Sistemas Operativos
2º Curso – Grado en Ingeniería Informática

Tema 1:

Estructuras de SOs

José Antonio Gómez Hernández, 2016.

Componentes de un SO

- Estructuras actuales de SOs:
  - Monolítica
  - Máquinas virtuales
- Tipos de sistemas operativo
- Soporte de Linux a la virtualización
- Seminario: Mecanismos de extensión de Linux - LKM (Linux Kernel Modules).
- Arquitecturas de SOs de producción

1.

Estructuras de SOs

Servicios y organización de los mismos

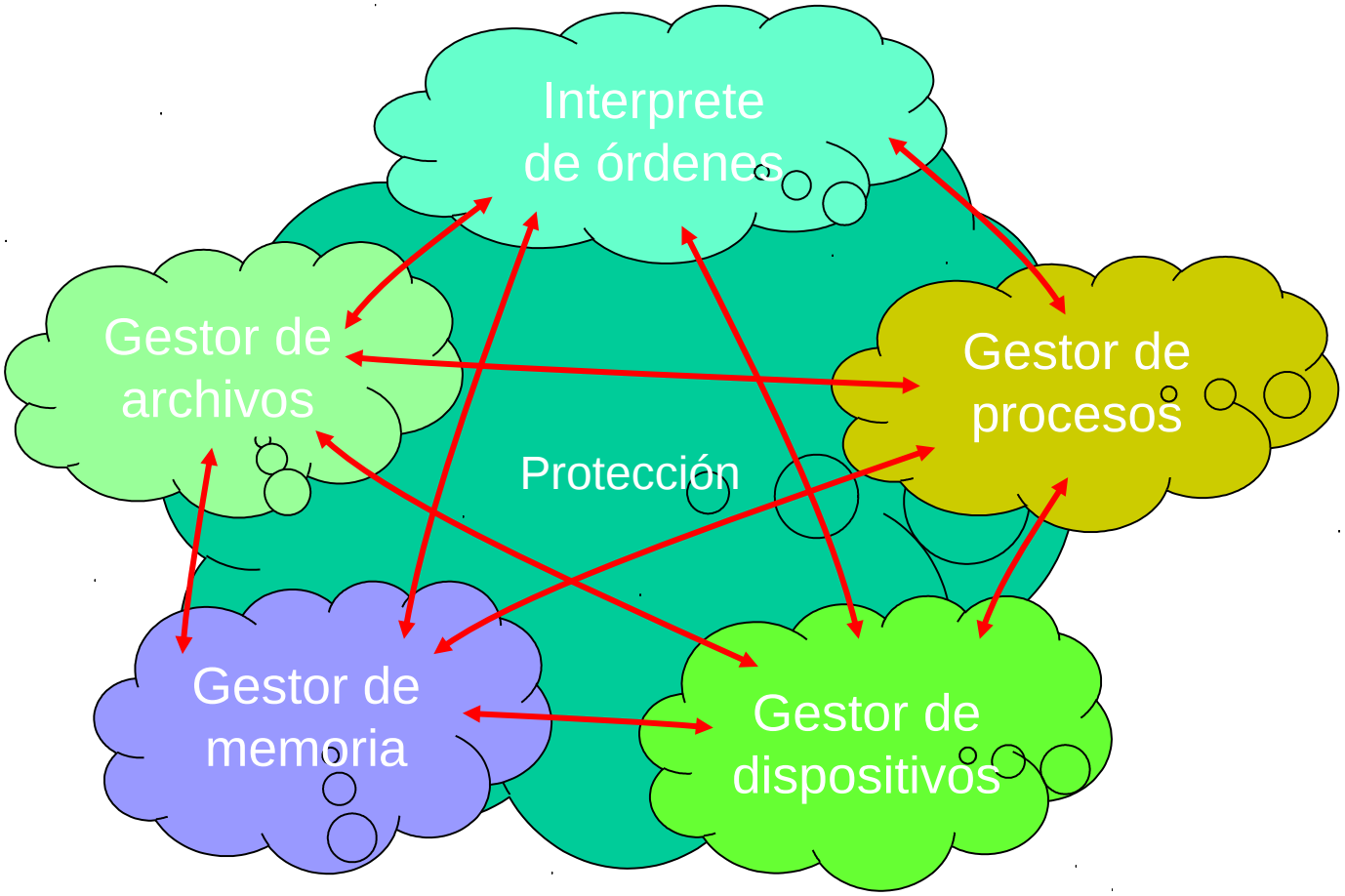
Servicios de un SO

▷Algunos de los servicios que suministra el SO a los procesos de aplicación:

- Ejecutar diferentes actividades
- Acceso a información permanente
- Acceso a los dispositivos
- Suministrar memoria para ejecutarlos
- El uso de los recursos debe ser seguro
- Interfaz de acceso a los servicios del SO

▷Estos servicios se suelen integrar en componentes.

Componente de un SO



La gran cuestión

▷Un SO consta de los elementos vistos entre los cuales existen muchas y complejas relaciones.

▷Las preguntas importantes son:

- ¿Cómo se organiza todo esto?
- ¿Cuales son los procesos y dónde están?
- ¿Cómo cooperan esos procesos?

▷Es decir, **¿cómo construir un sistema complejo que sea eficiente, fiable y extensible?**

Modelo de procesos

- La estructura de SO multiprogramado propuesta en el Tema 0 no contempla la existencia de procesos ejecutándose el modo kernel.
- El problema es que si no hay procesos del SO ejecutándose en modo kernel, éste no se ejecuta, provocando poca responsibidad especialmente en el tratamiento de dispositivos.
- En Unix, el problema se abordó introduciendo procesos, **demonios**, que ejecutan labores de sistema en modo kernel.
- En Linux, se han sustituido por **hilos kernel**.

Características

- **Eficiente**: el SO debe ser lo más eficiente posible para no degradar el funcionamiento gobal del sistema.
- **Fiable**: Dos aspectos:
  - **Robusto** – El SO debe responder de forma predecible a condiciones de error.
  - El SO debe protegerse activamente a si mismo y a los usuarios frete a acciones accidentales o malintecionadas.
- **Extensible**: Deberíamos poder añadir/modificar funcionalidad del SO de forma flexible.

Arquitectura de SOs

▷ Para alcanzar estos objetivos se ha recurrido generalmente a la **arquitectura software**: estructura con la cual construimos el software (sus componentes y sus interacciones).

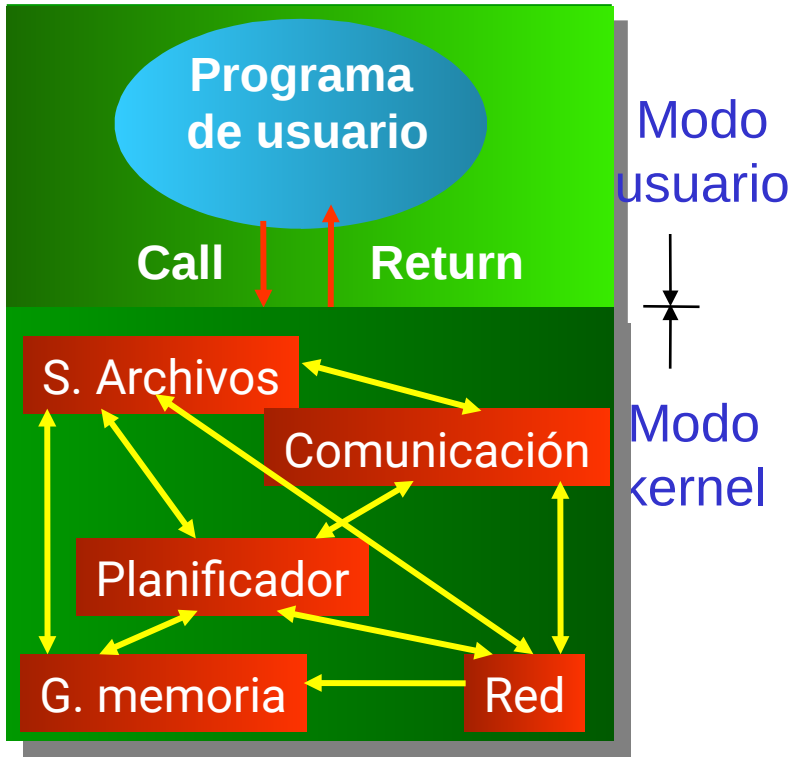
▷Debemos distinguir entre:

- **Arquitectura de diseño**: cómo se diseña.
- **Arquitectura de ejecución**: cómo se construye y ejecuta el sistema.

▷ Ambas arquitecturas pueden diferir para un mismo sistema. La arquitectura de ejecución de muchos SOs actuales es esencialmente monolítica si bien su arquitectura de diseño puede ser de capas, microkernel, etc.

Estructura monolítica

- Todos los componentes del sistema se ejecutan en modo kernel. El SO es un único programa–ejecutable). Las relaciones entre ellos son complejas.
- El modelo de obtención de servicios es la llamada a procedimiento "protegido" que tiene el menor coste.



Monolíticos: caracteristicas

▷ Son **difíciles de comprender**, ya que son un único programa (decenas de MB), por tanto, difíciles de modificar y mantener.

▷ **No confinamiento de errores**: un fallo de algún módulo puede provocar la “caída” del sistema.

- P.ej, Pantalla azul de Windows o panic de Unix.

▷ Por esto, los diseñadores han buscado mejores formas de estructurar un SO para simplificar su diseño, construcción, depuración, ampliación y mejora de sus funciones.

Monolíticos: Linux

- Los kernel actuales de Linux son de este tipo (al igual que la mayoría de sistemas operativos de producción) por razones de eficiencia.
- Esto no quiere decir que el código fuente del sistema no esta estructurado (ver directorio */usr/src/linux/*).
- Indica que el programa ejecutable correspondiente en complejo, en componente e interacciones. (http://www.makelinux.net/kernel_map/)

Virtualización

"Virtualización es una tecnología que combina o divide recursos de computación para presentar uno o varios entornos de operación utilizando metodologías como particionamiento o agregación ya sea hardware o software, simulación de máquinas completa o parcial, emulación, tiempo compartido, y otras"

[Susanta Nanda y Tzi-cker Chiueh, "A Survey on Virtualization Technologies", RPE Report, SUNY at Stony Brook, New York. Feb. 2005.]

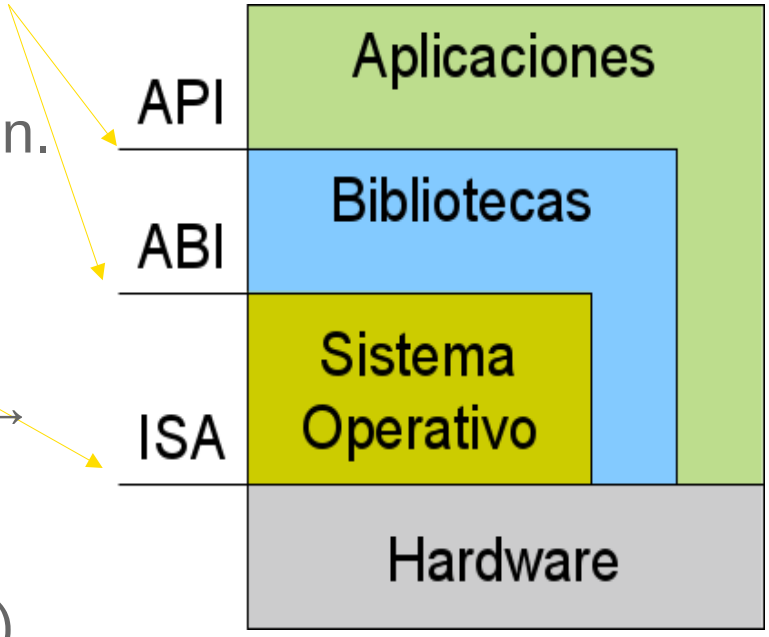
Máquina virtual

- Denominamos Máquina Virtual (MV) al software que simula una computadora y que puede ejecutar programas como si fuese una computadores “real”.
- Podemos verlo como un duplicado eficiente y aislado de una máquina real.
- Los procesos que ejecuta una MV estan limitados por los recursos y abstracciones que proporciona. Además, estos procesos están aislados del resto de procesos en otras Mvs.

Monolíticos: caracteristicas

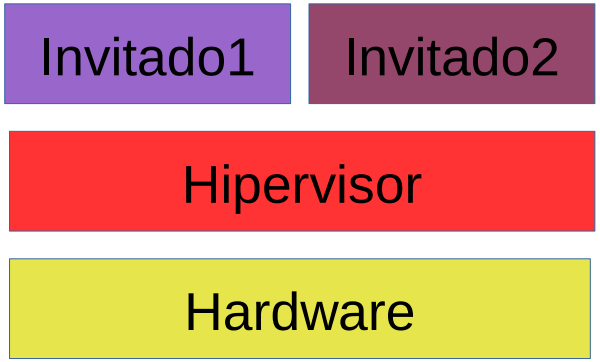
▷ Podemos clasificar las MVs:

- **Máquina virtual de procesos**:
  - Nivel biblioteca: Wine
  - Nivel Lenguaje programación. JVM, Microsoft .NET CLI
  - Nivel SO: Jail, Containers
- **Máquina virtual de sistemas**
  - Emulación: Convertir ISA1 → ISA2
  - Virtualización:
    - Hipervisor nativo (Tipo 1)
    - Hipervisor anfitrión (Tipo 2)



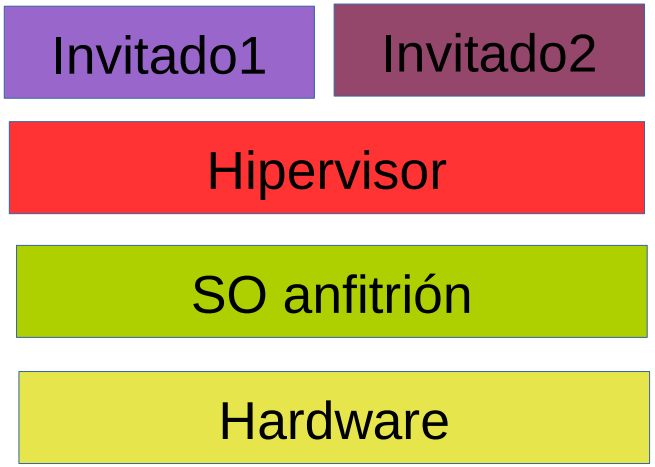
Tipos de hipervisores

- **Hipervisor Tipo 1 (*bare metal*):**



- Ejemplos:
  - Xen
  - Vmware ESX

- **Hipervisor Tipo 2 (*hosted*):**



- Ejemplos:
  - VMWare Workstation
  - KVM

Hipervisores: implementación

▷ Los SOs están construidos para ejecutarse en modo kernel. Ahora es el hipervisor el que se ejecuta en modo kernel ¿cómo se solventa? = ¿cómo manejo las instrucciones privilegiadas?

▷Según la implementación podemos separarlos en:

- *Traducción binaria* de código binario “crítico” del SO invitado (no de las aplicaciones).
- *Paravirtualización*: El código del SO invitado es recompilado para amoldarlo a la API anfitriona, eliminando la necesidad de que la MV atrape las instrucciones privilegiadas.
- *Virtualización asistida por hardware*: trata de resolver problemas del hardware relativos a la gestión de trampas. Ej. VT-x de Intel o SVM de AMD.

2.

Tipos de sistemas operativos

Qué características especiales tienen algunos SOs

Sistemas de tiempo compartido

▷ Soportan el uso “interactivo” del sistema:

- Cada usuario tiene la ilusión de disponer de la máquina completa.
- Tratan de optimizar el tiempo de respuesta.
- Basados en asignar fracciones de tiempo - se reparte el tiempo de CPU de forma equitativa entre los procesos.

▷ Permiten la participación activa de los usuarios en la edición, depuración de programas, y ejecución de los procesos.

▷Implementación: ¿cómo se haría?

La RSI de reloj se modifica para contabilizar el tick actual de reloj al proceso en ejecución, cuando `n.º ticks consumidos = n.º asignado → Planificar()`.

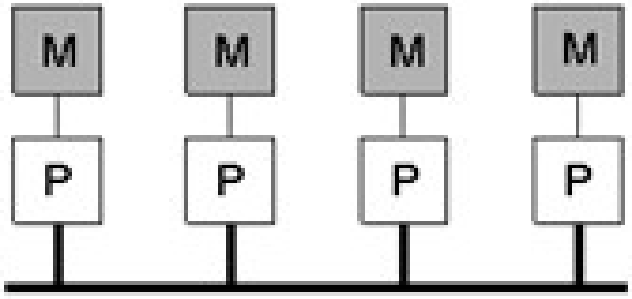
Sistemas de tiempo-real

- ▷ Un sistema de tiempo-real (RTS) es un sistema informático que reacciona con el entorno (responde a eventos físicos) a los que debe responder en un plazo determinado.
- ▷ Las tareas ejecutadas tienen asociado **tiempo límite** (*deadline*) en el que deben ejecutarse. Si no se satisface el *deadline* los resultados pueden ser inútiles e incluso catastróficos.
- ▷ Un RTS necesita el soporte de un **SO de tiempo-real** (RTOS), que debe garantizar la planificación de todas las tareas de forman que cumplan sus plazos.
- ▷ Prácticamente todos los SOs actuales incorporan alguno de los requisitos de tiempo-real.

Sistemas distribuidos

- Un **sistema distribuido** es un sistema multicomputador donde las diferentes CPUs no comparten memoria, ni reloj.
- El objetivo básico es compartir recursos (hard o soft).
- También permite un aumento de la fiabilidad del sistema: si una parte falla, el resto puede seguir con la ejecución.

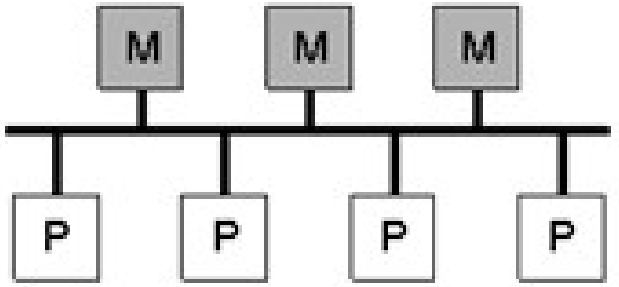
P = procesador
M = memoria
— = Red comm.



Sistemas paralelos

- SOs para **sistemas multiprocesadores** – sistemas de computador con varios procesadores que comparten memoria y un reloj.
- Sustentan aplicaciones paralelas cuyo objetivo es obtener aumento de velocidad computacional.
- **Multiprocesamiento Simétrico** (*SMP*) - todos los procesadores pueden ejecutar tanto código del SO como de las aplicaciones.

P = procesador
M = memoria
= bus



Wireless Information Devices

- Los dispositivos de información inalámbrica (WID) son sistemas de computador con escasos recursos y móviles.
- En la mayoría de los casos son necesarios sistemas operativos específicos que sepan gestionar y ejecutarse con pocos recursos, en especial la limitada potencia eléctrica, CPU y memoria.
- Podemos incluir aquí redes de sensores, IoT, etc.

3.

Soporte de Linux a la virtualización

Describiremos los elementos de los kernels actuales de Linux destinados a soportar virtualización

Soporte de Linux a la virtualización

> Linux soporta una virtualización ligera basada en dos mecanismos:

- **Espacios de nombres** (*namespaces*) – permite que se vean propiedades globales del sistema desde diferentes aspectos o “vistas”.
- **Grupos de control** (*cgroups*)- permite la gestión particionada de recursos asignables a tareas o grupos de tareas.

> Además, el kernel da soporte a MV como UML, XEN, KVM, VirtualBox, VMWare, Wine, etc.

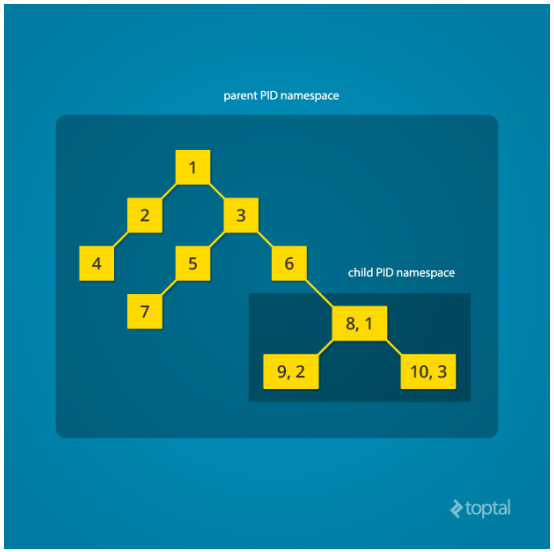
Namespaces

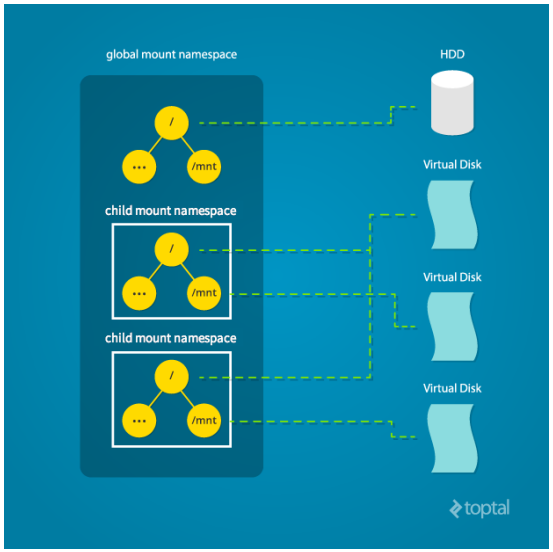
▷ Los espacios de nombre actualmente incluidos son:

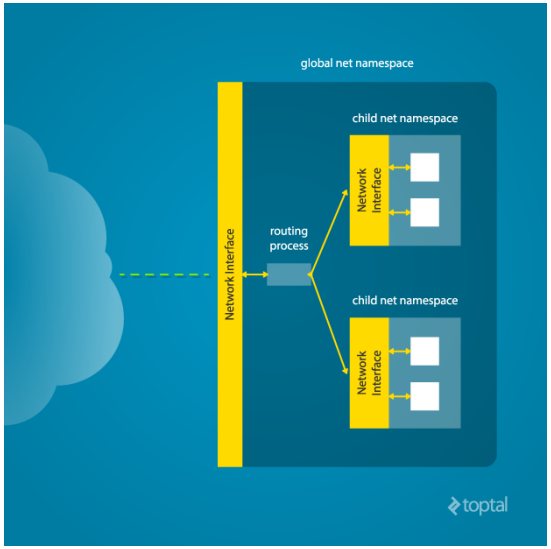
- *System V IPC* – mecanismos de comunicación entre procesos.
- *mounts* - sistemas de archivos montados.
- *UTS (Unix Timesharing System)* – información del sistema (nombre, versión, tipo arquitectura, etc.)
- *pid* – espacio de nombres de identificadores de procesos
- *network* – conjunto de dispositivos de red.
- *userid* – permite limitar los recursos por usuario

> Otros SOs que soportan tipos de virtualización similar a namespaces: Zones de Solaris o Jail de FreeBSD.

Namespaces: ejemplos







Fuente: https://www.toptal.com/linux/separation-anxiety-isolating-your-system-with-linux-namespaces

Namespaces: usos

▷ Posibles usos de los *namespaces*:

- Servidores Privados Virtuales (VPS) como por ejemplo Linux Containers (lxc.sourceforge.net)
- Application checkpoint and restart (ACS)
- En cluster:
  - Sustitución de NFS
  - Re-construcción de /proc

Grupos de control

> Controlan los recursos asignados a una tarea/grupo de tareas. Diferentes subsistemas (controladores de recursos):

- *cpu*: utilizado por el planificador para suministrar el acceso de las tareas de un cgroup a la CPU.
- *cpuset*: asigna CPUs individuales y memoria en multicores.
- *devices*: permite/deniega el acceso a un dispositivo.
- *memory*: limita el uso de memoria a tareas de un cgroup, y genera informes automáticos.
- *blkio*: establece los límites de accesos de E/S desde/hacia dispositivos de bloques (discos, USB,…)
- ns: subsistemas de espacios de nombres.

Grupos de control: uso

> Varias formas de uso:
  - Seudo-sistema de archivos
  - cgroupsfs
  - herramientas *libcgroup*
  - demonio *engine rules*

> Ejemplos, podemos:
  - Ver los grupos con `cat /proc/cgroups`
  - Ajustarlos a través de `/sys/fs/cgroup/`
  - `Algunas ordenes se construyen sobre cgroups: cpuset`

4.

Seminario: Linux Kernel Module

Cómo podemos adaptar la funcionalidad del kernel

Cuestiones a tratar

> ¿Que son los LKM?
  - ¿Qué estructura tienen?
  - ¿Cómo se construyen?
> ¿Qué órdenes suministra el sistema para manejarlos?
  - instalar/desinstalar
  - módulos disponibles
  - dependencias entre ellos
  - propiedades de un módulo
> Ventajas e inconvenientes de los módulos de carga dinámica

Seminario: organización

> **Objetivo**: Cada grupo de alumnos debe responder a las cuestiones que se han planteado concretadas en kernel de Linux superiores al 2.6, para lo cual ha de buscar y organiza material sobre el tema.
> **Material**: se elaborará un documento que se subirá a la plataforma hasta el 26 de octubre. Máximo de 5 páginas.
> **Celebración**: viernes 28 de octubre.
> **Dinámica**: en clase, cada grupo, a petición del profesor, responderá a una de las cuestiones planteadas. El resto de alumnos/profesor pueden plantear preguntas que deberán contestar.
> **Criterio de éxito**: cualquier miembro del grupo debe responder a las cuestiones realizadas por el resto de compañeros o el profesor.
> **Calificación**: Hasta 0,25 puntos.

5.

Trabajo grupal: SOs de producción

Descripción de un SO de producción

Trabajo en grupo

> **Objetivo**: Estudio de la estructura y principales componentes de un SO de producción (procesos, memoria, archivos y seguridad) a elegir por el grupo de entre: Windows, Android, iOS o Xen.
> **Fecha entrega**: hasta 11 de enero de 2017.
> **Tamaño máximo**: 1 pg portada, 1 pg índice y bibliografía, 10 paginas de contenido a doble cara (estructura del SO, gestión de procesos, memoria, archivos y seguridad).
> **Grupo**: 5 alumnos por grupo.
> **Criterio de éxito**: cualquier miembro del grupo debe demostrar, en resumen oral o escrito, que ha alcanzado el objetivo propuesto.
> **Calificación**: Hasta 1 punto (se utilizará rúbrica).