

Sistemas operativos.
Componentes. Estructura.
Funciones. Tipos.

TEMA 15

ABACUS NT

Oposiciones 2021

Índice

1. Introducción

2. Componentes y funciones

2.1. Funciones del sistema operativo

- 2.1.1. Como gestor de recursos:
- 2.1.2. Como máquina extendida
- 2.1.3. Como interfaz de usuario

3. Estructura

3.1. Sistemas operativos monolíticos

3.2. Sistemas por capa

3.3. Sistemas con microkernel

3.4. Sistemas cliente-servidor

3.5. Máquinas virtuales

3.6. Exokernel

4. Clasificación de los Sistemas Operativos

4.1. Según el número de usuarios

4.2. Según el número de procesos

4.3. Según el número de procesadores del sistema informático

4.4. Según el tiempo de respuesta

5. Historia de los Sistemas Operativos

6. Implementación de los Sistemas operativos actuales

6.1. Ejemplo de un Sistema Operativo actual: Android

- 6.1.1. Máquina virtual java
- 6.1.2. Gestión de Memoria en Android.
- 6.1.3. Gestión de Procesos en Android.
- 6.1.4. Planificación de procesos
- 6.1.5. Gestión de Usuarios en Android

7. Conclusión.

7.1. Relación del tema con el currículo

8. Bibliografía

1. Introducción

Un ordenador sin sistema operativo es un cacharro inútil. Más de un neófito con su recién adquirida máquina ha vuelto a la tienda a reclamar que “no hace nada” por no haber sido aún instalado.

Y es que la interacción máquina-usuario se produce precisamente mediante la interfaz del sistema. El sistema operativo controla todos los recursos de la computadora y proporciona la base sobre la cual pueden escribirse los programas de aplicación.

Una de las principales funciones del sistema operativo es ocultar toda esta complejidad y proporcionar al programador y al usuario un conjunto más conveniente de instrucciones y programas con el cual trabajar.

Por encima del sistema operativo está el resto del software de sistema. Aquí se encuentran el intérprete de comandos (shell), compiladores, editores y programas de aplicación independientes similares a ellos.

La principal función de los sistemas operativos (SSOO) según el autor **William Stallings**, en su libro Sistemas operativos (2005), es “**controlar la ejecución de programas y aplicaciones y actuar como interfaz entre las aplicaciones y el hardware del computador**”.

En este tema vamos a ver cómo realizan su trabajo los sistemas operativos, entendiendo los tipos que existen y los componentes con los que cuentan.

2. Componentes y funciones

Un sistema operativo es el software encargado de ejercer el control y coordinar el uso del hardware entre los programas y los usuarios. Conceptualmente está formado por tres capas:

- **Núcleo**: interacciona directamente con el **hardware**, ocultando sus detalles y ofreciendo una vía sencilla y flexible de acceso al mismo.
- **Servicios del sistema/Máquina extendida**: ofrecen una interfaz de programa (API) a los programas para que se puedan abstraer de aspectos concretos del hardware.
- **Interfaz**: a través de la cual el usuario puede dialogar de forma interactiva con el ordenador.

2.1. Funciones del sistema operativo

2.1.1. Como gestor de recursos:

- **Gestión de procesos**: Se encargan de la creación, planificación y destrucción de procesos.
- **Gestión de memoria**: Se encargan principalmente de asignar y liberar memoria para los procesos.

- **Seguridad y protección:** Identifica únicamente a los usuarios y establece lo que pueden hacer con los recursos del sistema.
- **Comunicación y sincronización entre procesos:** Ofrecen mecanismos para que los procesos puedan comunicarse y sincronizarse.
- **Contabilidad y monitorización:** Mantiene un conjunto de estadísticas de utilización de los diversos recursos del sistema.

2.1.2. Como máquina extendida

- **Gestión de la E/S:** Facilita el manejo de los dispositivos periféricos.
- **Gestión de archivos y directorios:** Permiten trabajar con archivos y directorios, y la administración del almacenamiento externo.
- **Detección y respuesta a errores:** Realiza el tratamiento adecuado en caso de errores internos y externos de hardware, errores de memoria, mal funcionamiento de dispositivos, etc
- **Control de acceso al sistema:** En accesos compartidos, proporciona protección a los recursos y datos ante usuarios no autorizados y resolución de conflictos de propiedad

2.1.3. Como interfaz de usuario

- Ofrece una interfaz de usuario en espera de órdenes, que se encarga de analizar y ejecutar a través de los servicios disponibles del sistema. Esta interacción puede ser a través de la interfaz gráfica, de comandos escritos o utilizando un asistente de voz.

3. Estructura

Cada SO estructura los componentes anteriores según un **enfoque de diseño diferente**. Así, el autor Andrews S. Tanenbaum, en su libro “Sistemas operativos modernos” (2009), presenta los siguientes seis tipos de diseños:

3.1. Sistemas operativos monolíticos

En este diseño, todo el SO se ejecuta como un único programa en **modo núcleo**. Este tipo de diseño tiene como dificultad principal que cualquier cambio implica la **recopilación** del núcleo y el reinicio del sistema (para paliar este problema, GNU/Linux permite la carga dinámica de **módulos ejecutables**).

3.2. Sistemas por capa

Estos SSOO se organizan como una jerarquía de capas, en donde cada capa ofrece una interfaz a la capa superior, y utiliza los servicios que le ofrece la capa inferior.

3.3. Sistemas con microkernel

Este diseño se basa en la división del SO en módulos muy pequeños y bien definidos, donde sólo uno de ellos se ejecuta en modo núcleo (el microkernel o micro-núcleo).

3.4. Sistemas cliente-servidor

En este diseño se diferencian dos clases de procesos, los servidores, los cuales proporcionan servicios, y los clientes, que los utilizan. La comunicación entre los clientes y servidores se realiza mediante el paso de mensajes.

3.5. Máquinas virtuales

En este diseño existe un módulo principal llamado hipervisor que presenta al exterior varias máquinas virtuales, cada una de las cuales es capaz de ejecutar un SO distinto. Si el hipervisor actúa directamente sobre el hardware es de tipo 1, si lo hace con un SO anfitrión, es de tipo 2. VMware Workstation es un ejemplo de hipervisor de tipo 2.

3.6. Exokernel

Con objeto de mejorar el anterior diseño, en este diseño los recursos hardware se partitionan, y es un programa llamado exokernel, que se ejecuta en modo núcleo, el que se encarga de asignar los recursos a las máquinas virtuales, y garantizar que son utilizados correctamente. Como ejemplo tenemos Mirage OS.

4. Clasificación de los Sistemas Operativos

Para hacer una clasificación de los Sistemas Operativos según su tipo, hay que tener en cuenta una serie de parámetros:

- Número de usuarios
- Número de procesos
- Número de procesadores
- Tiempo de respuesta

4.1. Segundo el número de usuarios

- **Monousuario.**

Sólo un usuario trabaja con un ordenador. En este sistema todos los dispositivos de hardware están a disposición de dicho usuario y no pueden ser utilizados por otros hasta que éste no finalice su sesión.

Los SO monousuario más claros son: **ChromeOS, Android o MS-DOS.**

- **Multiusuario.**

En este sistema, varios usuarios pueden utilizar simultáneamente los recursos del sistema. Pueden compartir, sobre todo, los dispositivos externos de almacenamiento y los periféricos de salida, fundamentalmente impresoras.

Ejemplos de sistemas multiusuario son Linux, Novell, todos los sistemas Windows desde la aparición de Windows NT, (Windows XP, 7, 8 y 10) etc.

4.2. Segundo criterio de clasificación: Según el número de procesos

Esta clasificación se hace atendiendo al número de programas o procesos que puede realizar simultáneamente el ordenador o sistema informáticos:

- **Monoprogramación** o monotarea.

En este caso, el sistema solamente puede ejecutar un programa a la vez. De esta forma, los recursos del sistema estarán dedicados al programa hasta que finalice su ejecución.

Esto no impide que el sistema pueda ser multiusuario; es decir, varios usuarios pueden intentar ejecutar sus programas en el mismo ordenador, pero de forma sucesiva. Para ello, se tienen que establecer las correspondientes colas o prioridades en la ejecución de los trabajos.

En este sistema la atención del procesador estará dedicada a un solo programa hasta que finalice.

Un claro ejemplo de sistema monoprogramación es el MS-DOS.

En este caso un programa utilizará parte de todos los procesadores. Si llega otro nuevo programa para ser ejecutado, se utilizarán también todos los procesadores, y así hasta su total utilización. De esta forma trabajarán, todos, pero es evidente que lo harán a bajo rendimiento.

- **Multiprogramación** o multitarea.

Con estos sistemas se pueden ejecutar varios programas o procesos concurrentemente. Para ello la CPU compartirá el tiempo de uso del procesador entre los diferentes programas que se van a ejecutar.

Así, todos los procesos tardarán individualmente más tiempo en ejecutarse; pero, comparándolo con la monoprogramación, el tiempo medio de espera será mucho menor.

Como ejemplos de sistema en multiprogramación podemos hablar de Windows en cualquier versión desde Windows NT o XP, de Linux, Android o en general de cualquier sistema operativo actual.

4.3. Tercer criterio de clasificación: Según el número de procesadores del sistema informático

Esta clasificación atiende a que el sistema u ordenador cuente con uno o varios procesadores para realizar los procesos:

- **Monoproceso**. En este caso, el ordenador consta de un único procesador.

Todos los trabajos pasarán por él.

Ejemplo de SO monoproceso es MS-DOS.

- **Multiproceso.**

El ordenador cuenta con varios procesadores. Estos procesadores pueden actuar de dos formas diferentes:

- **Multiproceso simétrico (SMP):**

El sistema utilizará la totalidad de los procesadores para realizar todas las tareas.

- **Multiproceso asimétrico (AMP):**

Existen ordenadores que irán saturando de trabajo a sus procesadores poco a poco. Con la primera tarea utilizará el primer procesador; si entra otra tarea, se utilizará lo que reste de potencia del primer procesador y lo necesario del segundo. Los demás procesadores se irán utilizando de forma sucesiva. De esta forma pueden quedar procesadores inactivos.

Son SO multiproceso (según el ordenador y teniendo en cuenta otras muchas condiciones) los da la familia Windows Server, Windows XP; Vista y 7, Windows 8 y 10, y muchas versiones de UNIX/Linux y Novell.

4.4. Segundo el tiempo de respuesta

Esta clasificación se hace teniendo en cuenta el tiempo que tarda el sistema en obtener los resultados después de lanzar un programa a ejecución:

- **Tiempo real.** La respuesta es inmediata (o casi inmediata) tras lanzar un proceso.
- **Tiempo compartido.** Cada proceso utilizará fracciones de tiempo de ejecución de la CPU hasta que finalice. En este caso, parece que el usuario dedica la CPU exclusivamente para él; pero esto no es cierto, ya que, aunque el usuario no lo perciba, la CPU está dedicada a varios procesos a la vez. Todos los SO multiusuario ofrecen a los usuarios un tiempo de respuesta compartido.

5. Historia de los Sistemas Operativos

Vamos a analizar los diferentes tipos que han ido surgiendo, tomando de referencia las 4 generaciones de SSOO.

Primera etapa (1943-1955) y Segunda etapa (1956 – 1963)

- **Sin SO:** En la primera etapa los sistemas carecían de SO.
- **SO por lotes:** En la segunda etapa, aparecen los primeros SSOO, los cuales permiten agrupar los trabajos en **lotes**.

Tercera etapa (1963-1979)

- **SO multitarea:** Este tipo de SO permiten ejecutar varios procesos concurrentemente.
- **SO de tiempo compartido:** Permiten que varios usuarios puedan trabajar de forma interactiva desde diferentes terminales.
- **SO en tiempo real:** Son SSOO, los cuales deben responder a unos requisitos temporales estrictos. Como ejemplo tenemos QNX.
- **SO de propósito general:** Son sistemas capaces de operar en lotes, en multitarea, en tiempo real, en tiempo compartido y en modo multiprocesador (el SO gestiona una arquitectura que tiene varios procesadores).

Cuarta etapa (1980 - actualidad)

- **SO orientado al usuario final:** Este tipo de SSOO se encuentran más enfocados a mejorar el rendimiento del trabajo del usuario que el rendimiento de la máquina. Como ejemplo tenemos Windows.
- **SO distribuido:** Son SSOO los cuales se ejecutan en varias máquinas a la vez, de tal modo que el usuario no tiene que saber dónde se está ejecutando la aplicación que está usando. Como ejemplo podemos citar Amoeba.
- **SO con middleware:** Son hipervisores de tipo 2 que se encargan de gestionar sistemas distribuidos. Uno de los middlewares más importantes en esta etapa ha sido CORBA.

6. Implementación de los Sistemas operativos actuales

Los SSOO más relevantes actualmente son:

- **Windows:** Creado por Microsoft. Posee un núcleo híbrido, en el que se encuentra, entre otros componentes, la capa de abstracción de hardware (hal.dll), y el micronúcleo (ntoskrnl.exe).
- **GNU/Linux:** Surgió en 1992 cuando Linus Torvalds creó un núcleo de un SO basado en Unix. Posteriormente, el proyecto se unió al proyecto GNU. Tiene un diseño monolítico, y es un SO multitarea, multiusuario, y multiprocesador.
- **Android:** Es un SO creado por Google basado en el núcleo de Linux, y ampliamente utilizados en dispositivos móviles. Las capas principales que encontramos en su arquitectura son: Kernel Linux, librerías, marco de trabajo de las aplicaciones, runtime de Android y las aplicaciones.

6.1. Ejemplo de un Sistema Operativo actual: Android

Android es un sistema operativo monousuario desarrollado por Google, basado en el Kernel 2.5 de Linux y otro software de código abierto. Está orientado a la plataforma ARM aunque existen versiones para x86 (Chrome OS).

Android es software libre bajo licencia Apache y actualmente (año 2020) es el sistema operativo móvil más utilizado en el mundo.

Su nombre proviene de la famosa novela de Philip K. Dick *Do Androids Dream of Electric Sheep?* (*¿Sueñan los androides con ovejas eléctricas?*) que fue adaptada en la película *Blade Runner* de Ridley Scott.

La compañía que lo desarrolló fue Android Inc, apoyada económicamente por Google y comprada por este en 2005, pero no fue presentado hasta el año 2007 tras la fundación del Open Handset Alliance que engloba numerosos fabricantes de hardware relacionados con la telefonía móvil.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan sobre un Middleware que las abstrae de la complejidad de las comunicaciones del sistema y de la máquina en particular, mediante un framework Java.

Bajo esta máquina virtual Java, se encuentra el núcleo del sistema escrito en lenguaje C y que incorpora una potente biblioteca que incluye un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una API gráfica OpenGL, un motor de renderizado WebKit (Safari), un motor gráfico SGL - SSL y una biblioteca estándar de C (Bionic).

6.1.1. Máquina virtual java

Uno de los elementos clave de Android es la máquina virtual Java. En lugar de utilizar una tradicional máquina virtual Java (VM), tales como Java ME (Java Mobile Edition), Android utiliza su propia máquina virtual personalizada diseñada para asegurar que la multitarea se ejecuta de manera eficiente en un único dispositivo.

Inicialmente se utilizó la máquina virtual Dalvik con la librería de ejecución JIT (Just In Time) que traducía a bytecode Java cualquier aplicación en el momento de su ejecución.

En la versión 4.4 de Android se introdujo la máquina ART (Android RunTime) que compila la aplicación a Java bytecode en el momento de su instalación; a partir de la versión 5 ART reemplazó por completo a Dalvik.

Las aplicaciones (Apps) en Android se distribuyen por tanto en archivos .APK (Android Application PackAge, similar al JAR del Java de Oracle) y son compiladas en el momento de su instalación por ART a Java bytecode.

La máquina virtual Java de Android no gestiona directamente la memoria o los procesos, sino que utiliza el núcleo de Linux subyacente para manejar la planificación de procesos y la gestión de la memoria.

Android funciona en modo multitarea ejecutando múltiples instancias de la máquina virtual java, una por cada tarea, lo que crea un entorno muy confiable, similar a un sandbox.

6.1.2. Gestión de Memoria en Android.

La plataforma de Android se basa en la premisa de que la memoria disponible es una pérdida de memoria, de modo que intenta utilizar toda la memoria disponible en todo momento.

Android Runtime (ART) y la máquina virtual Dalvik usan las funciones de paginación y mapeo de memoria (mmapping) para administrar la memoria.

Jerarquía de Memoria

La jerarquía de Memoria en Android se basa en la gestión de tres niveles: RAM, zRAM y ROM.

- La memoria RAM es la memoria principal del dispositivo y se gestiona mediante paginación.
- La zona libre de RAM se utiliza como una memoria de intercambio SWAP en la que se almacenan datos y programas de forma comprimida, aumentando así su aprovechamiento en término de capacidad, de ahí el nombre de zRAM, para diferenciarla.
- La memoria ROM está compuesta por dispositivos extremadamente lentos compuestos por memoria EEPROM Flash Integrada y opcionalmente una tarjeta SD.
- Existe un nivel cero en esta jerarquía compuesto por la memoria caché y registros del microprocesador, pero Android no los gestiona directamente.

Paginación de la memoria.

La memoria RAM está dividida en páginas. Por lo general, cada página tiene 4 KB de memoria.

Las páginas se consideran **libres o usadas**. Las páginas libres son memoria RAM sin usar. Las páginas usadas son memoria RAM que el sistema está utilizando de manera activa y se agrupan en las siguientes categorías:

- Almacenamiento en caché: Memoria respaldada por un archivo del almacenamiento (por ejemplo, código o archivos asignados a la memoria). Hay dos tipos de memoria caché:
 - Privada: Es propiedad de un proceso y no se comparte.
 - Compartida: Es utilizada por varios procesos.
- Anónima: Memoria no respaldada por un archivo en almacenamiento (por ejemplo, asignada por mmap() con la marca MAP_ANONYMOUS).

Las páginas a su vez pueden ser **Limpias o Sucias**: Las páginas limpias contienen una copia exacta de un archivo (o parte de un archivo) que existe en el almacenamiento. Una página limpia se convierte en una página sucia cuando ya no contiene una copia exacta del archivo (por ejemplo, del resultado de la operación de una app).

Es posible borrar las páginas limpias porque siempre se pueden volver a generar con los datos del almacenamiento. En cambio, no se pueden borrar las páginas sucias, ya que se perderían los datos.

Algoritmo de Reemplazo de página.

El algoritmo de reemplazo de página es el mismo que utiliza el núcleo Linux 2.6 y que se denomina “Page Frame Reclamation Algorithm”. Es una modificación del algoritmo LRU pero basado en listas y con prioridad para los procesos activos.

Las páginas de memoria en uso se dividen en dos listas por cada zona de memoria, una para aplicaciones en uso y otra para aplicaciones inactivas. Las páginas de memoria de la lista de inactivas pueden además ser marcadas como swappables o descartables dependiendo del tiempo de inactividad y previsión de uso.

El algoritmo PFRA Sólo se ejecuta cuando la memoria empieza a escasear, momento en el cual selecciona todas las páginas marcadas como “descartables/limpias” y a continuación se seleccionan las páginas menos usadas recientemente (LRU) de la lista de aplicaciones inactivas.

Este algoritmo se ejecuta de forma incremental, intentando liberar el máximo de memoria pero cargando en zRAM/swap páginas inactivas sucias, volcando a memoria masiva (limpiando) solo en las últimas iteraciones si no se alcanza liberar la memoria deseada.

La implementación en Android de este algoritmo la realiza el demonio kswapd que se ejecutará automáticamente al alcanzarse un umbral mínimo de memoria disponible y dejará de utilizarse cuando se alcance un umbral máximo determinado también de antemano.

Si tras la ejecución de kswapd aún no hay suficiente memoria para la carga de una determinada aplicación en memoria, se ejecutará un asistente de limpieza (LMK) que sacará de memoria procesos activos, según un nivel de prioridad asignado, comenzando con los procesos en segundo plano menos usados, y siguiendo incluso con la app de inicio, servicios, apps en primer plano, procesos del sistema, etc.

6.1.3. Gestión de Procesos en Android.

Procesos:

Con frecuencia la documentación de Android se refiere a la gestión de procesos como **ciclo de vida** de Android.

Android implementa la **multitarea** mediante la ejecución de **múltiples hilos** para cada proceso; el proceso más simple consta de un sólo hilo principal que sólo es lanzado en el caso de que no exista ninguna instancia del proceso en ejecución.

De igual forma un proceso puede utilizar múltiples hilos, uno por cada subproceso. También se da la situación en que un proceso comparte subprocesos con otros, lanzando para ello el sistema los hilos necesarios.

Tipos de procesos en Android.

El tipo de proceso en Android determina el comportamiento del planificador respecto a este: no es lo mismo un proceso “activo” y “visible” cuya finalización por parte del sistema tendría una **experiencia muy negativa** para el usuario, que por ejemplo un servicio del sistema.

Android, por tanto, clasifica los procesos como:

- **Activos:** Se están ejecutando y además son requeridos para la interacción con el usuario, bien sean procesos en primer plano o servicios dando soporte a éstos.
- **Visibles:** Un nivel por debajo de Activo, ya que un proceso visible estaría en segundo plano.
- **Servicio:** Procesos que no son visibles de forma directa, pero dan un servicio importante al usuario, por ejemplo, mantener una conexión a Internet. Estos procesos no deben ser finalizados a no ser que su mantenimiento suponga una merma para un proceso activo.
- **Segundo Plano:** Procesos en ejecución, pero no visibles, a los que el sistema intentará dotar de recursos siempre que haya disponibles.
- **Vacíos:** Es una forma rápida de inicialización: por ejemplo, un proceso que intenta ser lanzado pero el liberador de memoria está ocupado haciéndole hueco.

6.1.4. Planificación de procesos

La planificación en Android es **expropiativa** lo cual debe ser tenido en cuenta por el programador, ya que si no se definen bien los diferentes componentes de la aplicación esta puede ser finalizada mientras está efectuando una tarea relevante para el sistema o para el usuario.

El algoritmo utilizado por el planificador es **Round-Robin con prioridad** basada en una **jerarquía de importancia** según el tipo de proceso (visto anteriormente) y otros factores:

El sistema mantiene una lista pseudo-LRU (Last Recently Used) para evitar que los procesos se apropien de los recursos, pero ordenada según prioridades. La prioridad de un proceso se puede incrementar si este es un subproceso de mayor prioridad, es decir, la prioridad de un conjunto de procesos interdependientes se iguala a la prioridad del componente más prioritario del conjunto.

Cuando un proceso pasa al estado de espera o de finalización, este se almacena en caché (zRAM). El sentido de almacenar también los procesos finalizados es acelerar su carga en RAM si se vuelven a solicitar.

Android asegura la respuesta de cualquier app lanzada por el usuario, deteniendo y matando a los procesos que impiden la fluidez y liberando recursos para las aplicaciones más prioritarias.

6.1.5. Gestión de Usuarios en Android

Android sólo puede ser accedido por un usuario en un momento dado, es decir por el usuario principal.

En algunos sistemas (Samsung, Xiaomi) es posible mantener usuarios adicionales mediante la adición de capas de software sobre el sistema operativo que emule el uso por varios usuarios, por ejemplo, para control parental o para tener un doble uso del teléfono en casa y en el trabajo.

Sin embargo, en el sistema sólo hay en realidad un único usuario, al que se le añaden o restringen permisos y visores sobre aplicaciones.

Este usuario al que se le ligan las distintas cuentas de proveedores (Google, Xiaomi, Amazon, etc.) viene con los permisos restringidos por defecto.

Si es necesario dar mayores niveles de prioridad al usuario debemos acceder a la configuración del teléfono para activar los permisos completos. A este proceso se le suele llamar ruteo del teléfono, porque al hacerlo activamos los permisos completos del usuario root en el kernel Linux. Con frecuencia esta escalada de privilegios está restringida y a apareja la pérdida de la garantía y otras ventajas de la marca.

7. Conclusión.

No se puede entender la evolución de la informática sin tener en cuenta también la evolución de los sistemas operativos. Las interfaces actuales van mucho más allá de las interfaces gráficas y comienzan a interactuar con nosotros mediante asistentes de voz, como es el caso de SIRI, Cortana o Google Assistance.

Sin embargo, la estructura y funciones del sistema operativo afortunadamente no están sujetas a cambios tan vertiginosos, y una vez entendidas, podemos comprender perfectamente como son internamente los sistemas operativos.

En los próximos temas se desarrollan minuciosamente todas las funciones del sistema.

7.1. Relación del tema con el currículo

Este tema es aplicado en el aula en los módulos profesionales siguientes, con las atribuciones docentes indicadas (PES/SAI):

Grado Medio

- Sistemas operativos monopuesto (SMR) (PES/SAI)

Grado Superior

- Sistemas informáticos (DAM / DAW) (PES/SAI)
- Implementación de sistemas operativos (ASIR) (PES/SAI)

8. Bibliografía

- De Anasagasti, Miguel. "Fundamentos de la Computadora" 9^aed 2004 Edt. Paraninfo
- Patterson D.A. y Hennessy JL. "Estructura y diseño de computadoras: la interfaz hardware/software" 4^a Ed. (2005) Edt McGraw-Hill
- Prieto A, Lloris A, Torres JC. "Introducción a la Informática" 4^aed. (2006) Edt. McGraw-Hill
- Stallings W. "Organización y Arquitectura de Computadoras" (2006) 5^a Ed. Edt. Prentice-Hall
- Ramos A, Ramos MJ y Viñas S "Montaje y Mantenimiento de Equipos" (2012). Edt McGraw-Hill
- Jiménez Cembreras, Isabel M^a "Sistemas Informáticos" 2^aEd (2018) Edt. Garceta