Módulo 3. Fundamentos de Sistemas Operativos

Francisco Medina López — correo@franciscomedina.net

7 de enero de 2011

Centro Tlatelolco de Extensión en Cómputo y Telecomunicaciones
Universidad Nacional Autónoma de México

1. Teoría de Sistemas Operativos
2. Administración de Unix/Linux
3. Administración de Microsoft Windows Server
4. Servicios Web

Teoría de Sistemas Operativos

1. Teoría de Sistemas Operativos
   - Introducción
   - Hardware
   - Firmware
   - Software
   - Sistema Operativo
   - Sistemas Abiertos y Cerrados
   - Software Libre

Objetivos

Al término de este tema será capaz de:

- Clasificar correctamente el software.
- Identificar los componentes de un sistema de cómputo.
- Definir un sistema operativo.
- Diferenciar entre un proceso, llamada al sistema y archivo.
- Conocer el software libre

Introducción

Arquitectura y Sistema de Cómputo

Arquitectura de Computadoras

Disciplina de la Ingeniería, relacionada con el diseño y la construcción de sistemas de cómputo a nivel lógico.

Arquitectura de Cómputo

Comprende la **estructura** de un sistema de cómputo.

Sistema de Cómputo

Conjunto formado por **hardware**, **firmware**, **software**, medios de almacenamiento, datos o información y personas involucradas.

Arquitectura de Computadoras

Definición

La Arquitectura de Computadoras comprende todas y cada una de las partes necesarias para que un **sistema de cómputo** funcione

Incluye:

- Sistema Operativo
- Chips de Memoria
- Circuitos
- Discos Duros
- Componentes de Seguridad
- Conexiones Bus
- Componentes de Red
- Etc.

Computadora vs Servidor

Computadora

Máquina digital programable.

Servidor

En una red cliente/servidor, son los equipos que proveen información (datos) y servicios (impresión de documentos, transferencia de archivos, correo electrónico, bases de datos, ...) a las estaciones de trabajo (clientes).

Hardware

Definición

Hardware

Conjunto de componentes *tangibles* (o físicos) de una computadora. [a]

[a] http://www.carlospes.com/minidiccionario/hardware.php

Componentes principales de la plataforma de hardware en la Arquitectura de Computadoras:

- CPU (Unidad Central de Procesamiento)
- Memoria
- Bus de conexiones
- Dispositivos de Entrada/ Salida (I/O)
- Dispositivos de Almacenamiento

CPU

Unidad Central de Proceso (CPU)

También conocido como **procesador**, es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de la computadora. [a]

[a] http://es.wikipedia.org/wiki/Unidad_central_de_procesamiento

Contiene:

1. **Almacenamiento primario (Registros)**
2. **Unidad de Control (UC)**
3. **Unidad de Aritmético Lógica (ALU)**
4. **Unidad de Administración de Memoria (MMU)**

Componentes principales del CPU

1. **Almacenamiento primario**
   - Registros[1] que almacenan instrucciones y datos que van a ser procesados
2. **Unidad de Control**
   - Coordina la actividad durante la ejecución de instrucciones de un programa
   - No procesa datos, solo controla los procesos que se están ejecutando
3. **Unidad de Aritmético Lógica**
   - Realiza operaciones matemáticas y lógicas
4. **MMU**
   - Manipula direcciones y catálogos de datos almacenados en memoria además de convertir las direcciones lógicas en físicas

[1] Memoria intermedia

Características principales de un Procesador

- Reducido SET de instrucciones
- MIPS (Millones de Instrucciones por Segundo)

Estructura interna de un Procesador

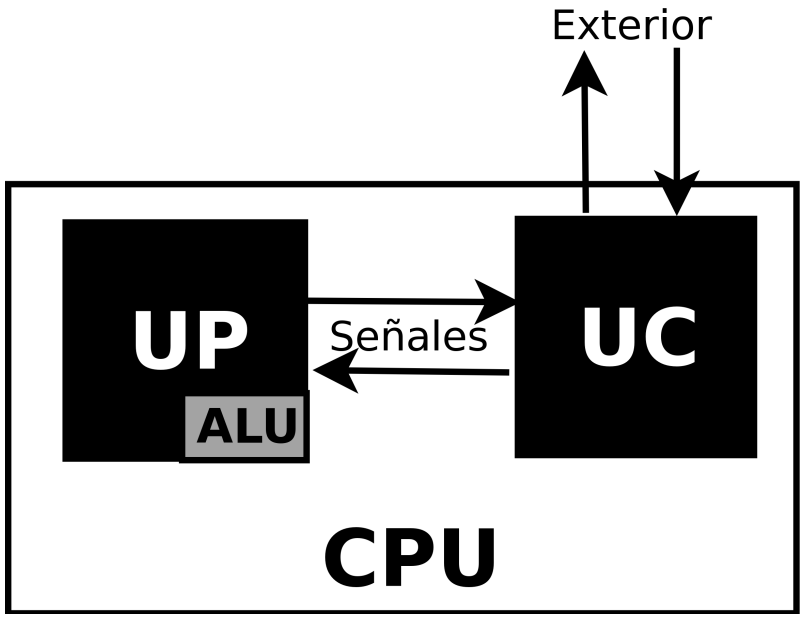
UC (Unidad de Control): La Unidad de Control es la encargada de gestionar y controlar el correcto funcionamiento de la Unidad de Proceso, indicando cuando una instrucción debe ser enviada al procesador. La UC no procesa datos, tan solo actúa como agente de tránsito.

UP (Unidad de Proceso): Formada por componentes tales como: la ALU, Registros, y buses.

ALU (Unidad Aritmético-Lógica): Encargada de llevar a cabo funciones matemáticas y operaciones lógicas.

Registros: Almacenan datos durante cierto tiempo, dentro la CPU.

Bus: Conjunto de circuitos y conectores



"Podríamos decir que la ALU es el cerebro del procesador, y el procesador el cerebro de la computadora."

Operación Básica de un Procesador

- La operación de un procesador consiste básicamente en dos fases: *obtener (fetch)* y *ejecutar (execute)*.
  - Durante la fase de **obtención**, la CPU localiza y recupera las instrucciones de memoria.
  - Durante la fase de **ejecución**, la CPU decodifica y ejecuta las instrucciones.
- Estas dos fases componen lo que se llama *ciclo de reloj* (clock singals).
- A los programas ejecutados por el procesador se llaman *procesos*.

Estados del procesador

La CPU, se encuentra siempre en alguno de los siguientes estados principales:

- **User State** En este estado, solo pueden ser ejecutadas instrucciones no- privilegiadas.
- **Supervisor State / Privileged Mode** En este estado, pueden ser ejecutadas tanto instrucciones no-privilegiadas como privilegiadas.

Procesos

- Un proceso es un programa en *ejecución*, el cual se encuentra compuesto de código ejecutable, datos e información relativa su ejecución.
- Un proceso trabaja en su propio espacio de direcciones y puede comunicarse con otros procesos, solo a través de pasos autorizados por el sistema operativo.
- Los "estados de un proceso" no es lo mismo que los "estados de la CPU". Mientras que los "estados de la CPU" define el modo operativo de la CPU, los "estados de un proceso" se refiere al modo en que los procesos se encuentran corriendo dentro de esta.
- El estado de los procesos o "Process State", es particular de cada sistema operativo. Ready, Waiting, Running y Stopped son solo algunos de los estados mas comúnmente encontrados.

Procesos vs Hilos

Un proceso es un programa en ejecución que posee su propio espacio de trabajo, y solo puede comunicarse con otro proceso de modo controlado.

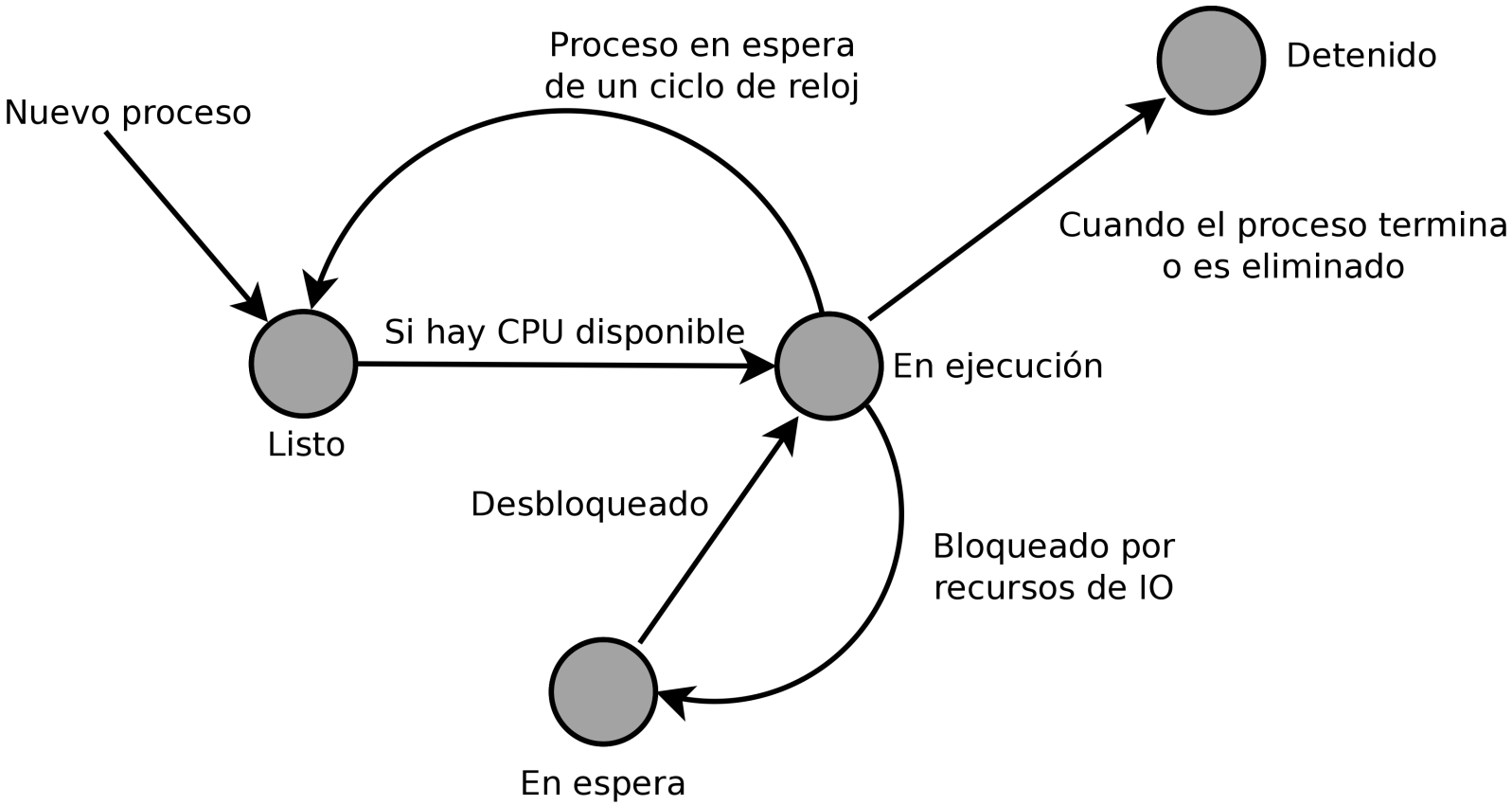
Un hilo (*thread*) en cambio, representa una pieza de código que esta siendo ejecutada dentro de un proceso.

- Un proceso puede incluir uno o mas Threads.

Estados de un proceso

- **Detenido**
  - El proceso no se encuentra en ejecución
  - Pudo haber sido detenido por el sistema operativo o un usuario
- **En espera**
  - El proceso se encuentra esperando por una interrupción (generalmente de IO) que le permita ser de nuevo procesado por la CPU
  - Estas interrupciones permiten compartir el *tiempo de procesamiento* de la CPU
- **En ejecución**
  - Las instrucciones del proceso están siendo ejecutadas por la CPU
  - Se le conoce también como *tiempo de ejecución*
- **Listo**
  - El proceso listo para ser utilizado y a la espera de una instrucción

Estados de un proceso



Diseños de CPU

Actualmente existen dos tipos de diseños ampliamente extendidos:

- **Complex-Instruction-Set-Computing (CISC)**: Conjunto de instrucciones que se caracteriza por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos
- **Reduced-Instruction-Set-Computing (RISC)**: Filosofía de diseño de CPU que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse.

Cuando se ejecuta un programa difícil, o extenso, los CISC son más rápidos y eficaces que los RISC. En cambio cuando se tiene en ejecución un conjunto de instrucciones sencillas, cortas y simples, los RISC son más rápidos.

Características de la CPU

- **Scalar Processor:** Procesador que ejecuta una instrucción por vez.
- **Superscalar Processor:** Procesador que permite la ejecución de varias instrucciones en la misma etapa del pipeline, como así también en diferentes etapas de pipeline. (IBM RS/6000)
- **Multitasking:** Ejecución de dos o mas tareas al mismo tiempo utilizando un solo CPU. Coordinado por el SO (Windows 2000, Linux, OS/3

Características de la CPU (II)

- **Multiprogramming:** Ejecución simultánea de dos o más programas utilizando un solo CPU.
  - A diferencia de lo que ocurre con Multitasking, cuya implementación suele encontrarse en sistemas operativos de PC tales como Linux y Windows, Multiprogramming suele encontrarse generalmente en mainframes o sistemas legacy.[2]
  - Multitasking es normalmente coordinado por el sistema operativo, mientras que Multiprogramming requiere que el software se encuentre especialmente escrito para coordinar sus propias acciones a través del sistema operativo.

[2] Un sistema heredado (o sistema legacy) es un sistema informático (equipos informáticos y/o aplicaciones) que ha quedado anticuado pero continúa siendo utilizado por el usuario (típicamente una organización o empresa) y no se quiere o no se puede reemplazar o actualizar de forma sencilla.

Características de la CPU (III)

- **Multiprocessing:** Ejecución simultánea de dos o más programas en múltiples CPUs.
- **SMP (Symmetric Multiprocessing)** Una computadora, con mas de un procesador, controlado por un solo sistema operativo (Bus de Datos y Memoria Compartidos).
- **MPP (Massively Parallel Processing)** Cientos o miles de procesadores, cada uno de los cuales utiliza su propio juego de recursos (sistema operativo, bus de datos y memoria).

Características de la CPU (IV)

- **Multithreading:** Ejecución de múltiples tareas al mismo tiempo utilizando un solo CPU. A diferencia de lo que ocurre con Multitasking donde las diferentes tareas ocupan diferentes "procesos", Multithreading permite ejecutar varias tareas en un solo "proceso".

  Quizás un buen ejemplo de Multithreading, sea cuando abrimos varios documentos de Word, lo cual no genera múltiples instancias de Word, precisamente porque todas ellas corren en un solo proceso utilizando diferentes hilos.

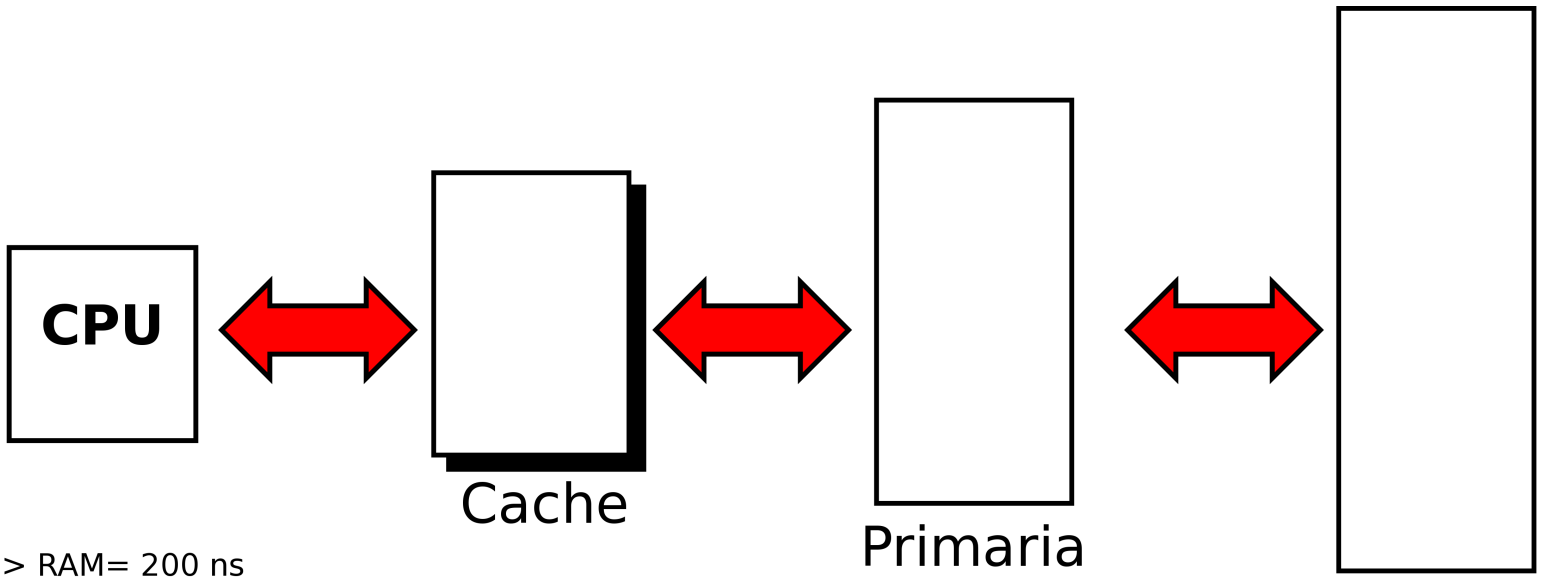
Memorias

- **Cache**
- **Flash Memory**
- **RAM (Random Access Memory)**
  - Dynamic Random Access Memory (DRAM)
  - Extended Data Output RAM (EDO RAM)
  - Synchronous DRAM (SDRAM)
  - Double Data Rate SDRAM (DDR SDRAM)
  - Burst Extended Data Output DRAM (BEDO DRAM)
- **ROM (Read Only Memory)**
  - Programmable Read Only Memory (PROM)
  - Erasable Programmable Read-Only Memory (EPROM)
  - Electrically Erasable Programmable Read-Only Memory
  - (EEPROM)

Memorias (II)

- **Memoria Primaria (Real Memory – Primary Storage)** Directamente accesible por la CPU y frecuentemente utilizada al momento de almacenar datos e instrucciones asociados con el programa que se encuentra en ejecución. Generalmente RAM.
- **Memoria Secundaria (Secondary Storage)** Almacenamiento no volátil, mas lento que la memoria primaria. Ejemplo: Discos Duros, CDs, DVDs, Floppys.
- **Memoria Virtual (Virtual Memory – Virtual Storage)** Combinación de memoria primaria y secundaria. Define un único espacio de direccionamiento. Habilidad para extender el tamaño aparente de la memoria RAM usando parte del disco rígido. (Swapping / Paging)

Jerarquía de acceso a memoria



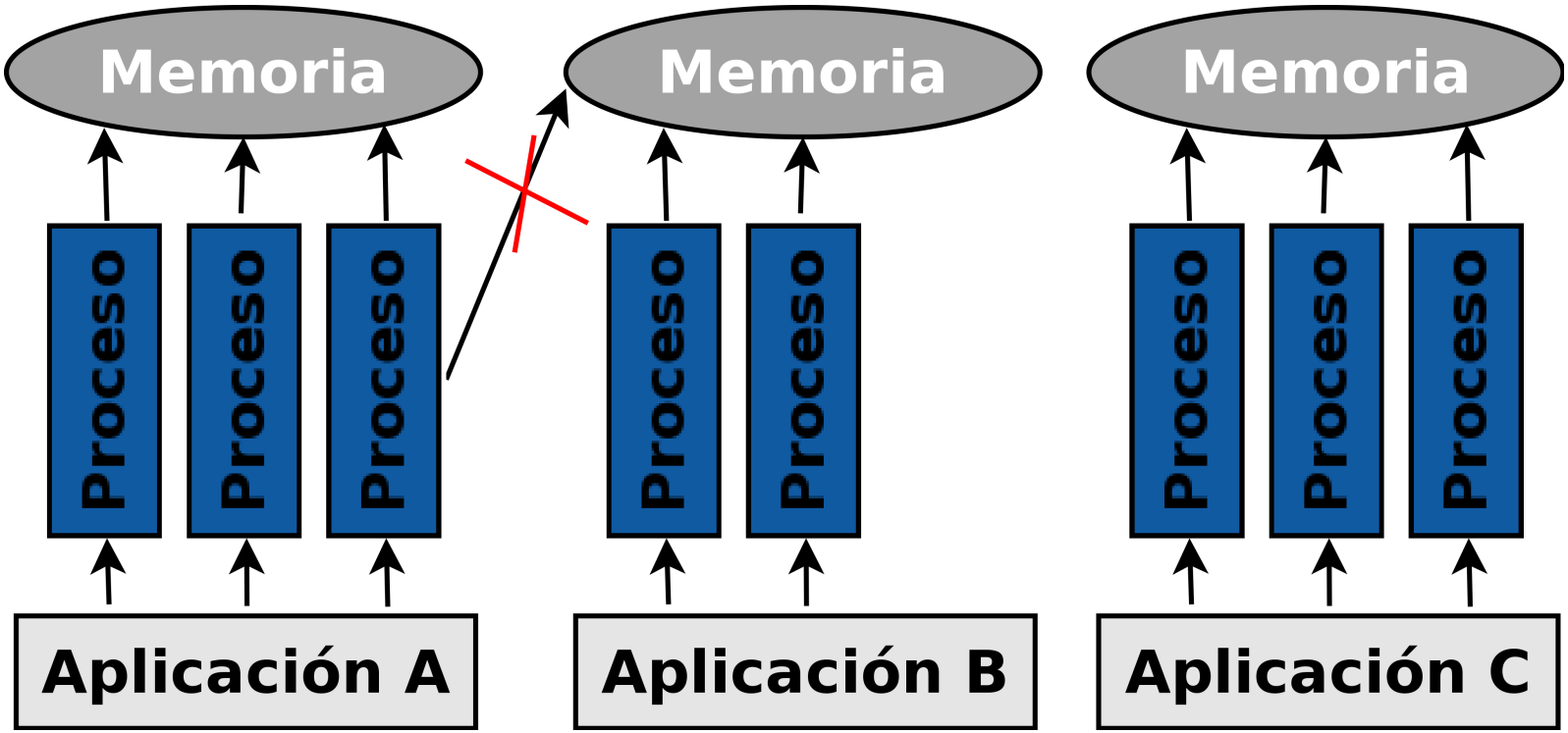
CPU -> RAM= 200 ns
CPU -> Disco Duro = 12,000,000 ns

Direccionamiento de Memoria

- Cuando se utilizan recursos de memoria, el procesador debe tener alguna forma de referirse a los diferentes lugares existentes dentro de ella. La solución a este problema, se conoce como *"Direccionamiento"* por su termino en ingles *"Addressing"*.
- Tipos:
  - Por Registro (Register Addressing)
  - Directo (Direct Addressing)
  - Absoluto (Absolute Addressing)
  - Indexado (Indexed Addressing)
  - Implícito (Implied Addressing)
  - Indirecto (Indirect Addressing)

Protección de Memoria

- Previene el acceso de un programa al espacio de memoria reservado para otro programa
- Se implanta a nivel de sistema operativo o hardware



Estructura de E/S

- **Input/Output (I/O) interface adapters:** Permiten la comunicación entre el procesador y los dispositivos externos
  - **Memory-Mapped I/O:** Se otorga una dirección de memoria "central" al dispositivo
  - **Isolated I/O:** Una señal especial en el bus de comunicación indica la ejecución de una operación de I/O. No utiliza memoria "central"
  - **Direct Memory Access (DMA):** Data es transferida en forma directa desde y hacia la memoria, no requiere del CPU
  - **Interrupt Processing:** Una señal externa interrumpe el flujo normal del programa para requerir servicio

Bus

- **Bus:** Circuitos impresos (o bien cables) que transmiten los datos del procesador.
  - **de transmisión de datos:** líneas físicas por dónde circulan los datos que se han leído o que se van a escribir (entrada/salida).
  - **de direcciones:** líneas físicas por dónde circulan las direcciones de memoria desde dónde se leerán (entrada), o se escribirán (salida), los datos.
  - **de control:** líneas físicas por dónde circulan las órdenes de control (entrada/salida).

Firmware

Firmware

Defínción

Bloque de instrucciones de programa para propósitos específicos, grabado en una memoria de tipo no volátil (ROM, EEPROM, flash,...), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo.

Al estar integrado en la electrónica del dispositivo es en parte *hardware*, pero también es *software*, ya que proporciona lógica y se dispone en algún tipo de lenguaje de programación. [3]

[3] http://www.carlospes.com/minidiccionario/software.php

Software

Definición

Software

Conjunto de programas y datos con los que trabaja una computadora el cual es inmaterial (o lógico).[a]

[a] http://www.carlospes.com/minidiccionario/software.php

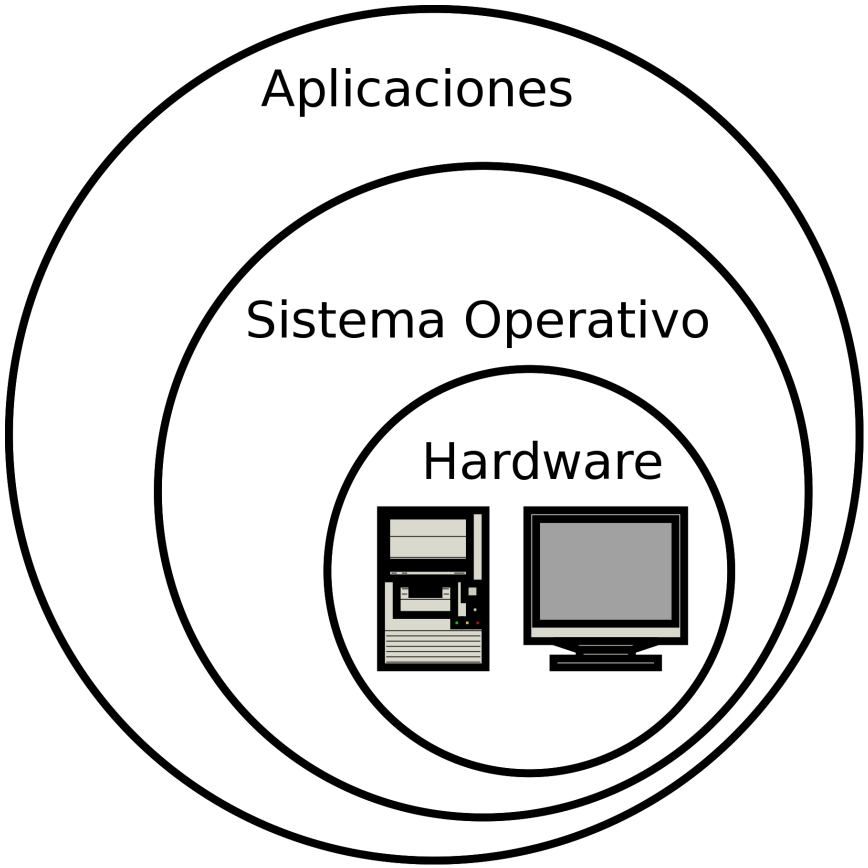
Clasificación del Software

1. **Programas de sistema** (*software de sistema*):controlan la operación de la computadora.
   - Compiladores
   - Sistema Operativo
2. **Programas de aplicación** (*software de aplicación*): resuelven problemas para los usuarios.
   - RDBMS
   - Sistemas
   - Juegos

Sistema Operativo

¿Qué es un Sistema Operativo?

Es un intermediario (interfaz) entre los programas y la computadora. Se puede definir de dos formas ligadas a sus objetivos.



Sistema Operativo (II)

- Software principal de toda plataforma de computo.
- Este es cargado en la computadora por medio de un programa denominado Boot, nombre con el que solemos referirnos al proceso responsable de la carga del sistema operativo.
- Grandes computadoras o *mainframes*, utilizan una secuencia boot conocida como IPL (Initial Program Load).
- Durante toda secuencia de *booteo*, un pequeño programa es cargado en memoria, el cual una vez inicializado realiza la carga total del sistema operativo.
- Una vez en ejecución, el sistema operativo es el responsable de controlar varios subsistemas tales como las utilidades de software, aplicaciones, sistemas de archivos, control de acceso, etc.

Sistema Operativo (III)

Todo sistema operativo persigue dos objetivos principales:

- Controlar el uso de los sistemas y recursos.
- Proveer de una interfaz entre usuario y computador *(máquina extendida)*

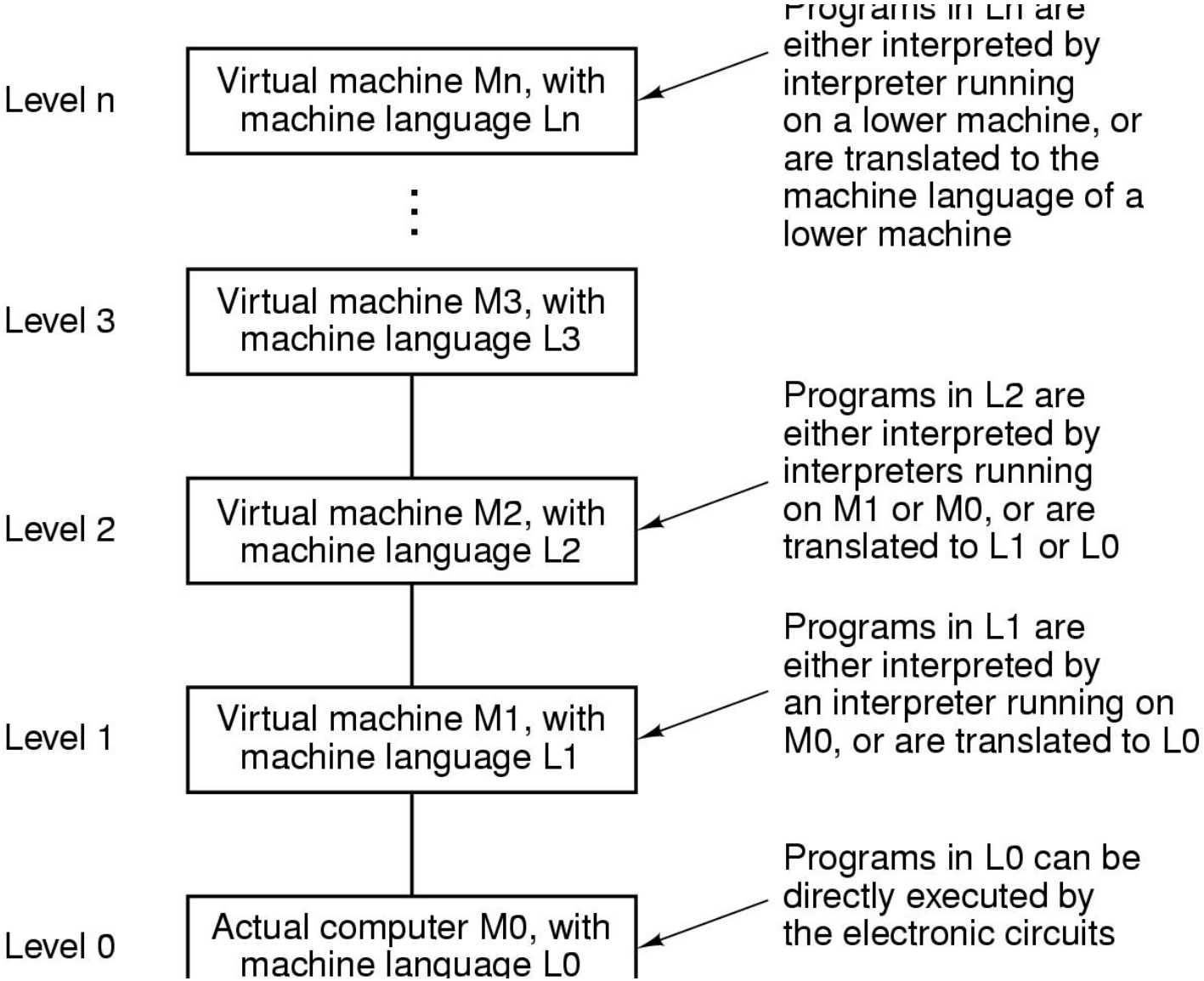
Como administrador de recursos

Es el encargado de proporcionar una asignación ordenada y controlada de los recursos (CPU, memoria y disco) para los varios programas que compiten por ellos.

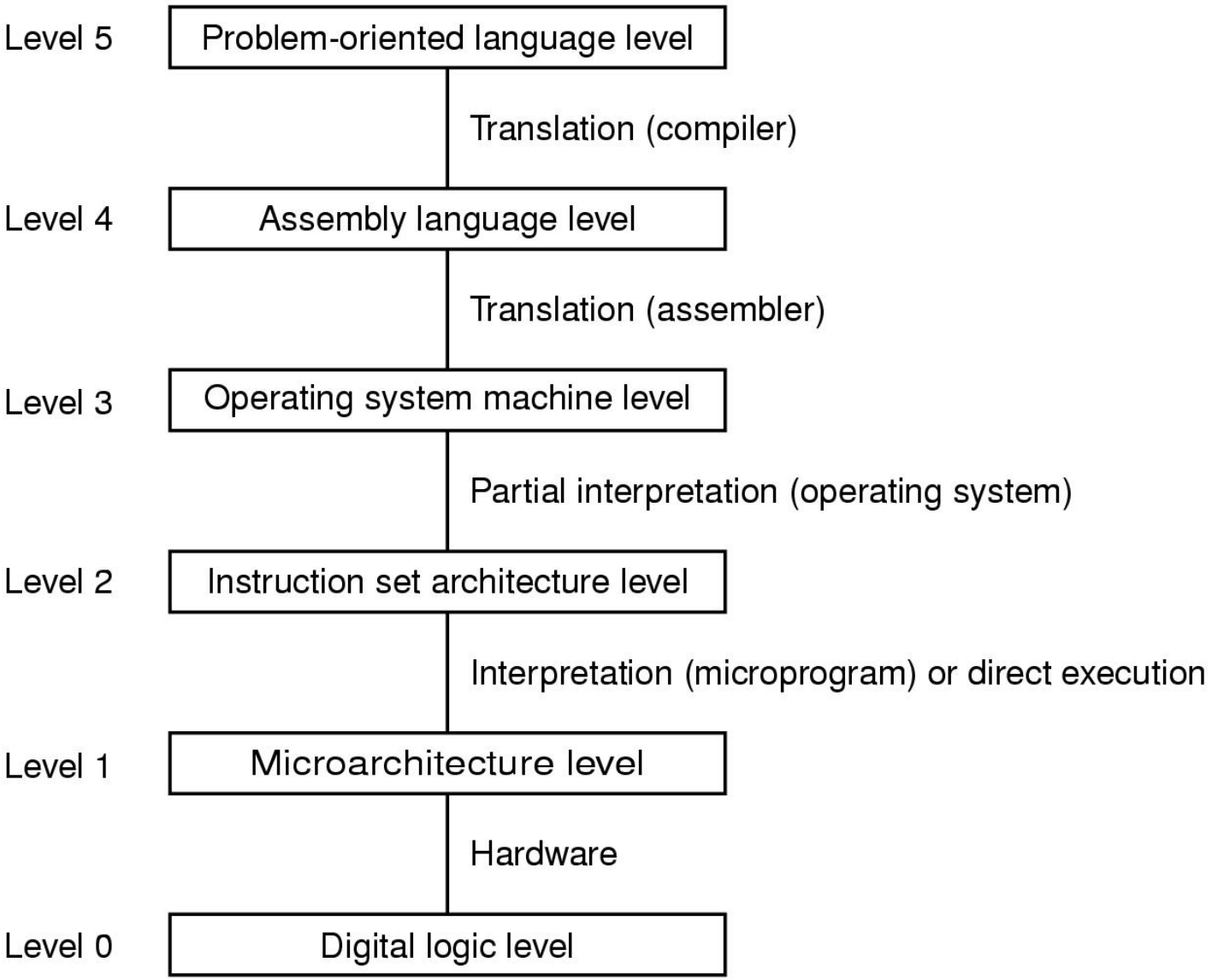
Como máquina extendida

Es el encargado de presentar al usuario el equivalente de un máquina *virtual* o *extendida* que sea más fácil de programar que el hardware subyacente.

Máquina extendida



Máquina extendida (II)



Sistemas Abiertos y Cerrados

Introducción

Los sistemas pueden ser desarrollados de forma tal de que resulte sencilla su integración con otros sistemas (Open), o pueden ser desarrollados con una naturaleza mas propietaria (Closed) construidos para funcionar solo con un sub-grupo de otros sistemas o productos

Definición

Sistema Abierto

Aquel sistema *independiente del fabricante* que cumple con ciertos estándares públicos y generalmente aceptados.

- Promueven la interoperabilidad y compatibilidad entre sistemas y componentes fabricados por distintos fabricantes.
- Pueden ser evaluados de manera independiente.

Sistema Cerrado

Aquel que usa hardware/software propietario que puede o no ser compatible con otros sistemas o componentes.

- El código fuente de los programas generalmente no esta a disposición del público.

Software Libre

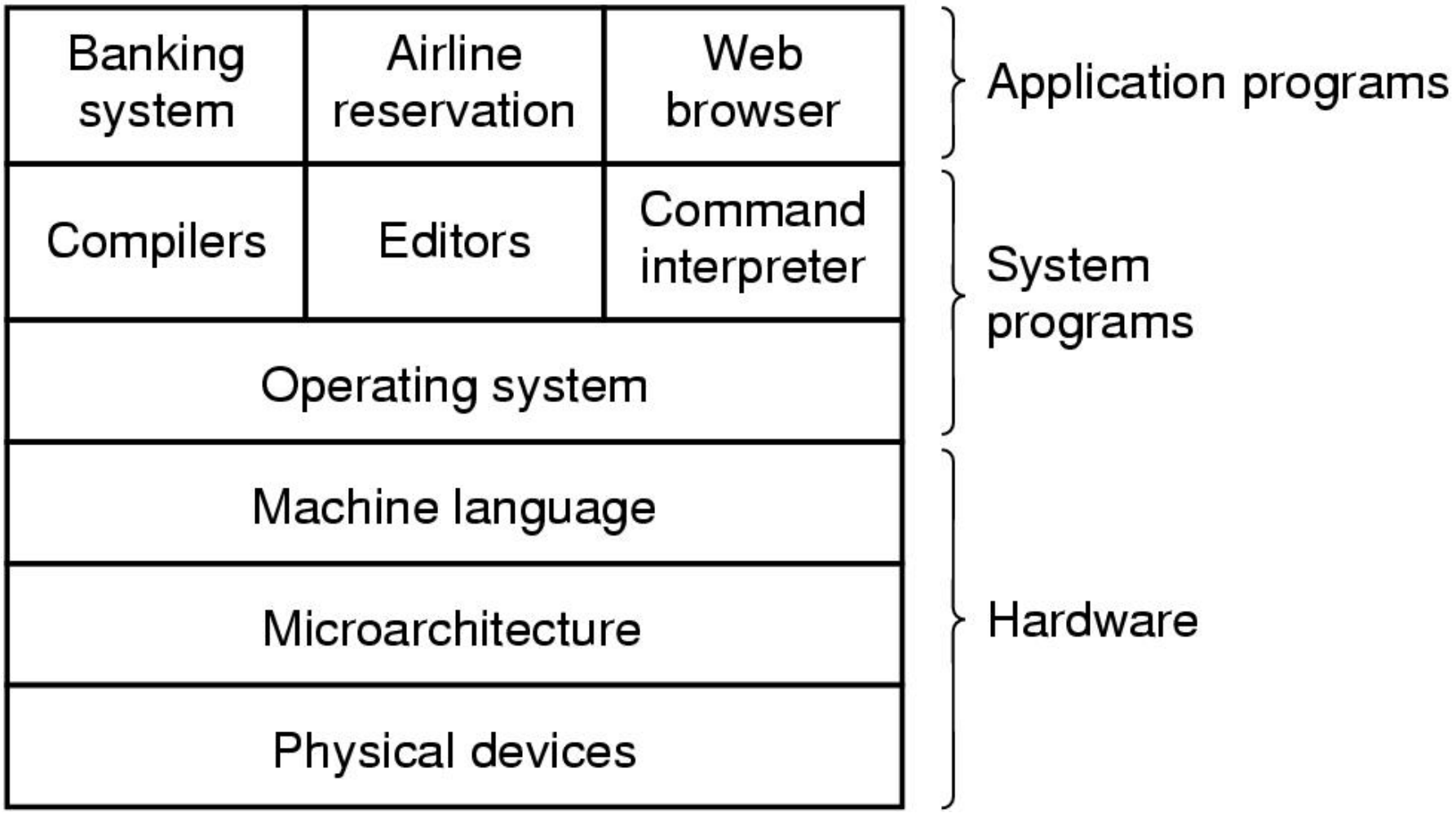
¿Qué es el Software Libre (open source)?

- El "software libre" es una cuestión de libertad, no de precio. Para entender el concepto, debería pensar en "libre" como en "libre expresión", no en "gratis".
- Un programa es software libre si los usuarios tienen las libertades:
  - La libertad de ejecutar el programa, para cualquier propósito (libertad 0).
  - La libertad de estudiar cómo trabaja el programa, y adaptarlo a sus necesidades (libertad 1). El acceso al código fuente es una condición necesaria.
  - La libertad de redistribuir copias para que pueda ayudar al prójimo (libertad 2).
  - La libertad de mejorar el programa y publicar sus mejoras, y versiones modificadas en general, para que se beneficie toda la comunidad (libertad 3). El acceso al código fuente es una condición necesaria.

`http://www.youtube.com/watch?v=jwyLTuVBfeA&feature=related`

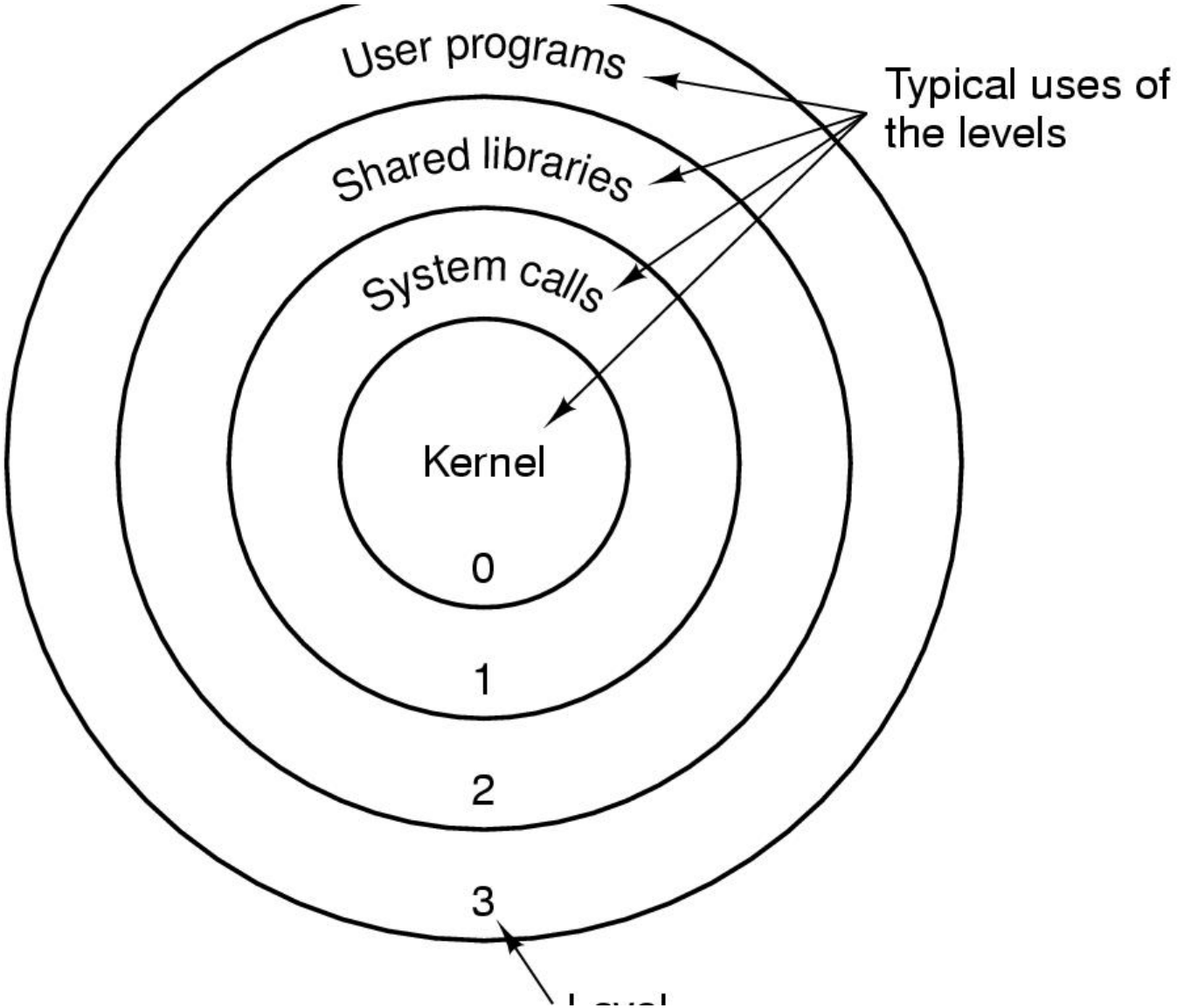
Resumen

Sistema de Cómputo



Modern Operating Systems, 2nd ed., Andrew S. Tanenbaum

El kernel o núcleo de un Sistema Operativo



Otros Conceptos

Proceso: Todo el software ejecutable de la computadora se organiza en varios procesos. *Un proceso es tan solo un programa en ejecución* , consta del programa ejecutable, sus datos y pila. Los estados en los que un proceso se puede encontrar son:

- En ejecución
- Listo (no existe cpu disponible para el)
- Bloquedo

Llamadas al sistema: Mecanismos de comunicación entre el interprete de comandos y el sistema operativo.

Archivo Secuencia de bits.

Fin del tema

- Dudas
- Resumen
  - Software de sistema vs software de aplicación
  - Sistema Operativo
  - Componentes de un sistema de cómputo

El Sistema Operativo Unix/Linux

2. Introducción a Unix/Linux
   - Conceptos básicos
   - Instalación de CentOS
   - Primeros pasos
   - Introducción al sistema de archivos
   - Usuarios, grupos y permisos
   - Introducción al bash shell
   - Entrada y salida estándar
   - Editor de texto vim
   - Utilerías para el procesamiento de texto
   - Utilerías para la búsqueda y procesamiento de archivos

3. Adminstración de GNU/Linux

Conceptos básicos

Objetivos

Al término de este tema será capaz de:

- Identificar las diferencias entre Unix y Linux
- Discutir sobre la historia de Unix y Linux
- Mencionar las principales distribuciones de Linux
- Explicar la filosofía de Linux

¿Qué Unix?

- Unix es un sistema operativo *portable*, *multitarea* y *multiusuario*; desarrollado en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

¿Qué Unix?

- UNIX es una marca registrada de The Open Group en Estados Unidos y otros países. Esta marca sólo se puede aplicar a los sistemas operativos que cumplen la "Single Unix Specification" de esta organización y han pagado las regalías establecidas.

Familias de Unix

- En la práctica, el término UNIX se utiliza en su acepción de familia. Se aplica también a sistemas multiusuario basados en POSIX tales como:
  - GNU/Linux,
  - Mac OS X,
  - FreeBSD, NetBSD, OpenBSD.

- los cuales no buscan la certificación UNIX por resultar cara para productos destinados al consumidor final o que se distribuyen libremente en Internet. En estos casos, el término se suele escribir como UN*X, *NIX, o *N?X.

Historia de Unix (I)

- En 1964 **MULTICS**, un ambicioso proyecto de sistema operativo para cientos de usuarios, fracasa y no llega a terminarse.
- Ken Thomson, desarrollador de MULTICS, con ideas y apoyo de algunos compañeros, escribe un nuevo MULTICS mas modesto en una máquina PDP-7 desechada (1969). Brian Kernighan, compañero de Thomson, lo llama irónicamente UNICS.
- UNIX pasa a una máquina PDP-11 (1970). Ritchie diseño y escribió un *compilador* para lenguaje C.
- Thomson y Ritchie reescriben UNIX en C, rompiendo la tradición de sistemas operativos escritos en lenguaje ensamblador (1973). Esto aumenta la portabilidad del sistema hacia otras maquinas.

Historia de Unix (II)

- Thomson y Ritchie reciben el premio Turing por un memorable artículo sobre UNIX escrito en 1974.[4]
- UNIX es adoptado en las universidades, por tratarse de un *"sistema abierto"* que proporciona todo el código fuente (1974).
- El desmembramiento de AT&T (1984) permite a esta empresa ingresar en el mercado de computadoras, y produce la primera versión comercial de UNIX, el Sistema III, que pronto es sustituido por el Sistema V versiones 2, 3 y 4.
- La *Universidad de Berkeley* produjo una versión mejorada para la PDP-11, llamada **BSD**; luego fueron la 3BSD y luego 4BSD, que incorporo el protocolo de redes TCP/IP.

[4] http://cm.bell-labs.com/who/ken/trust.html

Historia de Unix (III)

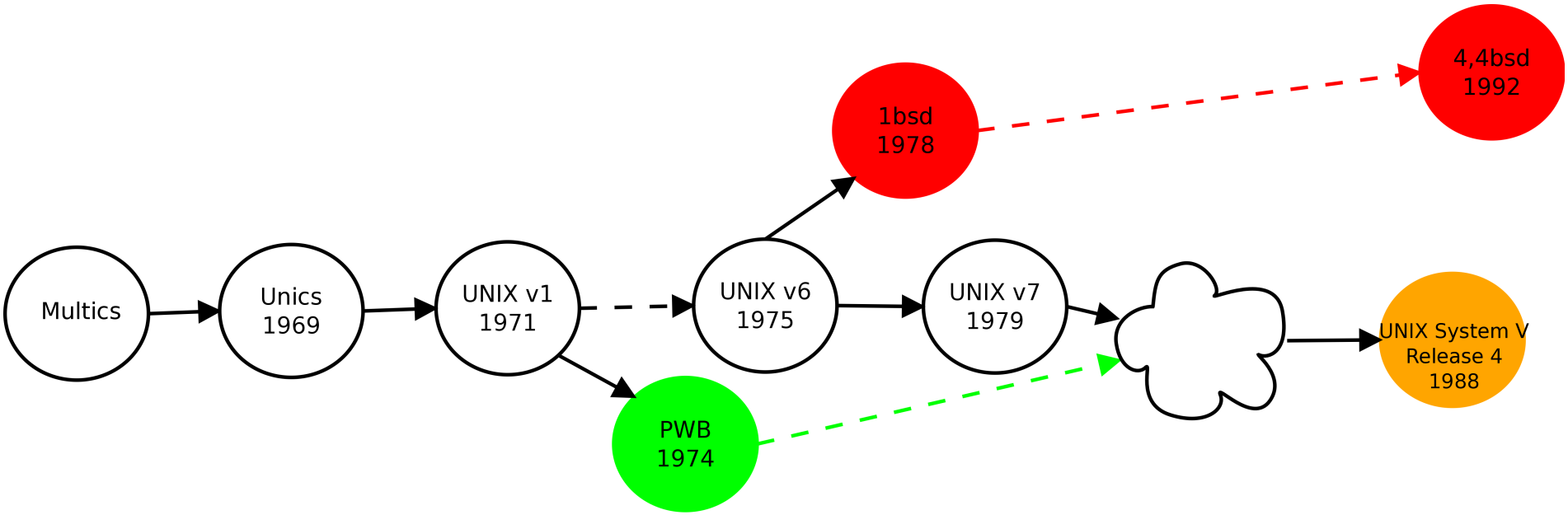
- El grupo **POSIX** estudió y propuso un estándar para UNIX, llamado 1003.1, que define un conjunto de bibliotecas que cualquier sistema UNIX debe proporcionar. Esto resolvió la controversia entre Sistema V y BSD.
- Se forma OSF, Open Software Foundation, integrada por IBM, DEC,HP y otros para enfrentar a AT&T. Crean un UNIX con mas prestaciones que el estándar de POSIX.
- AT&T, para enfrentar a OSF, crea UI, UNIX International, junto con otras empresas es otro UNIX ampliado de POSIX.
- IBM crea su propia variante de UNIX, llamada AIX. La confusión de versiones continua.

Historia de Unix (IV)

- Surgen las versiones UNIX de dominio público como FreeBSD y Linux, que se distribuyen sin costo. Linux es desarrollado por una multitud de personas y equipos de trabajo a través de Internet. FreeBSD es desarrollado por un grupo de trabajo cerrado.

- UNIX hacia el usuario final: distribuciones fáciles de instalar (SuSE, Redhat, Mandriva, Ubuntu), aplicaciones de escritorio (Applixware, OpenOffice).

Historia de Unix (Resumen)



http://www.levenez.com/unix/

Unix Actuales

Producto	Fabricante	Características
Solaris	Sun Microsystems	Basado en AT&T, con muchas extensiones. Arquitectura Sparc y x86.
HP-UX	Hewlett-Packard	Híbrido AT&T y BSD, con particularidades propias. Arquitectura propietaria
Linux	Público	BSD (SunOS) en lo interno, AT&T en la administración. Arquitectura Intel x86, sparc , alpha, y otras. Múltiples distribuciones; RedHat, S.u.s.e., Slackware, Debian, Mandriva. FreeBSD Público Basado en BSD.

¿Qué GNU/Linux?

Definición

GNU/Linux es un clon del sistema operativo Unix, escrito desde cero por el finlandés Linus Torvalds con la asistencia de un pequeño grupo de hackers esparcidos por la red.[a]

[a] http://www.kernel.org/

Estrictamente, Linux se refiere al *núcleo* o kernel. En un sentido más amplio, comprende el núcleo del sistema operativo más un conjunto de programas que permiten compilar lenguajes de programación, editar texto, interpretar comandos, manejar archivos y discos, acceder a otras máquinas, establecer comunicaciones, enviar y recibir correo electrónico, manejar las colas de impresión y un sinfín de tareas más.

Características

- *portable:* el mismo sistema operativo corre en un espectro de máquinas que van desde notebooks a supercomputadoras. Es el único sistema operativo con estas características.
- *flexible:* se adapta a muchas aplicaciones diferentes.
- *potente:* dispone de muchos comandos y servicios ya incorporados.
- *multiusuario:* atiende a muchas personas simultáneamente.
- *multitarea:* hace muchas cosas a la vez.
- *elegante:* sus comandos son breves, coherentes, específicos para cada tarea y muy eficientes.
- *orientado a redes* desde el comienzo.
- Dispone de un estándar (*POSIX*) que debe cumplir todo sistema operativo que pretenda ser Unix, lo que asegura una evolución predecible y compatiblidad con otros Unix.

Historia de Linux (I)

- Lo que en un principio no era más que un proyecto personal de un joven que se creía el mejor programador del mundo[5], terminó siendo uno de los mejores sistemas operativos; usado ampliamente en todo el mundo, desde instituciones educativas hasta comerciales, pasando por gubernamentales.

- Fué en Julio de 1991 cuando Linus aún siendo estudiante de Computer Science en Finlandia, envió su primer mensaje al grupo de noticias `comp.os.minix`, respecto a un proyecto personal sobre el sistema operativo Minix[6].

[5] A sus 21 años

[6] Minix es un clon del sistema operativo Unix distribuido junto con su código fuente y desarrollado por el profesor Andrew S. Tanenbaum en 1987. La última versión oficial de Minix es la 3.0 y data de octubre del 2005.

Historia de Linux (II)

```
From:torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroup: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: 1991 Aug 25, 20578.9541@klaava.Helsinki.FI
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki.


Hello everybody out there using minix- I'm doing a (free) operating system
(just a hobby, won't be big and professional like gnu) for 386(486) AT clones.
This has been brewing since april, and is starting to get ready. I'd like any
feedback on things people like/dislike in minix; as my OS resembles it
somewhat (same physical layout of the file-sytem due to practical reasons)
among other things.


I've currently ported bash (1.08) an gcc (1.40), and things seem to work.
This implies that i'll get something practical within a few months, and I'd
like to know what features most people want. Any suggestions are welcome, but
I won't promise I'll implement them :-)
Linux Torvalds torvalds@kruuna.helsinki.fi
```

Historia de Linux (II)

Versión	Año	Usuarios Estimados	Tamaño del kernel (KBytes)
0.01	1991	100	63
0.99	1992	1000	431
0.99	1993	20,000	938
1.0	1994	100,000	1,017
1.2	1995	500,000	1,850
2.0	1996	1,500,000	4,718
2.2	1999	7,500,000	10,593
2.4	2001	10,000,000	19,789
2.6	2003	20-50,000,000	32,476

Objetivos

GNU/Linux fue diseñado teniendo en mente los siguientes objetivos:

- crear un sistema interactivo de tiempo compartido diseñado por programadores y para programadores, destinado a usuarios calificados.
- que fuera *sencillo*, *elegante*, *escueto* y *consistente*.
- que permitiera resolver problemas complejos combinando un número reducido de comandos básicos.

Filosofía

Los objetivos con que se creó determinaron una "filosofía" caracterizada por:

- comandos cortos, simples, específicos y muy eficientes, que "hacen una sola cosa pero la hacen muy bien".
- entrada y salida estandarizadas que permiten la interconexión de comandos. Esto se llama entubamiento ("pipeling"): la salida de un comando es tomada por el siguiente como entrada.
- todo es un *archivo*.

Distribuciones (I)

- Slackware: Una de las primeras distribuciones Linux (`http://www.slackware.com`), diseñada por Patrick Volkerding a partir de *SLS Linux* . Tuvo una gran aceptación al principio hasta llegar a ser la distribución más popular del mercado. Actualmente ha perdido terreno a favor de distribuciones mas modernas, siendo relegada a aplicaciones especializadas.
- Debian (`http://www.debian.org`) es una distribución bastante popular que no está desarrollada por ninguna compañía comercial sino que es fruto del trabajo de diversos voluntarios en toda la comunidad de Internet.

Distribuciones (II)

- SuSE: Compañía recientemente comprada por Novell, combina el sistema de paquetes de Red Hat (RPM) con una organización derivada de Slackware. Esta distribución es la mas popular en Europa y tiene un gran soporte para diferentes lenguas incluido el Español. Es una de las más fáciles de instalar y configurar, además viene con una gran cantidad de paquetes.
- Mandriva (`http://www.mandriva.com/community/`) antes Mandrake Linux es una distribución Linux que hizo su aparición en julio de 1998 propiedad de Mandriva, enfocada a principiantes o usuarios medios.

Distribuciones (III)

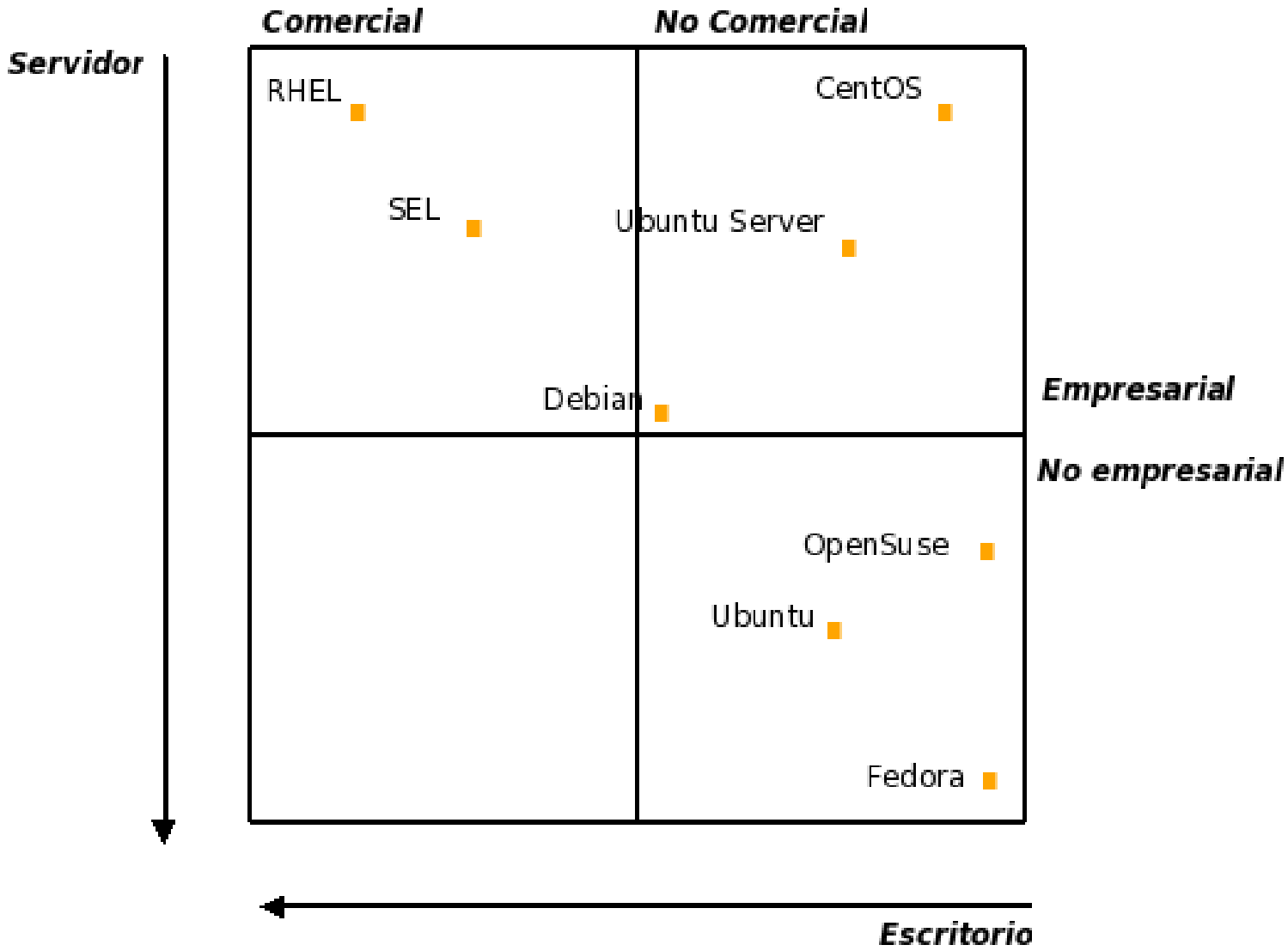
- Red Hat: (`http://www.redhat.com`) es la distribución mas popular del mercado hoy en día, siendo emulada por muchas otras. Muy sencilla de instalar, excelente auto-detección de dispositivos, instalador gráfico y un excelente conjunto de aplicaciones comerciales en su distribución oficial.
- Fedora Core Muchos opinamos que es el laboratorio de pruebas para la versión comercial de RedHat. Esta distro es libre y puede ser descargada del sitio: `http://fedora.redhat.com/`.
  Es ideal para estaciones de trabajo o laptops.

Distribuciones (IV)

- CentOS (`http://www.centos.org`) Acrónimo de Community ENTerprise Operating System es un clon a nivel binario de la distribución Red Hat Enterprise Linux [7]
  - Alternativa libre a distribuciones comerciales de compañías como Red Hat, Suse y Mandriva.
  - Compilado por voluntarios a partir del código fuente liberado por Red Hat, empresa desarrolladora de RHEL

[7] `ftp://ftp.redhat.com/pub/redhat/linux/enterprise/4/en/os/i386/SRPMS`

Distribuciones (V)



Fin del tema

- Dudas
- Resumen
  - Historia de Linux
  - Linus Torvalds y el kernel de Linux
  - Filosofía y Objetivos de Linux

Instalación de CentOS

Objetivos

Al termino de este tema, será capaz de:

- Validar si un equipo es susceptible de ser instalado con GNU/Linux
- Entender los diferentes métodos de instalación
- Crear un esquema de particionamiento personalizado
- Conocer el métodos automatizado de instalación kickstart

Consideraciones de Hardware

- El equipo a instalar **debe** cumplir ciertos requerimientos mínimos.
- Existen listas de compatibilidad de Hardware publicadas en por cada proveedor :
  - Para RHEL/Fedora/CentOS: https://hardware.redhat.com/index.cgi
  - Para OpenSUSE: http://en.opensuse.org/OpenSUSE_HCL
  - Para Ubuntu: https://wiki.ubuntu.com/HardwareSupport
  - Para Linux en general: http://tldp.org/HOWTO/Hardware-HOWTO/

Anaconda, el instalador de CentOS

- Soporta diferentes métodos de instalación:
  - Kickstart para instalaciones automatizadas
  - Actualización de versión
  - Modo de Rescate para recuperar sistemas dañados

- Consta de dos fases o etapas:
  1. Inicio del programa de instalación
  2. Ejecución del programa de instalación.

Primera fase: Inicio del Programa de Instalación

- Los elementos de esta fase son el kernel de instalación y un disco RAM llamado `initrd.img`

- Actividades de la primera fase:
  1. Inicio el instalador
  2. Reconocimiento el hardware
  3. Carga de controladores adicionales
  4. Selección del idioma, configuración del teclado y el método de instalación
  5. Configuración de la red si se requiere para la instalación

Primera Fase

Métodos de inicio (*boot*) soportados:

- `boot.iso` o el DVD/CD de instalación
- Memoria USB con el archivo `bootimg.img`
- Network boot con PXE
- Floppies ya no tienen soporte.

Interacción con el Instalador

- Instalación gráfica
  - Método de instalación por default
  - Parámetros útiles: `lowres`, `resolution`, `skipddc`
- Instalación basada en texto
  - Se inicia usando el parámetro **text**
  - Interfaz basada en menús

Primera Fase: Métodos de Instalación

- CD-ROM
- Disco Duro
- NFS
- FTP
- HTTP

Segunda Fase: Actividades

- Selección del idioma y Configuración del teclado
- Particionamiento el Disco Duro
- Configuración del gestor de arranque
- Configuración de la red y zona horaria
- Selección de grupos de paquetes

Configurando los Sistemas de Archivos (File System)

- Durante el proceso de instalación se eligen los puntos de montaje, tamaño de las particiones y el tipo de sistema de archivo.
  - Puede ser de forma manual o automática

- Existen varias formas de particionar el disco duro :
  - / debe de incluir /etc, /lib, /bin, /sbin
  - El espacio de la memoria de intercambio (swap) es normalmente 2x RAM
  - Puntos de montaje recomendados: **/boot, /home, /usr, /var, /tmp, /usr/local, /opt**

Particionamiento Avanzado

- **Software RAID**
  - Crear una nueva partición y seleccionar la opción Software RAID como tipo de sistema de archivos
  - Combinar las particiones RAID en un dispositivo con la opción RAID

- **LVM**
  - Elegir Physical Volume para crear un volumen físico
  - LVM crea un Volume Group
  - Añadir crea un nuevo Logical Volumes

Selección de paquetes

- Un grupo predeterminado de paquetes se instalan de manera automática
- Seleccionar Personalizar ahora *(Customize now)* para cambiar los grupos de paquetes
- La personalización es necesaria para añadir soporte para idiomas adicionales
- Anaconda de forma automática resuelve las dependencias de paquetes
- Los paquetes pueden ser personalizados después de la instalación con el comando **yum** o **system-config-packages**

Primer Inicio: Configuración Post-Instalación

- De ser necesario se configura el sistema X Window
- Configuración del Firewall y SELinux
- Configuración del módulo Kdump
- Ajuste de fecha y hora
- Creación de usuarios
- Configuración de la tarjeta de sonido
- Instalar RPMs adicionales

Kickstart

- Método de instalación vía scripts
- Soporta todas las opciones de Anaconda
- El archivo `/root/anaconda-ks.cfg` es generado de manera automática durante la instalación
- Herramienta de configuración: **system-config-kickstart**
- Para revisar sintaxis: **ksvalidator**
- Para iniciar en modo Kickstart usar el parámetro *ks* al inicio de la instalación

Fin del tema

- Dudas
- Resumen
  - Pasos para llevar a cabo la instalación
  - Opciones de Anaconda

Primeros Pasos

Objetivos

Al termino de este tema, será capaz de:

- Entrar a sesión en CentOS
- Iniciar el servidor X desde la consola
- Acceder a una terminal de linea de comandos desde el servidor X
- Cambiar su contraseña
- Entender los permisos/privilegios de root
- Conocer sus permisos
- Editar archivos de texto
- Ejecutar comandos desde el prompt
- Explicar el propósito y uso de algunos comandos
- Usar la ayuda

Inicio de sesión (logging)

- Dos tipos de ventanas de inicio: consolas virtuales (modo texto) y consolas gráficas
- Para iniciar sesión es necesario de un nombre de usuario y contraseña
- Cada usuario tiene su propio directorio llamado hogar (home)
- Un sistema Linux típico tiene seis consolas virtuales y una consola gráfica
  - Los servidores usualmente solo cuentan con consolas virtuales
  - Las computadoras personales (desktops) y las estaciones de trabajo (workstation) normalmente cuentas con las dos
- Para cambiarse entre consolas se usa la combinación de teclas **Ctrl-Alt-F[1-6]**
- Para acceder a la consola gráfica se teclea **Ctrl-Alt-F7**

Componentes del Sistema X Windows

- El sistema X Windows fue desarrollado a mediados de los años 1980 en el MIT para dotar de una interfaz gráfica a los sistemas Unix
- Xorg es la versión usada como sistema X Windows por CentOS
  - Implementación de software libre de X
- La apariencia y el comportamiento son controlados por el ambiente de escritorio
- CentOS provee dos ambientes de escritorio:
  - GNOME: ambiente de escritorio predeterminado
  - KDE: alternativa para ambiente de escritorio

Iniciando el servidor X

- En algunos sistemas, el servidor X inicia de manera automática al iniciar el sistema operativo (boot time)
- En caso de que el sistema solo inicie consolas virtuales, es necesario iniciar el servidor X de manera manual
  - El servidor X debe estar preconfigurado por el administrador del sistema
  - Iniciar una sesión en la consola virtual y ejecutar el comando **startx**
  - Para cambiarnos a la sesión gráfica tecleamos **Ctrl-Alt-F7**

Cambiando la contraseña

- Las contraseña controlan el acceso al sistema
  - Es conveniente que la primera vez que se firma[8] al sistema se cambie la contraseña.
  - Cambiar la contraseña con frecuencia
  - Elegir una contraseña robusta
- Para cambiar la contraseña desde una terminal usamos el comando **passwd**

[8] Acción que comprende dar nombre de usuario y contraseña, y que en adelante llamaremos entrar en sesión.

El usuario *root*

- El usuario *root* es el administrador en los sistema Unix
  - Es también conocido como *superuser*
  - root tiene el control total del sistema y por consecuencia casi ilimitada capacidad para dañarlo

- No entrar al sistema como *root* a menos que sea estrictamente necesario
  - Una cuenta de usuario normal tiene una capacidad de daño limitada

Cambiando de identidad

- **su** - crea un nuevo shell como root
- **sudo** *comando* ejecuta el *comando* como root
  - Requiere que el administrador configure previamente la herramienta
- **id** muestra información del usuario actual

Editando archivos de texto

- El editor **nano**
  - Fácil de aprender, fácil de usar
  - No presente en los Unix estándar
- Otros editores
  - **gedit**, un simple editor gráfico
  - **vim**, avanzado y completo editor
  - **vi**, avanzado, completo y difícil de usar, pero disponible en casi todos los *NIX

Ejecutando Comandos

- Todos los comandos en *NIX tiene la siguiente sintaxis:
  - **comando** *opciones parámetros*
- Cada elemento es separado por un espacio
- Las *opciones* modifican el comportamiento del comando
  - Las opciones de una sola letra se preceden por -
    - Pueden ser indicadas como -**a** -**b** -**c** o -**abc**
  - Las opciones que son palabras son precedidas por –
    - Ejemplo: –**help**
- Los *argumentos* son nombres de archivos o cualquier otra información que necesite el comando
- Si se van a ejecutar mas de un comando por línea, es necesario separarlos con **;**

Algunos Comandos Simples

- **date** - muestra la fecha y hora
- **cal** - muestra un calendario
- **who** - muestra los nombres de usuarios conectados al sistema en este momento.
- **hostname** - muestra el nombre de la máquina *NIX.

Obteniendo Ayuda

- Es imposible memorizarse todo (hay otras cosas en que pensar)
- Varios niveles de ayuda
  - **whatis**
  - **comando** *–help*
  - **man** o **info**
  - `/usr/share/doc`
  - Documentación de CentOS

El comando whatis

- Muestra una breve descripción de los comandos
- Usa una base de datos que se actualiza de manera nocturna
- Normalmente no esta disponible inmediatamente después de una instalación

Example

```
$ whatis cal
cal (1) - displays a calendar
cal (1p) - print a calendar
```

Opción –help -h

- Muestra un resumen de como usar el comando y la lista de argumentos que puede recibir
- Usado por casi todos los comandos

Example

```
$ tar --help Usage: tar [OPTION]... [FILE]...
Examples:
tar -cf archive.tar foo bar # Create archive.tar from files foo and bar.
tar -tvf archive.tar # List all files in archive.tar verbosely.
tar -xf archive.tar # Extract all files from archive.tar.
```

El comando man

- Provee de información detallada sobre los comandos
- Casi todos los comandos tiene su página de **man**
- Las páginas están agrupadas en *capítulos*
- Sintaxis:
  - **man [<capítulo>] <comando>**

Leyendo las páginas del man

- Mientras se esta viendo una página del man
  - Para desplazarse por el documento se usan las "flechitas",**PgUp**,**PgDn**
  - */texto* busca por el texto
  - **n/N** se desplaza a la siguiente/anterior ocurrencia del texto
  - **q** para salir

- Buscando el Manual
  - **man -k** *keyword* lista las páginas que correspondan al *keyword*
  - Usar la base de datos **whatis**

El comando info

- Similar a **man**, pero con mayor detalle
- Ejecutar **info** sin argumentos para listar todas las páginas
- Las páginas de **info** están estructuradas como un sitio web
  - Cada página esta dividida en "nodos"
  - Cada nodo esta precedido por un *
  - **info [comando]**

Leyendo las páginas de info

- Mientras se esta viendo una página info
  - Para desplazarse por el documento se usan las "flechitas",**PgUp**,**PgDn**
  - **Tab** para moverse al siguiente link
  - **Enter** para seguir el link seleccionado
  - **n/p/u** para ir al siguiente/anterior/un nivel arriba del nodo
  - **s** *text* busca el texto (default: última búsqueda)
  - **q** para salir

Documentación Extendida

- El directorio `/usr/share/doc`
  - Contiene un subdirectorio por paquete instalado
  - Aquí se encuentra la documentación que no entra en otro lugar
    - Ejemplos de archivos de configuración
    - Documentación en formato HTML/PDF/PS
    - Detalles del licenciamiento

Salir del Sistema

- **exit** termina la sesión
- Las teclas **Ctrl+D** también terminan la sesión.

Fin del tema

- Dudas
- Resumen
  - Nombre de usuario y contraseña
  - Ejecutar comandos
  - **startx**
  - **gnome-terminal**
  - **passwd**
  - **su**
  - **nano**
  - **vi**
  - Obtener ayuda

Introducción al Sistema de Archivos

Objetivos

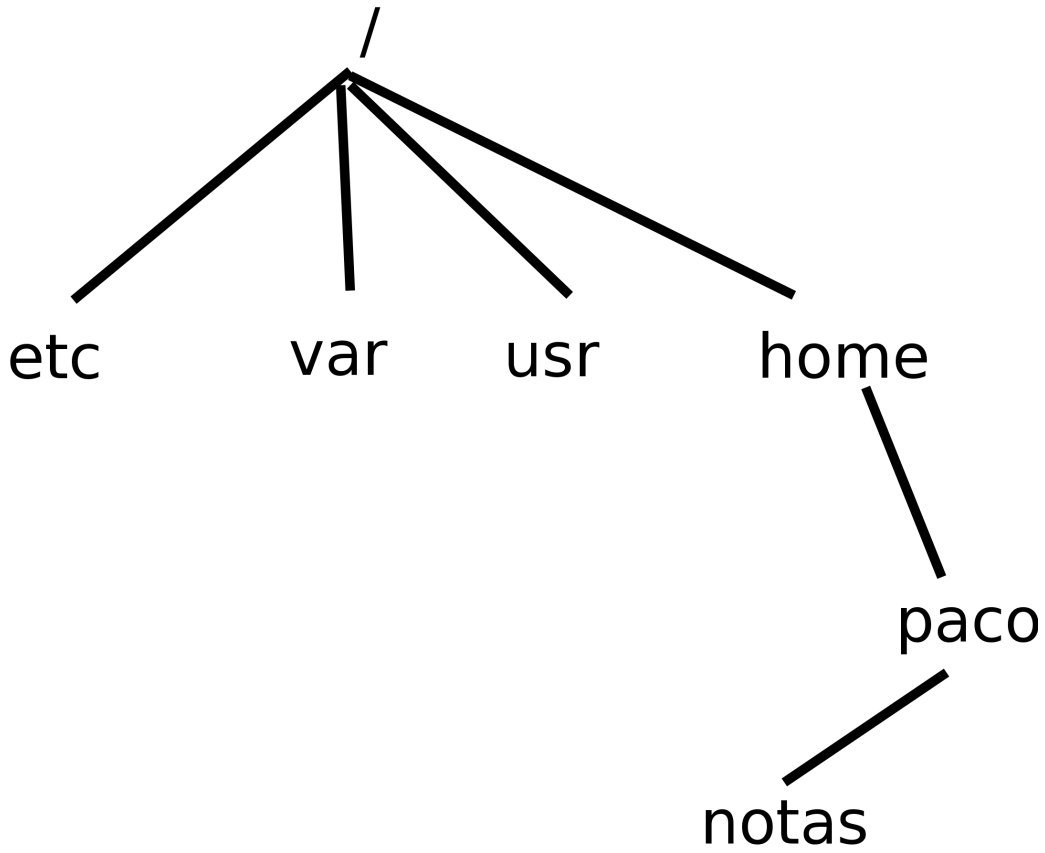
Al termino de este tema, será capaz de:

- Describir los elementos más importante de la jerarquía del sistema de archivos
- Copiar, mover y borrar archivos
- Crear y ver archivos

Jerarquía del Sistema de Archivos

- La estructura del sistema de archivos es jerárquica, es decir una gráfica dirigida o, vista de otro modo, una estructura arbórea.
- Nombre sensibles a mayúsculas y minúsculas (case-sensitive)
- El directorio principal, llamado raíz, representado por el caracter / que a su vez es utilizado para separar los nombres de los subsiguientes directorios.
- El estándar se puede consultar en http://www.pathname.com/fhs/

Sistema de Archivos



Algunos Directorios Importantes

- **Directorios hogar**: `/root`, `/home/username`
- **Ejecutables del usuario**: `/bin`, `/usr/bin`, `/usr/local/bin`
- **Ejecutables del sistema**: `/sbin`, `/usr/sbin`, `/usr/local/sbin`
- **Puntos de montaje**: `/media`, `/mnt`
- **Configuración**: `/etc`
- **Archivos temporales**: `/tmp`
- **Kernel y Gestor de Arranque**: `/boot`
- **Datos del servidor**: `/var`, `/srv`
- **Información del sistema**: `/proc`, `/sys`
- **Bibliotecas compartidas**: `/lib`, `/usr/lib`, `/usr/local/lib`

Directorio Actual de Trabajo

- Cada shell y proceso del sistema tiene un *directorio actual de trabajo* (cwd por sus siglas en inglés)

- **pwd**
  - Imprime la ruta absoluta del cwd del shell

Nombre de Archivos y Directorios

- Los nombres puedes ser de hasta 255 caracteres
- Todos los caracteres son válidos, con excepción de la /
  - No es recomendable usar caracteres especiales en nombre de directorios o archivos
  - Algunos caracteres deben protegerse con comillas para poderlos referenciar
- Los nombres son sensibles a mayúsculas y minúsculas
  - Ejemplo: MAIL, Mail, mail y mAil
  - De nuevo, es posible pero no recomendable

Rutas absolutas y relativas

- Rutas Absolutas
  - Inician con una /
  - Ruta completa a la ubicación del archivo
  - Puede ser usado en cualquier momento para indicar el nombre de un archivo
- Rutas Relativas
  - No inician con una /
  - Indican la ruta desde el directorio actual de trabajo
  - Pueden usarse como una manera rápida para indicar el nombre de un archivo

Cambiando de Directorio

- **cd** nos cambia de directorio
  - a una ruta absoluta o relativa
    - **cd** /home/paco/cursos
    - **cd** coapa/Modulo5
  - A un directorio superior
    - **cd ..**
  - Al directorio hogar
    - **cd**
  - Al directorio previo de trabajo
    - **cd -**

Visualizar el contenido de un directorio

- El comando utilizado para visualizar el contenido de un directorio es **ls**
- Uso:
  - **ls [opciones]** *[archivos_o_directorios]*
- Ejemplos:
  - **ls -a** (lista archivos ocultos)
  - **ls -l** (despliega información extendida)
  - **ls -R** (lista recursiva)
  - **ls -ld** (información de directorios y ligas suaves)

Copiando Archivos y Directorios

- **cp** copia archivos y directorios
- Uso:
  - **cp [opciones]** *origen destino*
- Es posible copiar más de un archivo a la vez si el destino es un directorio:
  - **cp [opciones]** *archivo1 archivo2 destino*
- Algunas consideraciones:
  - Si el destino es un directorio, los archivos se copian en el directorio
  - Si el destino es un archivo,la copia sobreescribe el destino
  - Si el destino no existe, la copia es renombrada

Moviendo y Renombrando Archivos y Directorios

- **mv** mueve y/o renombra archivos y directorios
- Uso:
  - **mv [opciones]** *origen destino*
- Uno más de un archivo puede ser movido al mismo tiempo si el destino es un directorio
  - **mv [opciones]** *archivo1 archivo2 dest*
- El destino trabaja igual que **cp**

Creando y Borrando Archivos

- **touch** - Crea archivos vacíos o actualiza las marcas de tiempo (timestamps)
- **rm** - borra archivos
- Uso:
  - **rm [opciones]** *<archivo>...*
- Ejemplo:
  - **rm -i** *archivo* (interactivo)
  - **rm -r** *directorio* (recursivo)
  - **rm -f** *archivo* (fuerza)

Creando y Borrando Directorios

- **mkdir** - Crea directorios
- **rmdir** - Borra directorios
- **rm -r** - Borra estructura de directorios

Determinando el tipo de archivo

- Los archivos contienen diferentes tipos de información
- Si es necesario validar el tipo de datos que contiene un archivo antes de abrirlo usamos:
- **file [opciones] <archivos>...**

Fin del tema

- Dudas
- Resumen
  - Jerarquía del sistema de archivos
  - Comandos para la administración de archivos

Usuarios, Grupos y Permisos

Objetivos

Al termino de este tema, será capaz de:

- Explicar el modelo de seguridad de Linux
- Explicar el propósito de las cuentas de usuario y los grupos
- Leer y asignar permisos

Usuarios

- A cada usuario se le asigna un único identificador conocido como **User ID** (UID)
  - root posee el UID 0
- Los nombre de usuario y UIDs son almacenados en el archivo `/etc/passwd`
- A cada usuario se le asigna un directorio hogar y un programa que se ejecuta cuando entra al sistema (comúnmente un shell)
- Los usuarios no pueden leer, escribir o ejecutar archivos de otros usuario sin su previo permiso.

Grupos

- Los usuarios son asignados a grupos
- A cada grupo se le asigna un único identificador conocido como **Group ID** (gid)
- Los grupos son almacenados en el archivo `/etc/group`
- Todos los usuarios deben de pertenecer a por lo menos un grupo
  - Posteriormente se pueden agregar a grupos adicionales para incrementar su nivel de acceso
- Todos los usuarios en un grupo pueden compartir archivos a los miembros de su grupo

Tipos de Permisos

- Se usan cuatro símbolos para indicar permisos sobre archivos o directorios:
  - `r`: permiso para leer un archivo o listar el contenido de un directorio
  - `w`: permiso para escribir o modificar un archivo o crear y borrar archivos en un directorio
  - `x`: permiso para ejecutar un programa o cambiarse a un directorio
  - `-`: ausencia de permiso (en lugar de `r`,`w` o `x`)

Analizando Permisos (I)

- Los permisos se pueden ver con el comando **ls -l**

Example

```
$ ls -l /bin/bash
-rwxr-xr-x 1 root wheel 1068844 Apr 10 2007 /bin/bash
```

- El tipo de archivo y los permisos son representados por 10 caracteres

Analizando Permisos (II)

```
-rwxr-x--- 1 paco unix 12 Feb 8 11:25 script
```

- Lectura, escritura y ejecución para el dueño del archivo, `paco`
- Lectura y ejecución para los miembros del grupo `unix`
- Ningún permisos para el resto del mundo

Cambiando Permisos - Método Simbólico

- Para cambiar los permisos de acceso usamos:
  - **chmod [-R]** *modo archivo*
- Donde `modo` es:
  - **u, g** u **o** para el usuario, grupo y otros
  - **+** o **-** para añadir o quitar permisos
  - **r, w** o **x** para lectura, escritura y ejecución
- Ejemplos:
  - **ugo+r** Permiso de lectura para todos
  - **o-wx** Quita los permisos de escritura y ejecución a otros

Cambiando Permisos - Método Numérico

- Usa tres números para el modo
  - el primer número especifica los permisos de dueño
  - el segundo número especifica los permisos del grupo
  - el tercer número representa los permisos del resto del mundo (otros)
- Los permisos son calculados añadiendo
  - 4 (para lectura)
  - 2 (para escritura o modificación)
  - 1 (para ejecución)
- Ejemplos:
  - **chmod 640 miarchivo**

Fin del tema

- Dudas
- Resumen
  - Todos los archivos tienen un solo propietario y pertenecen a un grupo
  - Los permisos de un archivo están agrupados en dueño, grupo y el resto del mundo
  - Se pueden otorgar tres permisos: lectura, escritura y ejecución

Introducción al bash shell

Objetivos

Al termino de este tema, será capaz de:

- Usar atajos en la línea de comandos
- Usar las expansiones a la línea de comandos
- Usar la historia de comandos
- Usar la terminal de gnome (**gnome−terminal**)
- Saber como usar las variables locales y de ambiente
- Crear aliases
- Entender como el shell analiza una línea de comandos
- Configurar los archivos de inicio del shell

Historia

- El bourne shell se convirtió en estándar en Unix desde 1979.
  - Se encuentra en la ruta `/bin/sh`
  - Disponible aún en la actualidad
- Berkeley C shell (csh) era más amigable al proveer características adicionales como el histórico de comandos,
  - Durante mucho tiempo la práctica común era trabajar en csh y programar en sh
- David Korn incluyo en el Bourne shell el histórico de comandos, control de tarea y capacidad adicionales de programación.
  - Eventualmente ksh se convirtió en estandar al mezclar lo mejor del sh con el csh
- La fundación de software libre desarrolló un clon del sh, nombrándolo bash (Bourne-Again SHell)
  - Hoy en día es el estandar.

Metacaracteres (Globbing)

- Caracter que tiene un significado especial para el interprete de comandos (shell)
  - ***** - Se sustituye por 0 o más caracteres
  - **?** - Se sustituye por un caracter
  - **[0-9]** - Se sustituye por un rango de números
  - **[abc]** - Se sustituye por una letra de la lista item `[^abc]` - Se sustituye por cualquier letra menos alguna de la lista

Tabulador

- La tecla **Tabulador** (Tab) permite completar las líneas de comando
  - En un comando, completa el nombre del comando
  - En un parámetro. completa el nombre del archivo

- Ejemplos:
  - `$ ali<tab>`
  - `$ alias`
  - `$ ls htt<tab>`
  - `$ ls httpd.conf`

Histórico

- **bash** almacena todos los comandos que se teclean, formado una *historia* de comandos
- El comando **history** se usa listar los comandos almacenados

Example

```
$ history
612 cd /tmp
613 ls -la
614 cd
615 cp /etc/passwd .
616 vi passwd
617 history
```

Trucos del histórico

- Usar las flechas de navegación **arriba** y **abajo** para navegar entre los comandos tecleados
- **Ctrl**−**r** para buscar un comando en el historial
- Para utilizar el último argumento del comando anterior:
  - **Esc,.** (Presionar la tecla escape y luego un punto)
  - **Alt**− (Presionar alt más punto)

Ampliación de la línea de comandos

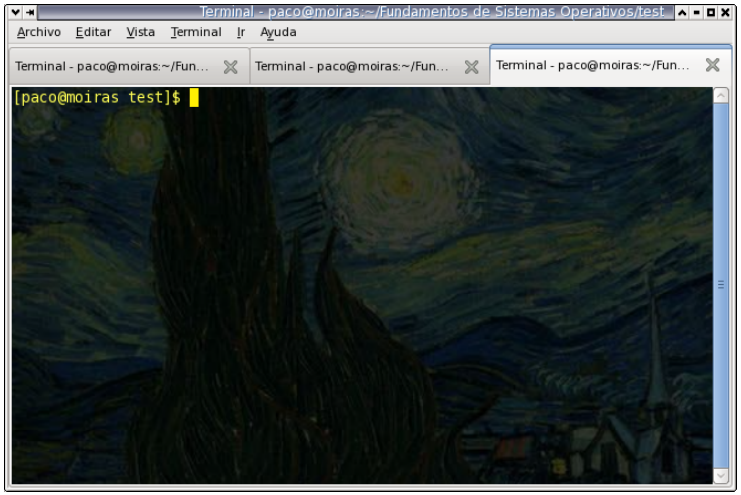
- Tilde (~)
  - Puede indicar el directorio hogar del usuario
    `$ cat ~/.bash_profile`
  - Puede indicar el directorio hogar de otro usuario
    `$ ls ~maria/public_html`
- $( ) o acento grave ` `
  - Ejecuta un comando dentro de otro.
    echo "El 'nombre' de este equipo es $(hostname)"
    `El 'nombre' de este equipo es moiras`
- { }
  - Usados para repetir caracteres.
    `$ touch archivo{1,2,3}`
    `$ rm -f archivo{1,2,3}`

Editando la línea de comandos

- **Ctrl-a** desplaza el cursos al principio de la línea
- **Ctrl-e** desplaza el cursos al final de la línea
- **Ctrl-u** borra hasta el inicio de línea
- **Ctrl-k** borra hasta el final de línea
- **Ctrl-flecha** desplaza el cursos izquierda o derecha palabra por palabra

gnome-terminal

- Aplicaciones -> Accesorios -> Terminal
- Emulador gráfico de terminal que soporta múltiples sesiones en forma de pestañas.
  - **Ctrl-Shift-t** crea una nueva pestaña
  - **Ctrl-PgUp/PgDn** cambia entre pestañas
  - **Ctrl-Shift-c** copia el texto seleccionado
  - **Ctrl-Shift-v** pega el texto



Introducción a la programación en bash

- Shell scripts son archivos de texto que contiene una serie de comandos o instrucciones que serán ejecutadas.
- Usos:
  - Automatizar comandos usados frecuentemente
  - Crear aplicaciones simples
  - Manipular cadenas de texto o archivos
  - Elaborar rutinas de diagnóstico

Creación de shell scripts

- Paso 1: Creación de un archivo de texto con algún editor como **vi**.
  - La primera línea indica el número mágico (magic shebang[9])
    #!/bin/bash
- No olvidar comentar los scripts
  - Los comentarios inician con #
- Paso 2: Convertir el script en ejecutable

  `$ chmod u+x miscript.sh`
- Para ejecutar el nuevo script:
  - Mover el script a un directorio en dentro del path de ejecución o
  - Indicar la ruta absoluta o relativa al script en la línea de comandos

[9] http://www.in-ulm.de/~mascheck/various/shebang/

Ejemplo de Shell Script

```
#!/bin/bash
# Muestra la hora y el directorio actual de trabajo
echo ‘‘Saludos. La fecha $(date)’’
echo ‘‘Tu directorio actual de trabajo es: $(pwd)’’
```

Variables

- Una variable es un símbolo que representa un elemento no especificado de un conjunto dado.
  - Usadas para almacenar datos o resultados de la ejecución de un comando.
- Se asignan **VARIABLE=VALOR**
- Se invocan con **$VARIABLE**

  $ **HOLA="Hola, bienvenido a $(hostname)"**

  $ **echo $HOLA**

Variables de Ambiente

- Las variables son *locales* a un shell por default.
- Las variables de ambiente se heredan a los procesos que el shell genere.
  - **export VARIABLE=VALOR**
  - Son usadas por algunos programas para configuración.

Variables más comunes

- Variables de configuración:
  - `PS1`: Modifica la presentación del **prompt** del bash
  - `PATH`: Indica los directorios donde se buscan los ejecutables
  - `EDITOR`: Editor por default
  - `HISTFILESIZE`: Número de comandos almacenados en el **histórico** del bash
- Variables informativas
  - `HOME`: Directorio hogar del usuario
  - `EUID`: User ID del usuario

Aliases

- Los alias permiten crear atajos (shortcuts) a los comandos
  $ **alias dir='ls -laF'**
- **alias** sin parámetros muestra los alias creados
- **alias** mas el nombre del alias se muestra el contenido
  $ **alias dir**
  $ **alias dir='ls -laF'**

Metacaracteres

- Backslash (\) hace el siguiente caracter literal

  $ **echo Precio: \$19.75**

  `Precio $19.75`

- Los caracteres usados para proteger los metacaracteres:
  - Comilla simple (') protege todo.
  - Comilla doble (") protege todo menos
    - $ (símbolo monetario) - usado en variables
    - ‘ (comilla invertida) - usado en sustitución de comandos
    - \ (diagonal invertida) - protege un solo metacaracter
    - ! (admiración) - usado en el comando **history**

Login vs non-login shells

- El inicio de sesión aplica de manera diferente para los shells creados desde el login y para los non-login
- Login shells son aquellos:
  - Creados desde el inicio de sesión (incluye el ambiente X)
  - su -
- Non-login shells son:
  - su
  - scripts
  - instancias del bash

Tareas de inicio del bash: profile

- Almacenadas en `/etc/profile` (globales) y ~/.bash_profile (usuario)
- Solo se ejecutan en login shells
- Uso:
  - Configurar variables de ambiente
  - Ejecutar comandos (ejemplo: revisar si hay correos nuevos)

Tareas de inicio del bash: bashrc

- Almacenadas en `/etc/bashrc` (globales) y `~/.bashrc` (usuario)
- Ejecutadas para todos los shells
- Uso:
  - Configurar variables de ambiente
  - Definir alias

Tareas al terminar la sesión

- Almacenadas en ~/.bash_logout (usuario)
- Se ejecutan cuando el shell termina su sesión
- Uso:
  - Creación de respaldos automáticos
  - Borrar archivos temporales

Scripting: Leyendo parámetros por posición

- Los parámetros por posicionales se almacenan en variables que almacenan los argumentos enviados a través de la linea de comandos desde un script
- Los parámetros por posicionales disponibles son $1, $2, $3, etc.
- $* que almacena todos los argumentos
- $# que almacena el número de argumentos

Scripting: Leyendo parámetros interactivos

- El comando **read** se usa para leer argumentos de forma interactiva
  - **-p** sirve para indicar un prompt
  - **read** lee de la STDIN y asigna una palabra por cada variable
    $ **read -p "Nombre de archivo: " ARCHIVO**

Fin del tema

- Dudas
- Resumen
  - Expansión: **$()**
  - Historial: **!cadena, !número**
  - Escapar caracteres: ' ',
  - Variables locales y globales
  - configuración el bash shell
  - parámetros posicionales y uso del comando **read**

Entrada y Salida Estándar

Objetivos

Al termino de este tema, será capaz de:

- Redirigir la salida de un comando a un archivo
- Interconectar comandos
- Usar **for** para iterar sobre valores

Entrada y Salida Estándar

- Linux provee tres canales de I/O a los programas:
  - *Entrada estándar* (STDIN) - por default el teclado
  - *Salida estándar* (STDOUT) - por default la terminal
  - *Error estándar* (STDERR) - por default la terminal
- STDOUT y STDERR pueden ser redirigidos a un archivo: `comando operador archivo`
- Operadores soportados:
  - $>$ Redirige STDOUT a un archivo
  - **2**$>$ Redirige STDERR a un archivo
  - **&**$>$ Redirige toda la salida del comando a un archivo
- El contenido del archivo es sobreescrito por default. Usar $>>$ para concatenar.

Entrada y Salida Estándar - Ejemplos

- Ejecutar este comando como un usuario diferente a root genera varios errores:

```
$ find /etc -name passwd
```

- Los operadores pueden ser usados para redirigir los errores:

```
$ find /etc -name passwd > find.out
$ find /etc -name passwd 2> /dev/null
$ find /etc -name passwd > find.out 2>find.err
```

Redirigiendo la STDOUT a un programa (Entubamiento)

- El entubamiento (con el caracter |) permite interconectar comandos: `comando1 | comando2`
  - La STDOUT del comando1 es enviado al STDIN del comand2 en lugar de la pantalla.
  - STDERR no es enviado en el entubamiento
- Usado para combinar la funcionalidad de varios comandos `comando1 | comando2 | comando3 ... etc`

Entubamiento - Ejemplos

- **less**: Muestra la salida del comando ls de manera paginada:
  `$ ls -l /etc | less`
  - Se puede buscar cadenas con /
- **mail**: Envía la entrada vía correo electrónico
  `$ echo ‘‘correo de prueba’’ | mail -s ‘‘prueba’’ usuario@ejemplo.com`
- **lpr**: Envía la entrada a una impresora
  `$ echo ‘‘prueba de impresión’’ | lpr`

Redirigiendo a múltiples objetivos

- `$ comando1 | tee archivo | comando2`
- Almacena STDOUT del *comando1* en *archivo* y entuba la salida al *comando2*
- Usos:
  - Localización y resolución de problemas en entubamientos complejos
  - Ver y registrar la salida de un comando de manera simultanea

Redirigiendo STDIN desde un archivo

- Para redirigir la entrada estándar se usa el caracter $<$
- Algunos comandos pueden aceptar redirigir la STDIN desde un archivo:

```
$ tr 'A-Z' 'a-z' < .bash_profile
```

  - Este comando cambia las mayúsculas por minúsculas del archivo `.bash_profile`

Equivalente a:

```
$ cat .bash_profile | tr 'A-Z' 'a-z'
```

Enviando múltiples líneas a la STDIN

- Para redirigir varias líneas desde el teclado a la STDIN se usa <<**PALABRA**
  - todo el texto hasta la **PALABRA** es enviado a la STDIN

```
$ mail -s ‘‘Comunicarse con el sysadmin’’ paco@ejemplo.com << END
> Hola paco,
>
> Favor de comunicarse al departamento de administración de servidores.
> Necesitamos agendar un mantenimiento para el servidor1
>
> Saludos cordiales
> --
> Sysadmin Team
> END
```

Scripting: for

- Realizar una acción por cada valor de un grupo
- Ejemplo:

```
for NOM in hugo paco luis
do
CORREO=‘‘$NOM@ejemplo.com’’
MENSAJE=’Enviar estatus de proyectos hoy!’
echo $MENSAJE | mail -s Recordatorio $CORREO
done
```

- También es posible usar una secuencia de números como lista:
  - **for num in $(seq 1 10)**
    - Asigna los número del 1 al 10 a la variable $num
    - **seq X Y** imprime una lista de números de X hasta Y
  - **for archivo in *.txt**
    - Asigna los nombre de archivos a la variable $archivo

Fin del tema

- Dudas
- Resumen
  - Entrada y Salida Estándar
  - Redirección de archivos
    - Entrada estándar (<)
    - Salida estándar (>)
    - Error estándar (**2**>)
  - Entubamiento de comandos
  - Breve descripción del comando **for**

Editor de texto vim

Objetivos

Al termino de este tema, será capaz de:

- Usar los tres modos de operación de **vi** y **vim**
- Desplazarse entre el texto y entrar a modo inserción
- Cambiar, borrar, copiar y pegar texto
- Deshacer cambios
- Buscar texto en un documento
- Grabar y salir

Introducción a vim

- Nueva versión de **vi**, el editor de texto estándar de Unix.
  - El comando **vi**, ejecuto **vim** por default
- **gvim**: Versión gráfica de **vim**
- Ventajas:
  - *Velocidad*: Mucho atajos disponibles
  - *Simplicidad*: No hay dependencia de mouse/GUI
  - *Disponibilidad*: Incluido en casi todos los Unix
- Desventajas:
  - *Dificultad*: Curva de aprendizaje
  - Los atajos no son intuitivos

vim: Un editor con modos

- El comportamiento del teclado varia dependiendo el *modo* de vi
- Existen tres modos:
  - *Modo comando* (default): Mover cursos, copiar/pegar, cambiar de modo
  - *Modo inserción*: Modificar texto
  - *Modo Ex*: Guardar, salir, etc
- **Esc** sale del modo actual
- **EscEsc** siempre regresa a modo comando

Primeros pasos con vim

- Conocimientos mínimos indispensables para usar vim:
  - Abrir un archivo
  - Modificar un archivo (modo inserción)
  - Grabar un archivo (modo ex)

Abriendo un archivo con vim

- Para iniciar **vi**:
  - **vim archivo**
  - Si el archivo existe, el contenido es mostrado
  - Si el archivo no existe, **vi** lo crea y los cambios son guardados por primera vez

Modificando un archivo - Modo inserción

- **i** para comenzar a insertar texto en la ubicación del cursor
- Otras opciones para insertar texto:
  - **A** inserta al final de la línea
  - **I** inserta al inicio de la línea
  - **o** inserta una nueva línea (abajo)
  - **O** inserta una nueva línea (arriba)

Grabar y Salir - Modo Ex

- **:** para ara entrar en modo ex
  - Crea una linea de comandos en la parte inferior de la pantalla
- Comandos de escritura/salida comunes
  - **:w** Guardar
  - **:wq** Guardar y salir
  - **:q!** Salir sin guardar

Usando el modo comando

- Modo por default en **vi**
- Teclas describen movimientos y comandos de manipulación de texto
- Para repetir comandos se precede por un número
- Ejemplos:
  - **Flecha de navegación derecha** Mueve el cursor un caracter a la derecha
  - **5, Flecha de navegación derecha** Mueve el cursor cinco caracteres a la derecha

Navegando en un documento

- Navegar caracter por caracter: **h**, **j**, **k**, **l**
  - Las flechas de navegación no funcionan en conexiones remotas y sistemas viejos
- Desplazarse por palabra: **w**, **b**
- Desplazarse por oración: **)**, **(**
- Desplazarse por párrafo: }, {
- Saltar a la linea x: **xG**
- Saltar al final de archivo: **G**

Buscando y Reemplazando

- Buscar es igual que con el comando **less**
  - /, **n**, **N**
- Buscar/Reemplazar como en **sed**
  - Afecta solo la línea actual
  - Usa x o y rangos o % para todo el archivo
    - :1,5s/gato/perro/
    - :%s/gato/perro/gi

Deshaciendo cambios

- **u** deshace el último cambio
- **U** Restaura la última línea
- **Ctrl-r** Rehace el último deshacer

Fin del tema

- Dudas
- Resumen
  - Tres modos de **vi**
  - Mover el cursos y entrar en modo inserción
  - Cambiar, borrar, pegar y cortar texto
  - Deshacer cambios
  - Buscar en el documento
  - Grabar y Salir

Utilerías para el Procesamiento de Texto

Objetivos

Al termino de este tema, será capaz de:

- Usar las utilerías para extraer, analizar y manipular texto

Utilerías para la extracción de texto

- Contenido de archivos: **more**, **less** y **cat**
- Extracto de archivos: **head** y **tail**
- Extraer por columna: **cut**
- Extraer por palabra: **grep**

Ver el contenido de archivos

- **cat**: muestra el contenido de uno o mas archivos en la STDOUT
  - Varios archivos se pueden concatenar
- **less**: muestra archivos o la STDIN de forma paginada.
  - Comandos útiles durante el despliegue:
    - **/texto** busca por el texto
    - **n/N** busca la siguiente/previa ocurrencia
    - **v** abre el archivo en un editor de texto
- **less** es usado por el comando **man**

Ver extracto de archivos

- **head**: Muestra las primeras 10 líneas de un archivo
  - Usar -**n** para cambiar el número de líneas a mostrar
- **tail**: Muestra las últimas 10 líneas de un archivo
  - Usar -**n** para cambiar el número de líneas a mostrar
  - Usar -**f** para mostrar las últimas líneas agregadas a un archivo
    - Muy útil para monitorear archivos de registros del sistema

Extraer texto con grep

- Muestran líneas que concuerdan con un patrón
  $ **grep 'paco' /etc/passwd**
  $ **date –help | grep year**
- **-i** Ignora si las letras son mayúsculas o minúsculas.
- **-n** Muestra cada línea de salida con el número de línea de su archivo de entrada correspondiente.
- **-v** Invierte el sentido o de la concordancia, para seleccionar las líneas donde no las hay.
- **-AX** Incluye X número de líneas antes de la concordancia.
- **-BX** Incluye X número de líneas después de la concordancia.

Extraer texto por columna con cut

- Despliega las columnas especificadas de los archivos o de la STDIN
  $ **cut -d: -f1 /etc/passwd**
  $ **grep root /etc/passwd | cut -d: -f7**
- **-d** Especifica el delimitador (por default es TAB)
- **-f** Indica la columna a mostrar.
- **-c** Corta por caracteres.
  $ **cut -c2-5 /usr/share/doct/words**

Utilerías para analizar texto

- Estadísticas: **wc**
- Ordenar texto: **sort**
- Comprar archivos: **diff** y **patch**
- Revisión ortográfica:**aspell**

Obteniendo estadísticas con wc (word count)

- Cuenta palabras, lineas, bytes y caracteres.
- Usa archivos o la STDIN

  $ **wc historia.txt**

  60 646 4133 historia.txt
- **-l** para contar líneas
- **-w** para contar palabras
- **-c** para contar solo bytes
- **-m** para contar separadores de palabra

Ordenando texto con sort

- Ordena texto y lo despliega en la STDOUT. El archivo original no cambia.
  $ **sort [opciones] [archivo(s)]**
- Opciones:
  - **-r** ordena en orden inverso.
  - **-n** ordena con base numérica.
  - **-f** ignora las mayúsculas y minúsculas.
  - **-t c** utiliza el caracter *c* como delimitador.
  - **-k X** ordena por el campo delimitado con *c* el campo **X**

Eliminando lineas duplicadas con sort y uniq

- **sort -u**: elimina lineas duplicadas.
- **uniq**: elimina las lineas duplicadas adyacentes de la STDIN
  - **-c** cuenta el numero de lineas repetidas
  - Si se usa en combinación con sort es mas efectivo:
    $ **sort lista_usuarios.txt|uniq -c**

Comparando archivos con diff

- Busca diferencias entre dos archivos

  $ **diff foo.conf-mala foo.conf-buena**

  ```
  5c5
  < use_widgets = no
  ---
  > use_widgets = yes
  ```

  - Indica una diferencia en la linea 5.
- En ambiente gráfico se puede usar **gvimdiff**
  - Incluido en el paquete *vim-X11*

Replicando cambios en archivos con patch

- La salida del comando **diff** puede ser usado para generar *parches*.
  - **-u** Emplea el formato de salida unificado usado en archivos *patch (parches)*
- **patch** replica los cambios en otros archivos.
  - **-b** Se usa para generar un respaldo automático.

```
$ diff -u foo.conf-mala foo.conf-buena > foo.patch
$ patch -b foo.conf-mala foo.patch
```

Revisión ortográfica con aspell

- Revisión ortográfica interactiva:
  $ **aspell check historia.txt**
- Revisión no interactiva:
  $ **aspell list** $<$ **historia.txt**
  $ **aspell list** $<$ **historia.txt** | **wc -l**

Utilerías para la manipulación de texto

- tr (**tr**anslate)
  - Convierte caracteres
  - Solo lee de la STDIN
    $ **tr 'a-z' 'A-Z' < minúsculas.txt**
- **sed**
  - **s**tream **ed**itor
  - Realiza búsquedas/reemplazos en un flujo de texto
  - Normalmente no afecta el archivo origen
  - **-i.bak** para realizar un respaldo antes de alterar el archivo original

Ejemplos sed

- Siempre usar comillas en las instrucciones de reemplazo
- Direccionamiento **sed**
  - **sed 's/perro/gato/g' mascotas**
  - **sed '1,50s/perro/gato/g' mascotas**
  - **sed '/laika/,/calcetín/s/perro/gato/g' mascotas**
- Para ejecutar varias instrucciones:

- **sed 's/perro/gato/' -e s/hi/lo/' mascotas**
- **sed -f cambios mascotas**

Caracteres especiales para búsquedas avanzadas

Expresiones Regulares

- ^ representa el inicio de línea
- **$** representa el final de línea
- Bash permite el uso de
  - **[abc], [^abc]**
  - **[[:upper:]]**
- Las expresiones regulares son usadas por los comandos **grep, sed, less**, etc.

Fin del tema

- Dudas
- Resumen
  - Utilerías para la extracción de texto
    - **cat, less, head, tail, grep, cut**
  - Utilerías para analizar texto
    - **wc, sort, uniq, diff, patch**
  - Utilerías para manipular texto
    - **tr, sed**
  - Expresiones regulares
    - **^, $, [abc], [^abc], [[:alpha:]], [^[:alpha:]], etc**

Utilerías para la búsqueda y procesamiento de archivos

Objetivos

Al termino de este tema, será capaz de:

- Uso del comando **locate**
- Uso de **find**

locate

- Busca en una base de datos rutas y archivos en el sistema
  - La base de datos tiene que ser actualizada por el administrador
  - Se busca la ruta completa, no solo el archivo
- Usualmente solo busca en directorio donde el usuario tiene permisos

Ejemplos locate

- **locate dummy**
  - Buscar archivos que contengan la cadena *dummy* en el nombre o el la ruta
- **locate -r '\.dummy$'**
  - Busca archivos que tienen con *.dummy*
- Opciones adicionales:
  - **-i** ignora la distinción de mayúsculas y minúsculas
  - **-n X** muestra únicamente las X ocurrencias

find

- **find [directorio...] [criterio...]**
- Busca en estructuras completas de directorio en tiempo real
- Notablemente mas lento pero más preciso que `locate`
  - CWD es usado si no se indica un directorio
  - Todos los archivos son encontrados si no se especifica un criterio
- Permite la ejecución de comandos a los archivos encontrados
- Solo permite la búsqueda en directorios donde el usuario tiene permisos

Ejemplos básicos **find**

- **find -name naviad.png**
  - Busca por archivos llamados navidad.png
- **find -iname navidad.png**
  - Busca los archivos ignorando la distinción de mayúsculas y minúsculas, navidad,png, Navidad.png, NAVIDAD.png
- **find -user paco -group paco**
  - Buscar los archivos que pertenecen al usuario paco y al grupo paco

Operadores lógicos en **find**

- La búsqueda con más de un criterio esta permitida
- El criterio puede ser adicional o puede negarse con **-o** y **-not**
- Los paréntesis se usan para determinar el orden lógico de la búsqueda, pero deben de protegerse para ser usados en el bash
- Ejemplos:
  - find -user paco -not -group paco
  - find -user jose -o -user maria
  - find -not $-$*userjose* $-$ *o* $-$ *usermaria*

Buscando archivos por tamaño

- Podemos buscar por tamaño del archivo:
  - **find -size 1024k** Busca archivos de 1024k exactamente
  - **find -size +1024k** Busca archivos de más de 1 megabyte
  - **find -size -1024k** Busca archivos de menos de 1 megabyte

Buscando archivos por fechas

- **find** permite buscar por inode timestamps Podemos buscar por tamaño del archivo:
  - **-atime** cuando los archivos fueron leidos
  - **-mtime** cuando los archivos fueron modificados
  - **-ctime** cuando el contenido o los metadatos fueron modificados
- Valores dados por día
  - **find -ctime -10**
    - Archivos creados en los últimos 10 días

Ejecutando comandos con **find**

- Es posible ejecutar comandos por cada archivo encontrado con el comando **find**
  - El comando debe ser precedido por la opción -**exec** o -**ok**
    - -ok pide confirmación antes de ejecutar el comando por cada archivo
  - Los comandos deben terminar con **espacio**\;
  - Las llaves { } son usadas para almacenar el nombre del archivo
- Ejemplo:
  - `find -size +102400k -ok gzip {} \;`
  - `find / -name core -exec rm {} \;`

Ejemplos de find

- `find -name "*.confexec cp {} {}.orig {} \;`
  - Respalda los archivos de configuración, añadiendo la extensión .orig a cada archivo
- `find /tmp -ctime +3 -user paco -ok rm {} {} \;`
  - Busca los archivos temporales de paco con mas de tres días, solicitando confirmación antes de borrarlos
- `find * * -perm +o+w -exec chmod o-w {} \;`
  - Corrige los permisos en mi directorio hogar

Fin del tema

- Dudas
- Resumen
  - Usar **locate** para buscar rápidamente archivos que no son nuevos
  - Usar find para buscar con base en un criterio específico y opcionalmente ejecutar un comando en los archivos encontrados

2 Introducción a Unix/Linux

3 Adminstración de GNU/Linux

- Configuración básica del sistema
- Administración de usuarios, grupos y permisos
- Investigación y administración de procesos
- Herramientas básicas para la administración del sistema
- Inicio del sistema

Configuración Básica del Sistema

Objetivos

Al termino de este tema, será capaz de:

- Ajustar la fecha y hora del servidor.
- Configurar correctamente los parámetros de red.

Configuración de redes TCP/IP

Definición

Conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre redes de computadoras. [a]

[a] http://www.w3schools.com/tcpip/default.asp

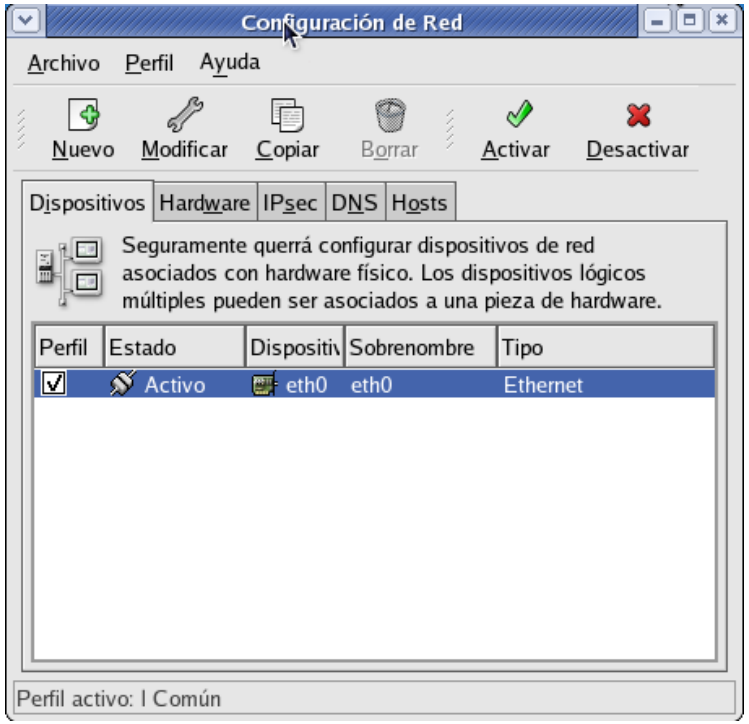
- Ajustes más importantes
  - Dirección IP.
  - Activación del dispositivo de red.
  - Configuración del DNS.
  - Default gateway.

Administrando conexiones Ethernet

- Nombre de los dispositivos de red: *eth0, eth1, eth2, ethN...*
  - Es posible asignar diferentes direcciones de ip a un mismo dispositivo de red.
  - Alias *eth0:1, eth0:1, eth0:2*
  - Alias son reconocidos como interfaces independientes.
- Para ver la configuración de los dispositivos de red: **ifconfig [ethN]**
- Para iniciar un dispositivo de red: **ifup ethN**
- Para detener un dispositivo de red: **ifdown ethN**

Interfaz gráfica de configuración de red

- `system-config-network`
- *Aplicaciones* > *Configuración del sistemas* > *Red*
- Activar/Desactivar interfaces.
- Asignar direccionamiento IP / DHCP.
- Modificar parámetros del DNS.
- Modificar gateway por default.



Archivos de configuración de red (I)

- La configuración de red, como casi cualquier archivo de configuración en Unix es almacenada en archivos de texto
  - `/etc/sysconfig/network-scripts/ifcfg-ethN`
  - La lista completa de opciones se encuentra en `/usr/share/doc/initscripts-*/sysconfig.txt`

Configuración Dinámica	Configuración Estática
DEVICE=ethN	DEVICE=ethN
HWADDR=00:11:22:33:44:55	HWADDR=00:11:22:33:44:55
BOOTPROTO=dhcp	IPADDR=192.168.1.1
ONBOOT=yes	NETMASK=255.255.255.0
Type=Ethernet	GATEWAY=192.168.1.254
	ONBOOT=yes
	Type=Ethernet

Archivos de configuración de red (II)

- Parámetros globales almacenados en: `/etc/sysconfig/network`
  - Algunos de estos parámetros son tomados del DHCP
  - **GATEWAY** puede ser sobreescrito en el archivo `ifcfg`

```
NETWORKING=yes
HOSTNAME=servidor1.dominio.com
GATEWAY=192.168.1.254
```

- La configuración referente al **DNS** se almacena en `/etc/resolv.conf`

```
search dominio.com dominio2.net
nameserver 192.168.2.1
nameserver 192.168.3.1
```

Ajustando la fecha y hora

- GUI: `system-config-date`
  - Aplicaciones > Configuración del sistema > Fecha y hora
  - El ajuste se puede hacer manual o usando el protocolo NTP
  - Es posible agregar servidores NTP
- CLI: `date [MMDDhhmm[[CC]YY][.ss]]`
  - `date 01011330`
  - `date 010113302010.05`

Fin del tema

- Dudas
- Resumen
  - **system-config-network**
  - `/etc/sysconfig/network-scripts/*`
  - **ifup, ifdown**
  - **date** configura fecha/hora desde la línea de comandos
  - **system-config-date** GUI de configuración de fecha/hora

Administración de usuarios, grupos y permisos

Objetivos

Al termino de este tema, será capaz de:

- Ubicar donde se almacena la información de los usuarios, grupos y contraseñas.
- Cambiar de identidad.
- Usar permisos especiales.

UserID y GroupID

- Los nombre de usuario corresponden a un número conocido como userID.
- Mismo caso para los grupos, a cada grupo le corresponde un groupID.
- La información guardada en los discos es almacenada haciendo referencia a este estos números.
- Archivos usados para guardar información de los usuarios:
  - `/etc/passwd`
  - `/etc/shadow`
  - `/etc/group`
  - `/etc/gshadow`

Herramientas para la administración de usuarios

- Herramientas gráficas
  - **system-config-users**
- Linea de comando
  - **useradd**
  - **usermod**
  - **userdel [-r]**

Usuarios y grupos de sistema

- Servicios como el web, impresión usualmente se ejecutan usando una cuenta no privilegiada.
  - Ejemplo: apache, daemon, mail, lp, nobody
- Al ejecutar los programas de este forma, se limita el daño que un programa puede hacer al sistema

Monitorizando los accesos de usuarios

- Usuarios conectados: **w**
- Últimos accesos: **last, lastlog**

Permisos por default

- **umask** es el mecanismo que provee GNU/Linux para establecer los permisos por default a archivos y directorios.
- Permisos por omisión en directorio **777**
- Para archivos es igual que un directorio pero sin el de ejecución.
- El comando **umask** se usa para determinar el modo de creación de archivos.

Permisos especiales

- **suid**: el programa se ejecuta con los permisos del dueño del programa, no con el ejecutor.
- **guid**: el programa se ejecuta con los permisos del grupo del programa.

Example

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 37140 2010-01-26 11:09 passwd
```

- **sticky bit**: archivos creados en este tipo de directorios solo pueden ser eliminados por el dueño y root, a pesar del permiso de escritura del directorio.

Example

```
$ ls -ld /tmp
drwxrwxrwt 17 root root 4096 2010-07-14 11:20 /tmp
```

Fin del tema

- Dudas
- Resumen
  - Información de usuarios es almacenada en `/etc/passwd`
  - Información de grupos es almacenada en `/etc/group`
  - Permisos especiales: Sticky Bit, SetUID, SetGID

Investigación y Administración de Procesos

Objetivos

Cuando se termine el tema usted podrá ser capaz de:

- Explicar lo que es un proceso
- Describir como se administran los procesos
- Usar las herramientas de control de procesos

¿Qué es un proceso?

- Un proceso es un conjunto de instrucciones cargadas en memoria.
  - A cada proceso se le asigna un número llamado PID *(Process ID)*
- Para ver un listado de procesos se usa el comando **ps**
  - **-a** incluye procesos de todas las terminales.
  - **-x** incluye procesos no asignados a terminales.
  - **-e** incluye todos los procesos
  - **-u** muestra la dueño del proceso
  - **-f** muestra al padre del proceso
  - **-o PROPERTY**
    - pid, comm %cpu, %mem, state, tty, euser, ruser.

Encontrando procesos

- **ps** opciones | comando
  - `ps axo comm,tty|grep tty1`
- Por patrones predefinidos: **pgrep**
  - `pgrep -U root`
  - `pgrep -G unix`
- Buscando el nombre del programa: **pidof**
  - `pidof bash`

Señales

- Mecanismo de comunicación entre procesos
  - Se envían directo a los procesos, no requiere interfaz de usuario.
  - Los programas asocian una acción a cada señal.
  - El tipo de señal se especifica usando el nombre o número
    - Señal 15, TERM (deafault) - Finaliza un proceso de manera limpia.
    - Señal 9, KILL - Finaliza un proceso inmediatamente.
    - Señal 1, HUP - Relee los archivos de configuración.
    - **man 7 signal** muestra la lista completa de señales.
- Envio de señales a los procesos
  - Por PID: **kill [señal] pid ...**
  - Por Nombre: **killall [señal] comm ...**
  - Por patrón: **pkill [-signal] patron**

Prioridades

- Asignar prioridades determina el acceso a la CPU.
  - La prioridad es afectada por el valor *nice*
- Rango de valores válidos: -20 a 19 el default es 0
  - A menor valor *nice* mayor prioridad de CPU.
- Para conocer la prioridad: **ps -o comm,nice**
- Para modificar las prioridades:
  - Al inicio de la ejecución del proceso:
    - `$ nice -n 5 comando`
  - Después del inicio:
    - `$ renice 5 PID`
  - Solo *root* puede modificar la prioridad de un proceso.

Monitoreo en linea de procesos

- CLI: **top**
- CLI: **htop**[10]
- GUI: **gnome-system-monitor**
- Características
  - Muestra información de procesos en tiempo real
  - Permite ordenar y enviar señales a los procesos.

[10] paquete no instalado por default, se puede descargar de la siguiente ruta: `http://dag.wieers.com/rpm/packages/htop/`

Control de tareas

- Ejecutar un proceso en background
  - Agregar un ampsersand al final de la línea: **firefox &**
- Para detener de manera temporal un programa en ejecución:
  - Usar **Ctrl-z** o enviar la señal 17 (STOP)
- Administrar el segundo plano (background) o suspender tareas
  - Para listar las tareas en ejecución o suspendidas: **jobs**
  - Continuar con la ejecución en segundo plano: **bg [% num_tarea]**
  - Continuar con la ejecución en primer plano : **fg [% num_tarea]**
  - Enviar una señal: **kill [-SEÑAL] [% num_tarea]**

Calendarización de tareas

- Para tareas de una sola ves usar **at**, para tareas rutinarias usar **crontab**.

Crear	**at** hora	**crontab -e**
Listar	**at -l** hora	**crontab -l**
Detalles	**at -c** num_tarea	N/D
Eliminar	**at -d** num_tarea	**crontab -r**
Editar	N/D	**crontab -e**

- La salida del programa es enviada por correo al usuario.
- *root* puede modificar tareas de otros usuario

Formato archivo crontab

- Cada entrada consiste en cinco campos delimitados por espacio seguidos por un comando.
  - Una tarea por linea.
- Campos: minuto hora, día del mes, mes y día de la semana.
- Los comentarios inician con el caracter #
- Para mas información **man 5 crontab**

Agrupando comandos

- Dos formas disponibles para agrupar comandos:
  - compuesto: **date;who| wc -l**
- Subshell: **(date;who | wc -l)** >> **/tmp/trace**
  - Toda la salida es enviada a la STDOUT y STDERR

Exit Status

- Cada proceso reporta el éxito o fracaso de su ejecución a través de un estado de salida (exit status)
  - 0 para éxito, 1-255 para falla.
  - **$?** almacena el código de salida del último comando ejecutado
- Ejemplo:

```
$ ping -c1 -W1 www.google.com &> /dev/null
$ echo $?
0
```

Operadores condicionales de ejecución

- Con base en el código de salida los comandos pueden o no ejecutarse
  - && equivale a AND THEN
  - || equivale a OR ELSE
- Ejemplo OR ELSE:

```
$ grep -q usuario_no_valido /etc/passwd || echo ‘‘Usuario
no existe’’
Usuario no existe
```

- Ejemplo AND THEN:

```
$ ping -c1 -W2 servidor1 &> /dev/null \
&& echo ‘‘Servidor1 ok’’ \
|| $(echo ’Servidor1 inalcansable’; exit 1)
```

El comando test

- Evalúa sentencias booleanas.
  - 0 para verdadero
  - 1 para falso
- Ejemplos en formato largo:

```
test ‘‘$A’’ = ‘‘$B’’ && echo ‘‘Cadenas son iguales’’‘
test ‘‘$A’’ -eq ‘‘$B’’ && echo ‘‘Enteros iguales’’‘
```

- Ejemplos en formato corto:

```
$ [‘‘$A’’ = ‘‘$B’’] && echo ‘‘Cadenas son iguales’’‘
$ [‘‘$A’’ -eq ‘‘$B’’] && echo ‘‘Enteros iguales’’‘
```

Test sobre archivos

- Pruebas sobre archivos:
  - **-f** valida que el archivo exista y sea regular.
  - **-d** valida que el archivo exista y sea un directorio.
  - **-x** valida que el archivo exista y sea un ejecutable.

```
[ -f ~/lib/functions ] && source ~/lib/functions
```

Scripting: sentencia if

- Ejecuta instrucciones basado en el código de salida de un comando.

```
if ping -c1 -w2 servidor1 &> /dev/null; then
echo 'Servidor1 ok'
elif grep ``servidor1''  /mantenimiento.txt &> /dev/null;
then
echo 'Servidor1 se encuentra en mantenimiento'
else
echo 'Servidor1 INALCANZABLE!'
exit 1
fi
```

Fin del tema

- Dudas
- Resumen
  - Un proceso es cualquier conjunto de instrucciones en memoria
  - Los procesos se pueden administrar con los comandos: **ps, kill, top, gnome-system-monitor**
  - Una tarea se suspende con **Ctrl-z**, y administrar con **fg, bg**

Herramientas Básicas para la Administración del Sistema

Objetivos

Al termino de este tema será capaz de:

- Identificar los servicios de Linux y su estado así como administrar los niveles de ejecución (*runlevels*) en donde se inician o detienen dichos servicios
- Instalar software usando diferentes métodos

Administrando de Software

- El software se distribuye como paquetes RPM
  - Fácil de instalar y desinstalar
  - La información sobre el software es almacenado en una base de datos local
- Los paquetes se encuentran en repositorios con varios nodos distribuidos mundialmente.

Administración de software con yum

- Front-end del comando **rpm**, reemplazo del comando **up2date**
- Archivos de configuración: `/etc/yum.conf` y `/etc/yum.repos.d/`
- Se usa para instalar, listar y eliminar software:
  - Instalar/Desinstalar/Actualizar:
    - **yum** *install* nombre_paquete
    - **yum** *remove* nombre_paquete
    - **yum** *update* nombre_paquete
  - Buscar paquetes:
    - **yum search** patron_búsqueda
    - **yum list** *(all|available|extras|installed|recent|updates*
    - **yum info** *nombre_paquete*

Administración de software con RPM

- Componentes:
  - Base de datos local
  - El comando **rpm** y algunos adicionales.
  - Paquetes
- Funciones principales
  - instalar/desinstalar
  - búsquedas
  - verificación

Instalación y Desinstalación de software

- Opciones principales de RPM:
  - Instalar: **rpm -i, –install**
  - Actualizar: **rpm -U, –upgrade**
  - Freshen: **rpm -F, –freshen**
  - Desinstalar: **rpm -e, –erase**
- Para mostrar mensaje en pantalla:
  - **-v, -h**
- Soporte de URL: `ftp://, http://`

Actualización del Kernel

- Es buena práctica actualizar el kernel de forma periodica.
- No usar **rpm -U** o **rpm -F!**
  - rpm -ivh kernel-version.arch.rpm
  - Iniciar el sistema (boot) con el nuevo kernel y validar el correcto funcionamiento del sistema.
  - Usar el kernel anterior si hay algún problema.
  - Si no hay algún problema **rpm -e kernel-oldversion**

Consultas rpm

- Opciones de paquetes instalados:
  - **rpm -qa** lista todos los paquetes instalados
  - **rpm -qf archivos** muestra a que paquete pertenece el archivo
  - **rpm -qi paquete** muestra información general sobre el paquete
  - **rpm -ql paquete** muestra el contenido del paquete
- Opciones para paquetes no instalados
  - **rpm -qlp paquete_i386.rpm**
  - **rpm -qip paqute_i686.rpm**

Fin del tema

- Dudas
- Resumen
  - Administrar servicios
  - ¿Cuales son las funciones básicas de RPM?
  - ¿Que opciones son usadas en el comando **rpm** para instalar un kernel?
  - Administración de paquetes con yum
  - Relación entre **yum** y **rpm**
  - Uso de yum

Inicio del sistema

Objetivos

Al termino de este tema será capaz de:

- Explicar el proceso de inicio de un sistema GNU/Linux
- Entender el rol del GRUB
- Entender el rol el proceso init
- Controlar los servicios System V

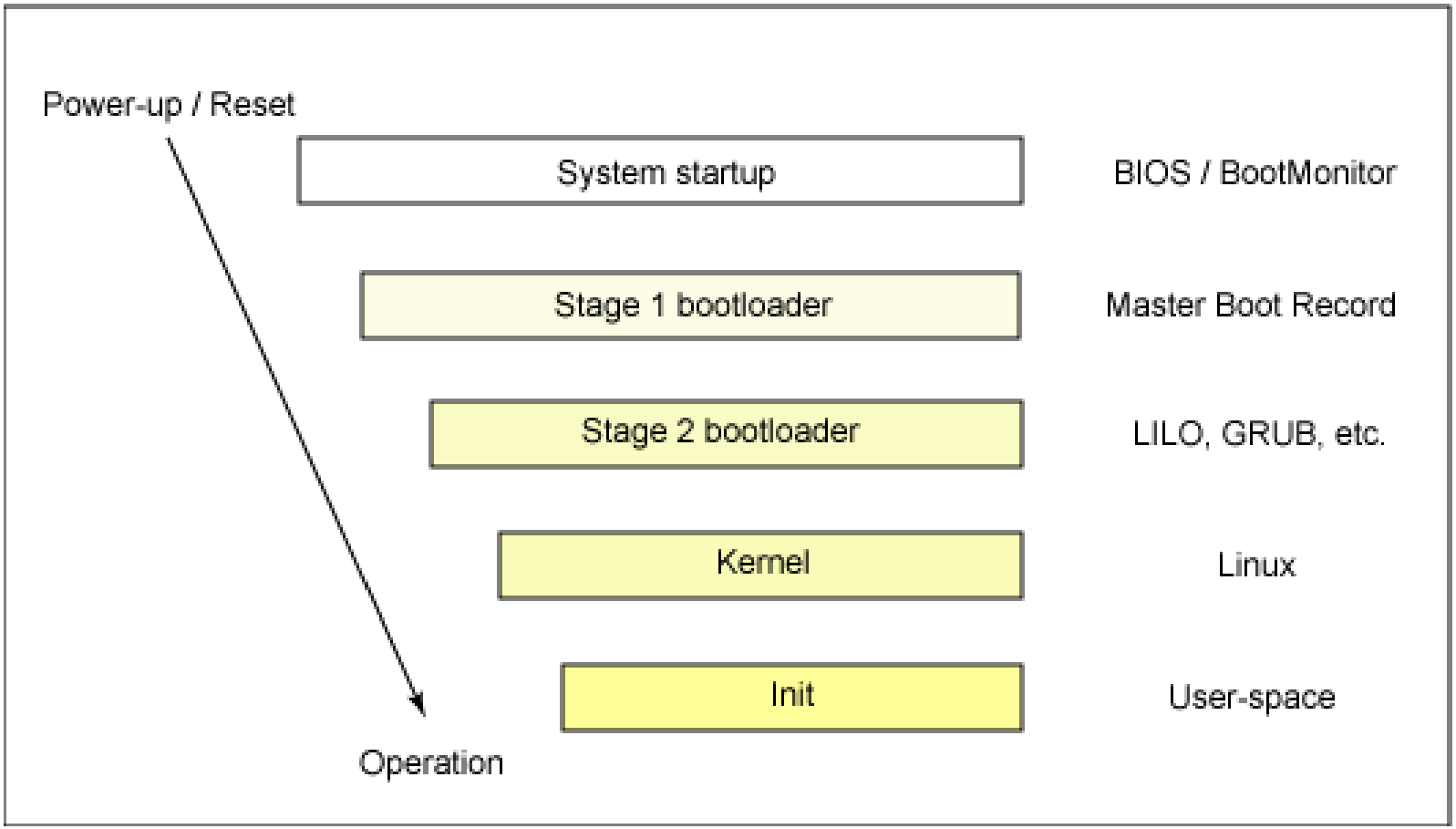
Proceso de arranque (boot)

- Inicio del BIOS
- Cargar del sector de arranque (Boot loader)
- Carga del Kernel
- inicio de **init** y entrar aun nivel de ejecución en particular ejecutando:
  - `/etc/rc.d/rc.sysinit`
  - `/etc/rc.d/rc` y `/etc/rc.d/rc?.d/`
  - `/etc/rc.d/rc.local`
  - Si esta configurado inicio del sistema X Windows

GRUB y grub.cof

- GRUB "GRand Unified Bootloader"
  - Interfaz de linea de comando disponible al inicio del proceso de arranque
  - Soporta sistemas de archivos ext2/ext3, ReiserFS, JFS, FAT, minix, o FFS
  - Protección a través de MD5
- Cambios en el archivo `/etc/grub.conf` tiene efecto inmediato.
- Si el MBR en el dispositivo `/dev/hda` se corrompe, se puede reinstalar el grub con el comando:
  - `/sbin/grub-install /dev/hda`

Proceso de arranque en GNU/Linux



http://www.ibm.com/developerworks/linux/library/l-linuxboot/

Administrando Servicios

- ¿Qué es un servicio?
- Interfaz gráfica para la administración de servicios
  - **system-config-services**
- Interfaz en la línea de comando:
  - **service**
  - **chkconfig**

Administración de Windows Server 2008

4. Administración de Windows Server 2008
   - Conceptos previos
   - Instalación de Windows Server 2008
   - Proceso de arranque de Windows Server 2008

Conceptos previos

Objetivos

Al termino de este tema, será capaz de:

- Explicar conceptos como Windows API
- Diferenciar entre las versiones de Windows Server 2008
- Explicar la historia de Windows Server 2008
- Mencionar las principales características de Windows Server 2008
- Identificar las mejoras de Windows Server 2008 con respecto a Windows Server 2003

Características

- Evolución de Windows NT 6.0
- Sucesor de Windows Server 2003
- Sistema operativo
  - 32 y 64 bits
  - Multiusuario
  - Multitarea
  - Multiproceso
  - Multihilo
- Soporte para
  - Sistemas multiprocesador (SMP)
  - AGP
  - USB
  - Administración de energía (APM)
  - Procesadores Intel y RISC
  - Varios sistemas de archivos (FAT, NTFS...)

Mejoras

- *Server Manager* reemplazo de las MMC de Windows 2003.
- Controladores de Dominio *solo lectura*.
- *Core Installation*.
- *Windows PowerShell* también disponible en Windows Vista, provee varias utilerías llamadas *cmdlets*.
- Nuevo *Print Management snap-in*
- Internet Information Services 7
- Internet Protocol version 6
- Server virtualization
- Windows Deployment Services
- Windows Firewall
- Reliability and Performance Monitor

Versiones

- Standard Edition (x86 y x8664)
- Enterprise Edition (x86 y x8664)
- Datacenter Edition (x86 y x8664)
- HPC Server (remplaza Windows Compute Cluster Server 2003)
- Web Server (x86 y x8664)
- Storage Server (x86 y x8664)
- Small Business Server (Nombre clave "Cougar") (x8664) para pequeñas empresas
- Windows Essential Business Server (Nombre clave "Centro") (x8664) para empresas de tamaño medio
- Windows Server 2008 para sistemas basados en Itanium
- Foundation Server

http://www.microsoft.com/windowsserver2008/en/us/r2-editions\discretionary{-}{}{}overview.aspx

Windows System Releases

Producto	Versión Interna	Fecha de Liberación
Windows NT 3.1	3.1	Julio 1993
Windows NT 3.5	3.5	Septiembre 1994
Windows NT 3.51	3.51	Mayo 1995
Windows NT 4.0	4.0	Julio 1996
Windows 2000	5.0	Diciembre 1999
Windows XP	5.1	Agosto 2001
Windows Server 2003	5.2	Marzo 2003
Windows Vista	6.0 (Build 6000)	Enero 2007
Windows Server 2008	6.0 (Build 6001)	Marzo 2003

Windows Internals 5th Edition P.1

Windows API

Definición

La **interfaz de programación de aplicaciones**, cuyo acrónimo en inglés es **API** (application programming interface), es un conjunto de funciones residentes en bibliotecas (generalmente dinámicas, también llamadas DLL por sus siglas en inglés) que permiten que una aplicación corra bajo el sistema operativo Windows.

Categorías principales:

- Servicios base
- Servicios de interfaz de usuario
- Servicios gráficos y de multimedia
- Mensajería y comunicación
- Red
- Servicios web

Fin del tema

- Dudas
- Resumen
  - Características de Windows Server 2008
  - Versiones de Windows Server 2008
  - Windows API

Instalación de Windows Server 2008

Objetivos

Al termino de este tema, será capaz de:

- Evaluar la factibilidad técnica de la instalación de Microsoft Windows 2008
- Instalar el sistema operativo Microsoft Windows 2008

Pasos previos

1. Comprobar la compatibilidad del HW
2. Obtener controladores compatibles
3. Determinar la configuración de la red
4. Definir la política de particionado
5. Escoger el tipo de sistema de archivos
6. Elegir una contraseña para el administrador
7. Elegir esquema de licenciamiento
   - Per server
   - Per device or per user

Hardware soportado

- Se encuentra en la Hardware Compatibility List (HCL)
  `http://www.windowsservercatalog.com/`
- Actualizada trimestralmente
- Aparecen los dispositivos que
  - Funcionan correctamente con Windows 2008
  - Existe controladores suministrado por Microsoft

Fases de la instalación

- Arranque
  - CD-Rom
  - Disquetes
- Fase de copia
  - Se copian los archivos a una carpeta temporal
- Primer inicio
  - Versión restringida del núcleo
  - Modo texto
  - Aceptar la licencia de usuario final
  - Seleccionar/crear la partición de instalación
  - Formatear/convertir la partición
  - Escoger directorio de instalación

Fases de la instalación

- Segundo inicio
  - Modo gráfico
  - Obtener información sobre el servidor
  - Escoger teclado
  - Elegir zona horaria
  - Seleccionar componentes a instalar
  - Instalación de la red
  - Establecer contraseña de administrador
- Fin de la instalación
  - Reiniciar
  - Primer arranque
  - Instalación de controladores adicionales

Fin del tema

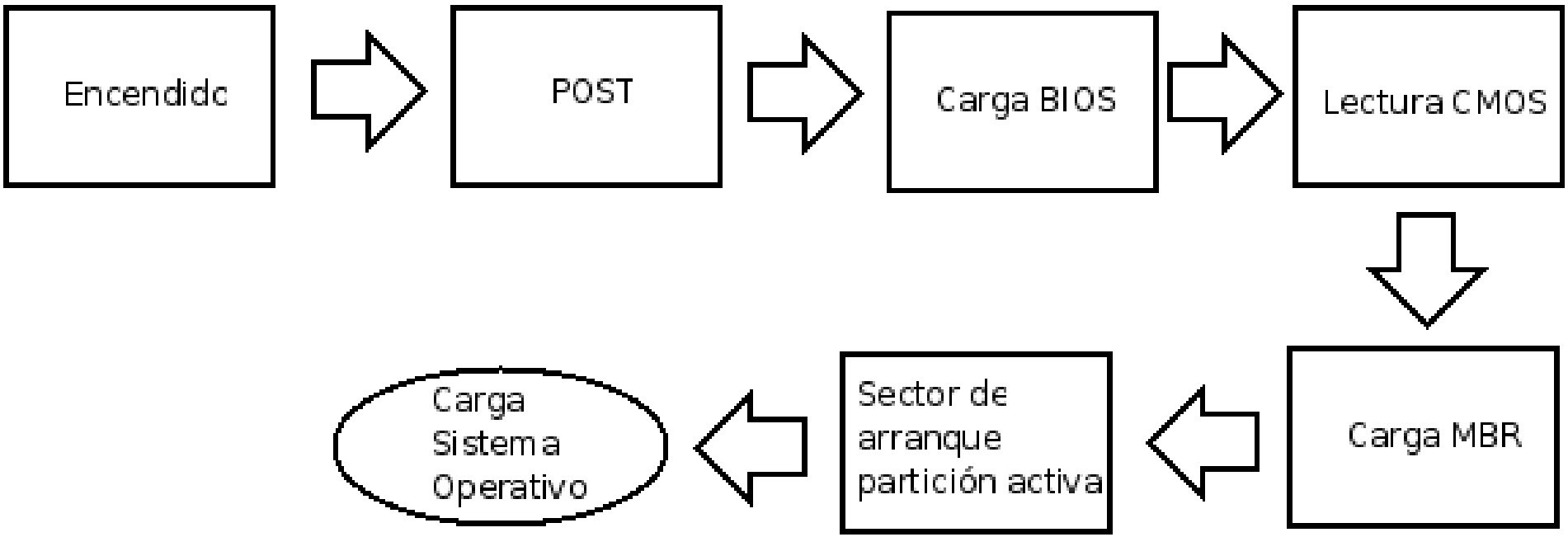
- Dudas
- Resumen
  - Hardware Compatibility List (HCL)
  - Esquemas de licenciamiento
    - *Per server*
    - *Per device or per user*

Proceso de arranque de Windows Server 2008

Objetivos

Al termino de este tema, será capaz de:

- Comprender el proceso de arranque de Microsoft Windows Server 2008
- Diagnosticar problemas al arranque de Microsoft Windows Server 2008


Encendidc
POST
Carga BIOS
Lectura CMOS
Carga MBR
Sector de arranque partición activa
Carga Sistema Operativo

Arranque de Windows 2k3

- POST
- Selección del Sistema Operativo
  - Se carga ntldr
  - Se carga el controlador del sist. de archivos
  - Comprueba el contenido de boot.ini
  - Muestra un menú con los SO
  - Si se selecciona otro SO -> bootsect.dos
  - Si se selecciona W2K3
    - Carga de ntdetect.com
    - Detección del HW presente en el equipo
- Carga del kernel
  - *ntdetect.com* carga ntoskrnl.exe
  - Se carga la configuración de HW (registro)
  - Se cargan controladores con valor 0x0 en el registro

Arranque de Windows 2k3 (II)

- Inicialización del kernel
  - Se inicializa el kernel
  - Se crea la entrada HARDWARE en el registro
  - Se inicializan los drivers
  - Se cargan controladores con valor 0x1
- Carga de servicios
  - Se carga el session manager (smss.exe)
  - Se ejecutan los programa de la entrada
  - BootExecute del registro
  - Se inician los distintos subsistemas
  - Se ejecuta winlogon.exe
  - Aparece la ventana Ctrl+Alt+Del
  - Se cargan servicios con valor 0x2