPI** (Kernel Programming Interfaces). Para ver dichas KPI y sus versiones se usa:
  $ kextstat | grep “com.apple.kpi”

- Existen dos módulos del Kernel de los que normalmente dependen las kexts: com.apple.kpi.bsd y com.apple.kpi.libkern.
- En general el término kext (extensión al kernel) se utiliza para referirse a todo el bundle mientras que el término kmod (modulo del kernel) se utiliza para referirse al binario del kext que se carga en el kernel.
- Se puede verificar con kextutil que una extensión se encuentre en buen estado:
  $ sudo kextutil name.kext

Kernel de Mac OS X

Extensiones

- Para cargar y descargar una extensión se usa:
  $ sudo kextload name.kext
  $ sudo kextunload name.kext

Kernel de Mac OS X

Llamadas al sistema

- Las **llamadas al sistema** son la forma en que los procesos de usuario llaman a los servicios del kernel.
- En el kernel existe una tabla llamada **sysent** en la cual se encuentran indexadas las distintas llamadas al sistema.
- Cada elemento de la tabla sysent tiene una estructura defina en un struct.
  - De los campos del struct uno interesante es sy_call que apunta al código que se ejecuta con la llamada al sistema.
  - Los rootkits frecuentemente modifican este puntero para insertar su propia **función hook** que intercepta las llamadas al sistema de las operaciones que quiere modificar.
- A partir de Mac OS X 10.4, para dificultar la manipulación de la tabla sysent, el código fuente d**el kernel no exporta** este símbolo. En consecuencia, los kext no puede hacer referencia directa a este símbolo.

Kernel de Mac OS X

Llamadas al sistema

- A nivel ensamblador, una llamada al kernel tiene la siguiente forma:

```
mov eax, 1    ; SYS_exit
int 0x80
```

- El número en el registro eax indica el número de llamada a sistema a ejecutar cuando se ejecute la interrupción 0x80.
- En el fichero /usr/include/sys/syscall.h se detallan los números de las llamadas al sistema de que dispone el sistema operativo.

En mi MacBook Pro anterior decía que había:
500 syscall

Kernel de Mac OS X

Bitácora

- Todos los sistemas operativos modernos llevan un registro de lo que realizan (particularmente el kernel).
- Este registro (bitácora, log) suele estar fraccionado en varios archivos porque:
  - Crece con el tiempo.
  - Guarda información confidencial de ciertas componentes del sistema.
- En Mac OS dicha bitácora se encuentra en:

  /var/log/

  Donde a su vez hay varias carpetas y archivos con extensión ".log", por ejemplo: kernel.log o system.log
- Sobra decir que dicha bitácora es un elemento importante para el administrador del sistema y para que el malware oculte su actividad (rootkit).

Windows

Arquitectura y kernel

Windows

Interfaces

APIs in Windows

- If you know C# or Java, interfaces should be a familiar concept.
- An interface defines a set of methods that an object can support, without dictating anything about the implementation.
- The interface marks a clear boundary between code that calls a method and the code that implements the method.
- In computer science terms, the caller is decoupled from the implementation.
- In Windows the concept of interface is implemented with **COM** (*Component Object Model*).

Windows

Interfaces

APIs in Windows

- COM is a binary-interface standard for software components introduced by Microsoft in 1993.
- It is used to enable inter-process communication and dynamic object creation in a large range of programming languages (frameworks):

  OLE, OLE Automation, Browser Helper Object, ActiveX,
  COM+, DCOM, the Windows shell, DirectX,
  UMDF and Windows Runtime

- Any Windows program that uses COM must initialize the COM library by calling the CoInitializeEx function.
- Each thread that uses a COM interface must make a separate call to this function.

Windows

Kernel

Links

- https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_components
- https://msdn.microsoft.com/en-us/library/windows/desktop/ff818516(v=vs.85).aspx
- Windows Vista
    - http://www.nirsoft.net/kernel_struct/vista/

Windows

Kernel

Sysinternals System Information Utilities

1. **Autoruns**: See what programs are configured to startup automatically when your system boots and you login. Autoruns also shows you the full list of Registry and file locations where applications can configure auto-start settings.
2. **ClockRes**: View the resolution of the system clock, which is also the maximum timer resolution.
3. **Coreinfo**: It is a command-line utility that shows you the mapping between logical processors and the physical processor, NUMA node, and socket on which they reside, as well as the cache's assigned to each logical processor.
4. **Handle**: This handy command-line utility will show you what files are open by which processes, and much more.
5. **LiveKd**: Use Microsoft kernel debuggers to examine a live system.

Windows

Kernel

Sysinternals System Information Utilities

6. **LoadOrder**: See the order in which devices are loaded on your WinNT/2K system.
7. **LogonSessions**: List the active logon sessions on a system.
8. **PendMoves**: Enumerate the list of file rename and delete commands that will be executed the next boot.
9. **Process Explorer**: Find out what files, registry keys and other objects processes have open, which DLLs they have loaded, and more. This uniquely powerful utility will even show you who owns each process.
10. **Process Monitor**: Monitor file system, Registry, process, thread and DLL activity in real-time.

Windows

Kernel

Sysinternals System Information Utilities

11. **ProcFeatures**: This applet reports processor and Windows support for Physical Address Extensions and No Execute buffer overflow protection.
12. **PsInfo**: Obtain information about a system.
13. **PsLoggedOn**: Show users logged on to a system.
14. **PsTools**: The PsTools suite includes command-line utilities for listing the processes running on local or remote computers, running processes remotely, rebooting computers, dumping event logs, and more.
15. **RAMMap**: An advanced physical memory usage analysis utility that presents usage information in different ways on its several different tabs.
16. **WinObj**: The ultimate Object Manager namespace viewer is here.

Android

Arquitectura y kernel

Android

Arquitectura

Últimas versiones

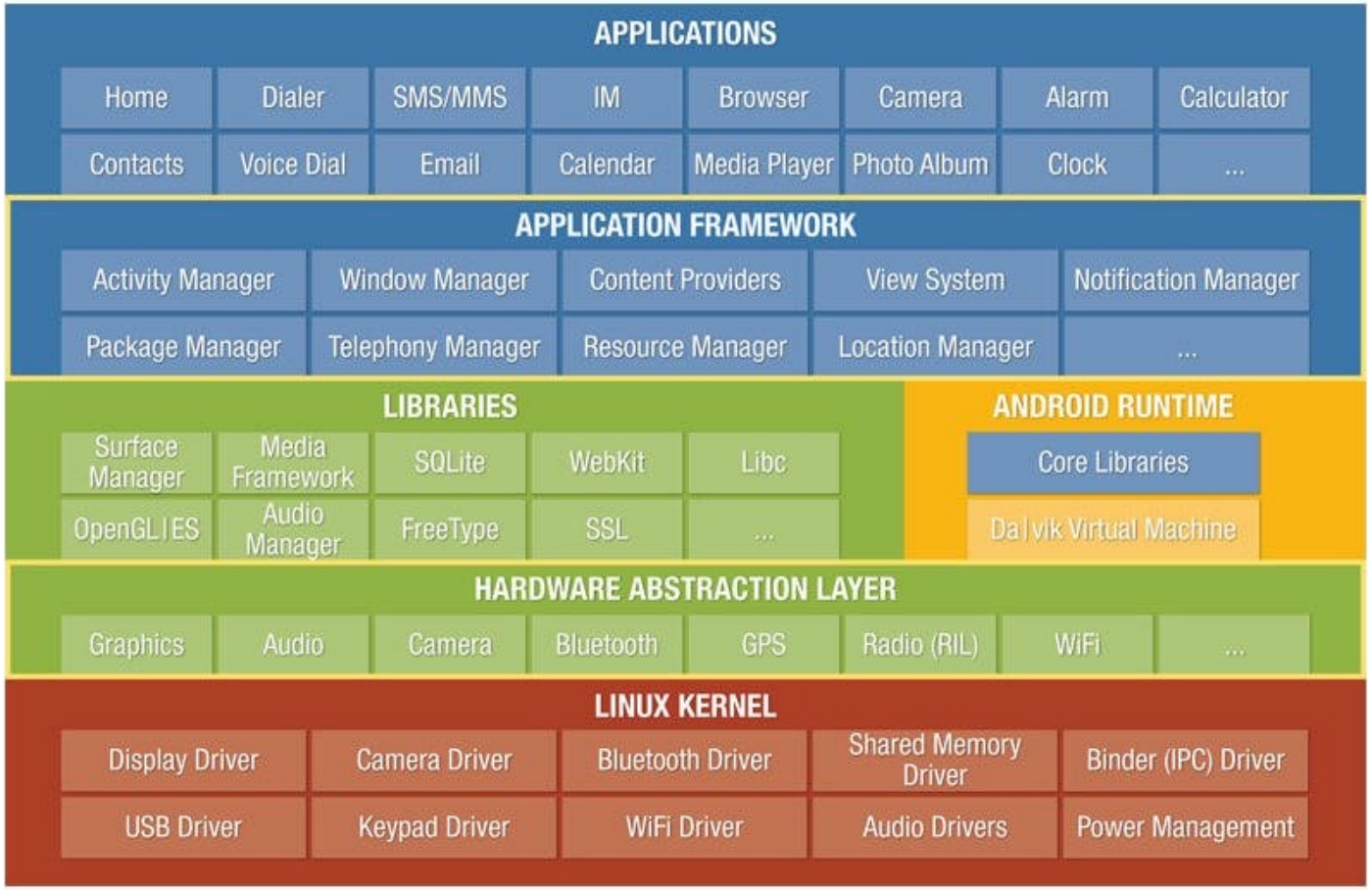
- Versión 14 (4/oct/2023), 15 es la Beta (13/ago./2024)
  - Major release is named in alphabetical order after a dessert or sugary treat. The first Android versions being called Cupcake, Donut, Eclair, Froyo, …, KitKat, Marshmallo, Nougat,etc.
  - https://www.android.com/versions/nougat-7-0/
- Código
  - https://source.android.com/

Diferencias con las distribuciones Linux

- Arranque: Fastboot
- ADB (Android Debug Bridge)
- GUI
- Toolchain / SDK Android
  - Android Studio (https://developer.android.com/studio/index.html)
  - Jack is a new Android toolchain that compiles Java source into Android dex bytecode.

Android

Arquitectura



Android

Arquitectura

Fastboot

- Es una pequeña herramienta que y puede ser usada para flashear particiones del dispositivo.
- Es un modo de recuperación alternativo al de recovery.
- El modo FastBoot arranca antes de cargar el SO Android (arranca incluso si Android no está instalado).
- Es el modo preferido para instalar la imagen de recovery en muchos dispositivos.
- También se usa para desbloquear el bootloader de los dispositivos Google Nexus.
- Instalación y ejecución de Fastboot: Viene con el Android SDK.
- No todos los dispositivos soportan fastboot, por ejemplo Heimdall(Linux) y Odin(Windows) para los terminales Samsung
- El comando fastboot se debe ejecutar siempre con privilegios de root.
- Los comandos de fastboot solo funcionan si el dispositivo está en modo fastboot.

Links

- http://elinux.org/Android_Fastboot.

Android

Arquitectura

ADB (Android Debug Bridge)

It is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device.

It is a client-server program that includes three components:

- A **client**, which sends commands. The client runs on your development machine. You can invoke a client from a shell by issuing an adb command. Other Android tools such as DDMS also create adb clients.
- A **daemon**, which runs commands on a device. The daemon runs as a background process on each emulator or device instance.
- A **server**, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

Standards

Estándares

Standards

Estándares

Los sistemas operativos suelen apegarse a varios de los siguientes:

- V7 Version 7. UNIX, released by AT&T/Bell Labs in 1979. After this point, UNIX systems diverged into two main dialects: BSD and System V.
- 3BSD (1980), 4BSD (1980) and 4.1BSD (1981).
- 4.2BSD Berkeley Software Distribution 4.2 release. It was released in 1983.
- 4.3BSD The successor to 4.2BSD, released in 1986.
- 4.4BSD The successor to 4.3BSD, released in 1993.
- System V This is an implementation standard defined by AT&T's milestone 1983 release of its commercial System V (five) release.
- The previous major AT&T release was System III, released in 1981.
- System V release 2 (SVr2), 1985.
- System V release 3 (SVr3) 1986.
- System V release 4 (SVr4) 1989. SVID 4 System V Interface Definition version 4, issued in 1995.

Standards

Estándares

The term "POSIX" was coined by Richard Stallman. "Portable Operating System Interface for Computing Environments"

- It was also ratified by ISO in 1990 (ISO/IEC 9899:1990)
- C99 Published in 1999 (ISO/IEC 9899:1999).
- C11 Published in 2011 (ISO/IEC 9899:2011).
- **POSIX**:

POSIX.1-1990 IEEE 1003.1-1990 , ISO/IEC 9945-1:1990)
POSIX.2 IEEE Std 1003.2-1992, ISO/IEC 9945-2:1993
POSIX.1b IEEE Std 1003.1b-1993, POSIX.1c IEEE Std 1003.1c-1995
POSIX.1d IEEE Std 1003.1c-1999, POSIX.1g IEEE Std 1003.1g-2000
POSIX.1j, POSIX.1-1996
SUS (SUSv1), SUSv2, POSIX.1-2001, SUSv3, POSIX.1-2008, SUSv4
XPG3, XPG4, XPG4v2

The end

Contacto

Raúl Acosta Bermejo

http:www.cic.ipn.mx
http://www.ciseg.cic.ipn.mx/

racostab@ipn.mx
racosta@cic.ipn.mx

57-29-60-00
Ext. 56652