Oposiciones cuerpo de secundaria. **Esquemas dos páginas sobre temario oposición profesorado Secundaria.**

**Especialidad informática**

short line

Autor: Sergi García Barea

Actualizado Mayo 2025

Licencia

**Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA)**: No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Índice

[**Introducción 2**](#_qgf6wc5twfuu)

[**Para el buen docente 2**](#_okzkg9mr3qa0)

[**¿Para qué prueba están adaptados estos esquemas? 2**](#_pzhjb2xpxluc)

[**Tema 1: Representación y comunicación de la información 3**](#_1t9rvy6xkpj2)

[**Tema 2: Elementos funcionales de un ordenador digital. Arquitectura. 5**](#_5jf768jmziq8)

[**Tema 20: Explotación y administración de sistemas operativos monousuario y multiusuario 7**](#_tzlwgjdokcvk)

[**Tema 72: Seguridad en sistemas de red: Servicios, protecciones, estándares avanzados 9**](#_9oxqu1aopsc4)

[**Tema 74: Sistemas multimedia 11**](#_q416mrtecr9g)

# Introducción

Este documento recoge una serie de **esquemas sintéticos del temario oficial para las oposiciones al cuerpo de profesorado de Secundaria, especialidad Informática**, con el objetivo de ofrecer una herramienta de estudio clara, útil y eficaz. Cada esquema está diseñado para ocupar como máximo **cuatro páginas**, facilitando así su consulta rápida, comprensión global y memorización eficaz.

# Para el buen docente

Pero estos esquemas **no son solo para superar una oposición**. Están pensados para ayudarnos a **ser mejores docentes**, personas que entienden la complejidad técnica de su materia, pero también su dimensión educativa, social y ética. Ser docente es una tarea de gran responsabilidad que trasciende un examen: **enseñamos a través de lo que sabemos, pero también a través de lo que somos**.

**Por eso, si has llegado hasta aquí, te pido algo importante: lleva contigo el compromiso de ser un buen docente más allá de la oposición.** Utiliza estos materiales como base, sí, pero hazlos crecer con tu experiencia, tus reflexiones y tu vocación. Que enseñar sea una decisión consciente, diaria, y no un trámite. Que lo que prepares hoy, lo apliques con compromiso durante toda tu carrera docente, pensando siempre en lo mejor para tu alumnado.

# ¿Para qué prueba están adaptados estos esquemas?

Estos esquemas están específicamente adaptados para la **prueba de exposición oral del procedimiento selectivo regulado por la ORDEN 1/2025, de 28 de enero**, de la Conselleria de Educación, Cultura, Universidades y Empleo de la Comunitat Valenciana, que establece lo siguiente:

*"La exposición tendrá dos partes: la primera versará sobre los aspectos científicos del tema; en la segunda se deberá hacer referencia a la relación del tema con el currículum oficial actualmente vigente en el presente curso escolar en la Comunitat Valenciana, y desarrollará un aspecto didáctico de este aplicado a un determinado nivel previamente establecido por la persona aspirante. Finalizada la exposición, el tribunal podrá realizar un debate con la persona candidata sobre el contenido de su intervención.”*

No obstante, estos materiales pueden ser también útiles para preparar **otras modalidades de oposición** (como ingreso por estabilización o pruebas de adquisición de especialidades), así como para otras especialidades cercanas, especialmente **la de Sistemas y Aplicaciones Informáticas**, ya que comparten gran parte del temario técnico

# 

# 

# 

# 

# Tema 1: Representación y comunicación de la información

**1.1 Introducción**

* Definición: transformación de fenómenos del mundo real en estructuras digitales binarias.
* Fundamento para todo procesamiento informático: desde el código hasta el hardware.

**1.2 Sistemas de Numeración**

* Base, dígitos, sistema posicional.
* Sistemas: binario, octal, hexadecimal y decimal.
* Conversión entre sistemas para debugging, direccionamiento y arquitectura.

**1.3 Representación de Datos**

* **Enteros:**
  + Códigos: signo+magnitud, CA1 y CA2 (uso de CA2 por su simplicidad en hardware).
* **Punto flotante (IEEE 754):**
  + Precisión simple (32 bits) y doble (64 bits), compuestas por signo (1 bit), exponente (8/11 bits) y mantisa (23/52 bits).
  + Se normaliza desplazando el punto binario para que quede en la forma 1.xxxxx, permitiendo maximizar la precisión y garantizar una representación única.
* **Texto:**
  + ASCII (7 bits, limitado), 8 bits extendido y Unicode (UTF-8/16/32) para múltiples idiomas y emoji.
* **Imágenes:**
  + Representación como matriz de píxeles con RGB + canal alfa.
  + Formatos: BMP, PNG (sin pérdida), JPEG (con pérdida).
* **Vídeo:**
  + Secuencia de imágenes y audio. Compresión intraframe e interframe.
  + Códecs: H.264, H.265 (HEVC), AV1 (compresión espacial e interframe).
* **Audio:**
  + Muestreo (44,1 kHz CD / 48 kHz vídeo); cuantificación (16/24 bits).
  + Formatos: WAV/FLAC (sin pérdida), MP3/AAC (con pérdida).

**1.4 Lógica y Operaciones Binarias**

* Aritmética: suma, resta, multiplicación, división en base 2.
* Lógica digital: puertas AND, OR, NOT, XOR.
* Aplicaciones: ALU, circuitos combinacionales.

**1.5 Detección y Corrección de Errores**

* Bit de paridad (detección simple).
* CRC (Cyclic Redundancy Check).
* Código de Hamming (corrige 1 bit; MEM ECC).
* Reed‑Solomon (múltiples errores; CDs, RAID, QR).

**1.6 Representación en Big Data y Nube**

* Formatos JSON/BSON, Avro, Protobuf, Parquet, ORC.
* Integración en pipelines cloud, microservicios y análisis masivo (Spark, Hadoop).

**1.7 Comunicación Digital**

* Modelo Shannon‑Weaver: emisor, codificador, canal (ruido), decodificador receptor + (Compresión y cifrado)
* Señales digitales vs analógicas.
* Protocolos: TCP/IP, UDP/IP, Ethernet, WebRTC.

**1.8 Seguridad**

* Hashing: SHA‑256 (integridad), bcrypt/Argon2 (contraseñas).
* Cifrado:
  + Simétrico: AES.
  + Asimétrico: RSA, ECC.
  + Cifrado simétrico pero compartiendo la clave con cifrado Asimétrico (SSL)
  + Aplicaciones: HTTPS, VPN, BitLocker, TLS.

**1.9 Compresión**

* Sin pérdida: Huffman, LZW (ZIP, PNG).
* Con pérdida: JPEG, MP3, H.264 (optimizadas para multimedia).

**1.10 Conclusión**

* La correcta representación y manipulación de datos es básica en informática.
* Aplica desde circuitos y sistemas embebidos hasta servicios cloud y Big Data.

**2. PARTE DIDÁCTICA - 1.º DAM – Módulo: Programación  
 🗂 Unidad didáctica: “Estructuras repetitivas aplicadas a la representación y tratamiento de información”**

**2.1 Contextualización didáctica**

* Nivel: 1.º de Ciclo Formativo de Grado Superior – Desarrollo de Aplicaciones Multiplataforma (DAM)
* Módulo profesional: Programación
* Currículo aplicable: Real Decreto 450/2010 y Decreto 48/2011 (Comunitat Valenciana)
* **Justificación:** Las estructuras repetitivas permiten automatizar tareas, manipular información y modelar procesos fundamentales en programación profesional.

**2.2 Objetivos de aprendizaje**

* Aplicar bucles (for, while, do-while) en la codificación de algoritmos.
* Manipular datos numéricos y textuales mediante estructuras de control repetitivas.
* Simular procesos de codificación, verificación y transmisión de datos.

**2.3 Metodología y principios pedagógicos**

* Metodología activa: basada en tareas prácticas y resolución de problemas reales.
* Progresión: ejercicios con dificultad creciente y trabajo individual seguido de refactorización colaborativa.
* Recursos utilizados: IDE Java (NetBeans, VS Code), pseudocódigo, diagramas de flujo, vídeos explicativos.
* Estrategias metodológicas: trabajo por parejas con roles diferenciados, flipped classroom, revisión entre iguales.

**2.4 Inclusión y atención a la diversidad (niveles III y IV):**

* Código base parcial con comentarios orientativos.
* Retroalimentación individualizada durante el proceso.

**2.5 Aplicación del Diseño Universal para el Aprendizaje (DUA):**

* Múltiples formas de representación (textual, visual, audiovisual).
* Variedad de formas de expresión del aprendizaje (código funcional, exposición oral, demos).
* Participación equitativa mediante ajustes de complejidad y agrupaciones heterogéneas.

**2.6 Actividad principal: “Procesando datos binarios con bucles”**

* Conversión manual de números decimales a binario, octal y hexadecimal mediante bucles.
* Codificación de texto carácter a carácter en binario utilizando la tabla ASCII.
* Cálculo de bit de paridad mediante conteo de unos en cadenas binarias.
* Simulación de un canal de transmisión con errores aleatorios y aplicación del Código de Hamming.
* Verificación de integridad de cadenas binarias utilizando un algoritmo de hash simplificado (XOR).

**2.7 Evaluación**

* Criterios de evaluación: uso correcto y eficiente de bucles, lógica de control adecuada, limpieza del código, comprensión de los procesos implicados.
* Instrumentos de evaluación: rúbricas detalladas, revisión entre compañeros, evaluación continua con entregas parciales.
* Resultados esperados: desarrollo de programas funcionales que evidencien el dominio de las estructuras repetitivas aplicadas a tareas reales de representación y transmisión de datos.

**2.8 Conclusión didáctica**

Esta unidad permite al alumnado integrar conocimientos fundamentales de programación y aplicarlos en situaciones prácticas. Fomenta el pensamiento algorítmico, el desarrollo de habilidades técnicas y competencias transversales, incrementando la motivación y la autonomía. Además, establece conexiones claras entre la teoría informática y su aplicación profesional, desarrollando un aprendizaje significativo.

# Tema 2: Elementos funcionales de un ordenador digital. Arquitectura.

#### **1. INTRODUCCIÓN**

#### **2. ELEMENTOS FUNCIONALES**

### **2.1. Unidad Central de Proceso (CPU - Central Processing Unit)** Encargada de ejecutar instrucciones del programa.

### **Registros**: PC, IR, MAR, MDR, FLAGS.

* **ALU / FPU**: operaciones lógicas y en coma flotante.
* **Unidad de Control**: cableada (rápida) o microprogramada (flexible).

### **2.2. Memoria principal** Memoria de acceso rápido que almacena temporalmente datos e instrucciones.

### **RAM (Random Access Memory)**:

### **DRAM (Dynamic RAM)**: económica, necesita refresco constante.

### **SRAM (Static RAM)**: más rápida, usada en cachés.

### **Jerarquía de memoria**: estructura escalonada que optimiza acceso:

### Registros > Caché (L1, L2, L3) > RAM > SSD/HDD.

### Afecta directamente al **rendimiento**: menor latencia en niveles superiores.

### 

### **2.3. Subsistema de Entrada/Salida (E/S)** Permite la interacción del procesador con dispositivos externos.

### **Dispositivos periféricos**: teclado, ratón, impresora, disco, red.

### **Modos de transferencia**:

### **Polling**: la CPU consulta activamente si hay datos disponibles.

### **Interrupciones**: el periférico avisa al procesador cuando necesita atención.

### **DMA (Direct Memory Access)**: transfiere datos directamente sin CPU.

### **2.4. Sistema de buses** Canales físicos que interconectan los componentes del sistema.

### **Tipos**: **Bus de datos** (transmite información)**, bus de direcciones** (localiza posiciones de memoria) y **bus de control** (gestiona operaciones como lectura o interrupciones).

### **Temporización**: puede ser **síncrona** (con reloj compartido) o **asíncrona** (mediante señales independientes, más flexible).

#### **3. MODELOS DE ARQUITECTURA**

### **3.1. Von Neumann**

### Memoria compartida para instrucciones y datos.

### Problema: **cuello de botella** en el acceso a memoria.

### **3.2. Harvard**

### Memoria separada para datos e instrucciones.

### Permite acceso paralelo, más eficiente.

#### **4. TAXONOMÍA DE FLYNN**

### Clasificación de arquitecturas según número de flujos de instrucciones y datos:

### **SISD (Single Instruction, Single Data)**: tradicional, una instrucción opera sobre un dato.

### **SIMD (Single Instruction, Multiple Data)**: una instrucción actúa sobre múltiples datos (p. ej., GPU).

### **MISD (Multiple Instruction, Single Data)**: redundante, escasa utilidad práctica.

### **MIMD (Multiple Instruction, Multiple Data)**: múltiples procesadores ejecutan múltiples instrucciones, típico en sistemas multinúcleo.

#### **5. MEMORIAS: TIPOS Y EVOLUCIÓN**

### **ROM (Read-Only Memory)**: no volátil, incluye BIOS/UEFI.

### **EEPROM**: puede reprogramarse eléctricamente.

### **Flash**: memoria no volátil usada en SSD y dispositivos móviles.

### **Tendencias actuales**:

### **HBM (High Bandwidth Memory)**: gran ancho de banda, muy cercana al procesador.

### **GDDR6**: memoria gráfica usada en GPUs.

### **Optane**: tecnología de Intel basada en memoria persistente de alta velocidad.

### **SoC (System on Chip)**: integración total en un único chip, común en móviles.

#### **6. CICLO DE INSTRUCCIÓN**

### Etapas secuenciales que sigue la CPU para ejecutar una instrucción:

### **Fetch**: se lee la instrucción desde memoria.

### **Decode**: se interpreta la instrucción.

### **Execute**: se realiza la operación.

### **Memory**: acceso a memoria si es necesario.

### **Write-back**: los resultados se guardan.

### **Técnicas de optimización**:

### **Pipeline**: ejecución en paralelo de etapas.

### **Superescalaridad**: ejecución de múltiples instrucciones por ciclo.

### **Out-of-Order Execution**: reordenamiento dinámico para mejorar rendimiento.

### **SMT (Simultaneous Multithreading)**: varios hilos por núcleo (ej. Hyper-Threading de Intel).

### **Multicore**: varios núcleos físicos en un chip.

#### 

#### **7. TENDENCIAS FUTURAS**

### **Computación cuántica**: uso de **qubits**, permite paralelismo masivo y algoritmos no clásicos.

### **Arquitecturas neuromórficas**: diseñadas para imitar el cerebro humano.

### **Aceleradores de IA**:

### **TPU (Tensor Processing Unit)**: de Google, optimizadas para redes neuronales.

### **NPU (Neural Processing Unit)**: en dispositivos móviles.

### **FPGAs (Field-Programmable Gate Arrays)**: configurables post-fabricación.

### **Sistemas heterogéneos**: combinación de CPU, GPU y otros aceleradores especializados.

### **APLICACIÓN DIDÁCTICA (CFGS DAM – Módulo “Programación de procesos y servicios”)**

#### 

#### **1. REQUISITOS PREVIOS**

#### **2. OBJETIVOS DE APRENDIZAJE**

* Analizar el impacto de la arquitectura del sistema en la ejecución de servicios.
* Programar de forma eficiente teniendo en cuenta núcleos, concurrencia y jerarquía de memoria.

#### 

#### **3. METODOLOGÍA**

* ABR (Aprendizaje Basado en Retos). Simulación de entornos reales.
* Uso de herramientas de análisis: htop, perf, taskset, systemd.

#### 

#### **4. ATENCIÓN A LA DIVERSIDAD (Niveles III y IV)**.

#### 

#### **5. DISEÑO UNIVERSAL PARA EL APRENDIZAJE (DUA)**

* **Representación:** diagramas de arquitectura, vídeos explicativos.
* **Expresión:** scripts, paneles, presentaciones.
* **Compromiso:** retos prácticos contextualizados.

#### 

#### **6. ACTIVIDAD PRINCIPAL**

**“Optimizando procesos según la arquitectura del sistema”** *Proyecto práctico por equipos con Python.*

**Fases:**

1. **Análisis del sistema**: detección de arquitectura (núcleos, RAM) con os, platform, psutil.
2. **Programación concurrente**:
   * Con multiprocessing y threading.
   * Afinidad a núcleos con os.sched\_setaffinity().
3. **Medición de rendimiento**:
   * Scripts instrumentados con time, tracemalloc.
   * Análisis con perf, htop, comparativa de configuraciones.
4. **Automatización**:
   * Scripts como demonios con systemd.
   * Monitorización de CPU, registro en log.
5. **Defensa y entrega**: Informe técnico + exposición oral con resultados y gráficos.

# 

# Tema 3: Componentes, estructura y funcionamiento de la Unidad Central de Proceso (CPU)

### **1. Introducción**

* La CPU es el núcleo funcional del ordenador: ejecuta instrucciones, coordina operaciones y gestiona recursos.
* Evolución histórica:
  + Mononúcleo: Intel 8086, primeros Pentium.
  + Multinúcleo: Core 2 Duo, AMD Ryzen.
  + Híbridas: Intel Alder Lake, Apple M1/M2 (big.LITTLE).
* Tendencias actuales:
  + Instrucciones específicas para IA (DL Boost, Neural Engine).
  + Integración CPU+GPU.
  + Alta eficiencia energética: clave en móviles y servidores.

### **2. Estructura interna de la CPU**

#### **2.1 Unidad Aritmético-Lógica (ALU)**

* Ejecuta operaciones matemáticas, lógicas y de comparación.
* Registros asociados: acumulador, operandos, flags.
* Unidades especializadas:
  + FPU (coma flotante).
  + SIMD: AVX, SSE, NEON.

#### **2.2 Unidad de Control (UC)**

* Decodifica instrucciones y genera señales de control.
* Tipos:
  + Cableada: más rápida.
  + Microprogramada: más flexible.
* Técnicas modernas:
  + Pipelining.
  + Ejecución fuera de orden y especulativa.
  + Predicción de saltos con IA.

#### **2.3 Memoria interna**

##### **Registros**

* Ultrarrápidos y limitados.
* Generales y especiales: PC, IR, FLAGS, MAR/MDR.

##### **Caché**

* L1: núcleo.
* L2: núcleo o compartida.
* L3: compartida global.
* Técnicas: prefetching, coherencia, reemplazo adaptativo.

##### **RAM**

* Área de trabajo externa.
* DDR5, LPDDR5X según entorno.

#### **2.4 Buses internos**

* Datos, direcciones, control.
* Evolución: FSB → QPI, Infinity Fabric, Unified Memory.

### **3. Funcionamiento de la CPU**

#### **3.1 Conjunto de instrucciones**

##### **CISC**

* Instrucciones complejas.
* Arquitecturas: x86, ARMv8-A.

##### **RISC**

* Instrucciones simples, eficientes.
* Ejemplos: ARM, RISC-V.

##### **Extensiones modernas**

* AVX-512, VT-x/AMD-V, AMX.
* IA integrada, virtualización nativa.
* RISC-V: abierta, modular, en crecimiento.

#### **3.2 Ciclo de instrucción**

* Fases: Fetch → Decode → Execute → Memory Access → Write-back.
* Optimización:
  + Multithreading: Hyper-Threading, SMT.
  + Predicción, ejecución especulativa.
  + Soporte IA en la CPU (Apple Neural Engine, Intel AMX).

### **4. Conclusión**

Las CPU modernas integran paralelismo, vectores, control avanzado, y capacidades IA. Su comprensión es clave para programar sistemas eficientes, optimizar procesos y entender el funcionamiento base de cualquier equipo digital.

## **PROPUESTA DIDÁCTICA: “SIMULANDO UNA CPU: PROGRAMACIÓN DE UN INTÉRPRETE DE INSTRUCCIONES”**

### **A. Contextualización**

* Nivel: 1.º FP Grado Superior DAM o DAW.
* Módulo: Programación.
* Perfil: alumnado con dominio básico de estructuras de control y memoria.

### **B. Objetivos**

* Simular mediante programación el ciclo de instrucción de una CPU.
* Representar digitalmente registros, memoria y operaciones básicas.
* Comprender cómo la CPU gestiona y ejecuta código.

### **C. Metodología**

* Aprendizaje basado en proyectos y resolución de problemas.
* Desarrollo incremental con pruebas y visualización.
* Programación individual o en parejas (Python, Java, C++).

### **D. Actividad principal**

* Crear un simulador básico de CPU que:
  + Interprete un pequeño conjunto de instrucciones (LOAD, ADD, JMP…).
  + Emule registros como PC, AC, IR, FLAGS.
  + Visualice el ciclo completo por consola o GUI.
  + Opcional: interrupciones, subrutinas, multithreading.

### **E. Atención a la diversidad**

* Nivel III: código base, guías, plantillas con instrucciones comentadas.
* Nivel IV: objetivos reducidos, apoyo continuo, evaluación formativa.

### **F. DUA**

* Representación: consola paso a paso, GUI opcional, esquemas.
* Acción/expresión: elección libre de lenguaje y estructura.
* Implicación: retos progresivos, gamificación por funcionalidades.

### **G. Evaluación**

* Rúbricas: ejecución correcta, estructura clara, rigor técnico.
* Instrumentos: revisión de código, presentación oral, demo funcional.

### **H. Conclusión didáctica**

* Programar una CPU permite entender su lógica interna desde el rol de programador.
* Favorece el pensamiento lógico, la abstracción computacional y la transferencia de conocimientos entre hardware y software.

# 

# Tema 4: Memoria interna: Tipos, Direccionamiento, Características y funciones

**1. Introducción**

* La memoria es fundamental para el rendimiento del sistema: almacena instrucciones y datos, afecta a la velocidad de ejecución y coordina con la CPU.
* Una jerarquía eficiente de memoria evita cuellos de botella y maximiza el rendimiento.

**2. Conceptos fundamentales**

**2.1 Elementos clave**

* Soporte físico: silicio (RAM, Flash), magnético (HDD), óptico (CD/DVD).
* Acceso: aleatorio (RAM), secuencial (cintas), asociativo (caché).
* Volatilidad: volátil (RAM), no volátil (Flash, ROM, HDD).

**2.2 Direccionamiento**

* 2D: decodificador único, memorias pequeñas.
* 3D: múltiples decodificadores, alto rendimiento.

**2.3 Características**

* Velocidad: latencia y ancho de banda.
* Unidad de transferencia: palabra, bloque, línea.
* Modos de direccionamiento lógico: directo, indirecto, paginado, segmentado.

**3. Tipos de memoria**

**3.1 Volátiles**

* SRAM: rápida, cara, sin refresco.
* DRAM: necesita refresco, más densa.
* DDR, GDDR, HBM: sincronizadas, especializadas para GPU o IA.

**3.2 No volátiles**

* ROM: solo lectura.
* Flash: base de SSD.
* NVRAM: combina velocidad y persistencia.

**4. Jerarquía de memoria**

* Registros: máxima velocidad, mínima capacidad.
* Caché (L1–L3): latencia reducida.
* RAM: datos activos.
* Almacenamiento (SSD, HDD): persistencia.
* Red/Nube: acceso remoto y respaldo.

**5. Conexión CPU–Memoria**

**5.1 Estructura**

* SRAM: biestables.
* DRAM: condensador.
* ROM: direccionamiento fijo.

**5.2 Acceso**

* Buses: direcciones, datos, control.
* Modos: lectura, escritura, modificación.
* Técnicas: paginación, acceso por columnas, refresco.

**6. Mejora de rendimiento**

**6.1 Memoria caché**

* L1–L3 según núcleo o CPU completa.
* Mapeo: directo, asociativo, por conjuntos.
* Reemplazo: LRU, FIFO, aleatorio.

**6.2 Memoria virtual**

* **MMU** (Unidad de Gestión de Memoria) convierte direcciones virtuales en físicas mediante
  + **Paginación**: divide la memoria en bloques fijos (páginas y marcos), lo que permite asignación no contigua y evita la fragmentación externa.
  + **Segmentación**: organiza la memoria en bloques lógicos (código, datos, pila), respetando la estructura del programa pero con riesgo de fragmentación externa.
  + **Segmentación** paginada: combina ambos modelos dividiendo cada segmento lógico en páginas, optimizando espacio y manteniendo organización lógica.
* **TLB**: caché de traducciones recientes rápida.

**7. Tecnologías modernas**

* PMEM: persistente como Intel Optane.
* Memoria 3D: mayor densidad, menor latencia.
* PIM: procesamiento en el chip de memoria (IA, HPC).

**PROPUESTA DIDÁCTICA: “SIMULADOR DE JERARQUÍA DE MEMORIA: PROGRAMANDO ACCESOS, LATENCIAS Y CACHÉ”**

**A. Contextualización**

* Nivel: 2.º DAM o DAW.
* Módulo: Programación o Sistemas Informáticos.
* Perfil: alumnado con nociones de estructuras de datos y control de flujo.

**B. Objetivos**

* Simular la jerarquía de memorias desde registros hasta almacenamiento.
* Comprender cómo la latencia y las políticas de caché afectan al rendimiento.
* Programar comportamientos reales de sistemas modernos de memoria.

**C. Metodología**

* Proyecto práctico por parejas o grupos pequeños.
* Enfoque incremental: fases de diseño, implementación, testeo y presentación.
* Lenguajes posibles: Python, Java o C++.

**D. Actividad principal**

* Desarrollo de un programa que simule:
  + Memorias con distinta latencia y capacidad (registros, caché, RAM, disco).
  + Políticas de reemplazo de caché (LRU, FIFO, aleatorio).
  + Mapeo directo y asociativo por conjuntos.
  + Acceso secuencial y aleatorio a datos simulados
  + Estadísticas: tasa de aciertos/fallos, tiempo medio de acceso.
* Interfaz: consola o simple GUI para ver operaciones y resultados.
* Ampliación opcional: paginación, uso de TLB y acceso virtual a disco.

**E. Atención a la diversidad**

* Nivel III: código base preconfigurado, ayuda estructurada.
* Nivel IV: simulaciones más básicas con componentes seleccionados.

**F. DUA**

* Representación: animaciones, diagramas, consola paso a paso.
* Acción: desarrollo libre o guiado, estilos de programación variados.
* Implicación: simulación con ejemplos reales, feedback inmediato.

**G. Evaluación**

* Rúbricas: precisión técnica, eficiencia del simulador, claridad del código.
* Instrumentos: demo, documentación del diseño, revisión por pares.

**H. Conclusión didáctica**

* Simular memoria refuerza la comprensión del rendimiento real del software.
* El alumnado conecta teoría, arquitectura y programación, desarrollando visión de optimización y eficiencia.

# 

# Tema 10: Representación interna de los datos

### **1. Introducción**

* Toda información digital se representa internamente en binario (base 2).
* También se utilizan otras bases para facilitar tareas específicas:
  + Octal (8): sintaxis compacta.
  + Decimal (10): interfaz humana.
  + Hexadecimal (16): dirección de memoria, colores, instrucciones.
* Comprender estas representaciones es esencial en programación, redes y sistemas.

### **2. Representación de caracteres**

* **ASCII**: estándar de 7 u 8 bits.
* **UNICODE (UTF-8, UTF-16)**: codifica miles de símbolos, compatible con la web.
* Conversión texto ↔ binario esencial en programación, bases de datos y redes.

### **3. Representación de booleanos**

* 1 bit: 0 = falso, 1 = verdadero.
* Usos en condiciones, lógica digital, estructuras de control.
* Aplicación en puertas lógicas y simplificación con mapas de Karnaugh.

### **4. Representación de números enteros**

* **Signo y magnitud**, **CA1**, **CA2 (complemento a 2)**: estándar en programación.
* **Exceso-Z**: usado en representación de exponentes (coma flotante).

### **5. Representación de reales**

* **Coma fija**: poco precisa.
* **Coma flotante (IEEE 754)**: 32/64/128 bits.
* Componentes: signo, exponente, mantisa.
* Problemas comunes: redondeo, desbordamientos.

### **6. Números complejos**

* Dos flotantes: parte real + imaginaria. Representado con estructura, objeto, etc.
* Usos: audio, señales, simulación física, computación cuántica.

### **7. Representación de estructuras**

#### **7.1 Lineales**

* Vectores, matrices, listas enlazadas: estructuras fundamentales.

#### **7.2 Jerárquicas y grafos**

* Árboles: Árboles: BST (búsqueda binaria), AVL (balanceo estricto), B+ (claves en hojas, ideal para BBDD), R-B (rojo-negro, balanceo por colores).
* Grafos: listas/matrices de adyacencia.
* Aplicaciones en rutas, grafos sociales, IA.

#### **7.3 Tablas hash y punteros**

* Acceso O(1), gestión dinámica de memoria.
* Punteros: manipulación directa de direcciones (C/C++).

### **8. Multimedia y datos complejos**

#### **Imagen**

* Raster vs. vectorial.
* Compresión con y sin pérdida (JPEG, PNG).

#### **Sonido y vídeo**

* Muestreo, resolución, formatos (MP3, FLAC, MP4, H.265).
* Codificación por frames (compresión intraframe e interframe), códecs para streaming.

#### **3D**

* Modelado por vértices, formatos: OBJ, GLTF.
* Aplicaciones: videojuegos, simulación física, realidad aumentada.

### **9. Seguridad y compresión**

#### **Cifrado**

* Simétrico, asimétrico, combinación de ambas
* AES, RSA, ECC.
* Criptografía post-cuántica en desarrollo.

#### **Compresión**

* ZIP, PNG (sin pérdida).
* MP3, JPEG (con pérdida).

#### **Hash**

* SHA, MD5.
* Aplicaciones: autenticación, integridad, búsqueda.

## **PROPUESTA DIDÁCTICA: “CREA TU CONVERSOR BINARIO MULTITIPO: EL ORDENADOR DESDE DENTRO”**

### **A. Contextualización**

* Nivel: 1.º FP DAM o DAW.
* Módulo: Programación.
* Perfil: alumnado con competencias en codificación, estructuras de datos y desarrollo de interfaces.

### **B. Objetivos**

* Simular mediante programación la conversión binaria de diversos tipos de datos.
* Visualizar estructuras internas y codificación real.
* Integrar teoría de representación con desarrollo de software funcional.

### **C. Metodología**

* Aprendizaje basado en proyectos.
* Desarrollo individual o en parejas.
* Iteración por funcionalidades, pruebas y presentación.

### **D. Actividad principal**

**Proyecto: Programa “BinarioTotal”**

* Desarrollo de una aplicación que permita:
  + **Codificar/decodificar**:
    - Caracteres (ASCII, UTF-8).
    - Enteros (CA2).
    - Reales (IEEE 754).
    - Booleanos.
  + **Visualizar**:
    - Codificación interna en binario y hexadecimal.
    - Comparación entre formatos.
  + **Simular estructuras**:
    - Arrays, listas enlazadas, árboles (con impresión en memoria).
  + **Extra** (opcional):
    - Codificar imágenes o sonidos simples.
    - Comprimir o cifrar una cadena o archivo.
* Lenguajes recomendados: Python (Tkinter), Java (Swing/FX), C++ (CLI/GUI simple).
* Interfaz: consola o ventana gráfica básica.

### **E. Atención a la diversidad**

* Nivel III: plantillas base, guía paso a paso.
* Nivel IV: estructura modular, evaluación progresiva, refuerzo individual.

### **F. DUA**

* Representación: visualización binaria, esquemas gráficos, interfaces.
* Expresión: código, documentación, presentación del proyecto.
* Implicación: simulador personalizado, trabajo en equipo, reto final.

### **G. Evaluación**

* Rúbricas: exactitud técnica, calidad del código, visualización, documentación.
* Instrumentos: pruebas funcionales, demo, defensa oral.

### **H. Conclusión didáctica**

* El alumnado transforma la abstracción binaria en lógica aplicada.
* Se potencia la comprensión de la arquitectura digital desde el desarrollo de software

# 

# Tema 20: Explotación y administración de sistemas operativos monousuario y multiusuario

### **1. PARTE CIENTÍFICA**

#### **1.1. Introducción**

El sistema operativo (SO) es el software fundamental que permite al usuario interactuar con el hardware.

#### **1.2. Clasificación de sistemas operativos**

* **Por número de usuarios:**
  + *Monousuario:* un usuario activo por vez. Ej.: Windows Home, macOS.
  + *Multiusuario:* múltiples sesiones simultáneas. Ej.: Linux, Windows Server.
* **Por tareas:**
  + *Monotarea:* obsoleto. *Multitarea:* alternancia (concurrente) o simultaneidad real (paralela).
* **Por núcleo (kernel):**
  + *Monolítico:* alto rendimiento, difícil de mantener (Linux).
  + *Microkernel:* modular, más seguro (Minix, QNX).
  + *Híbrido:* rendimiento y seguridad (Windows NT, macOS).
* **Por arquitectura:**
  + *Sistemas en red:* comparten recursos (FreeBSD, Windows Server).
  + *Distribuidos:* múltiples máquinas como un único sistema lógico (Hadoop, Mesos).
  + *Cloud:* escalabilidad y ejecución bajo demanda (AWS, Azure).
* **Por procesadores:**
  + *SMP (Symmetric Multiprocessing):* núcleos comparten memoria.
  + *NUMA (Non-Uniform Memory Access):* varias CPU cada una su propia memoria..
* **RTOS (tiempo real):** QNX, VxWorks.
* **Emergentes:** SO para IoT (Zephyr, RIOT).

#### **1.3. Explotación de sistemas monousuario**

* Instalación del SO, controladores y software inicial.
* Gestión de cuentas locales y configuración básica.
* Actualizaciones automáticas y medidas de seguridad sencillas.
* Ej.: Windows 11 Home, macOS (Time Machine, Gatekeeper, FileVault).

#### **1.4. Explotaicón y administración de sistemas multiusuario**

* Los procesos se ejecutan en modo usuario, accediendo a recursos del sistema mediante llamadas al sistema (syscalls), que transicionan temporalmente al modo kernel para seguridad y estabilidad.
* **Procesos:** múltiples procesos por usuario, planificación (FIFO, RR, prioridades), IPC (pipes, sockets).
* **Memoria:** paginación, segmentación, swapping, espacio separado usuario/kernel.
* **Servicios:**
  + Linux: systemd, daemons, cron.
  + Windows: Task Scheduler, servicios en segundo plano.
* **Almacenamiento:**
  + Sistemas de archivos: NTFS (Windows), EXT4, Btrfs (copy-on-write, snapshots), ZFS (integridad de datos, replicación). Volúmenes: LVM, Storage Spaces.
* **Gestión avanzada:**
  + Linux: sudo, ACLs, journalctl, cgroups.
  + Windows Server: Active Directory, GPOs, RDP, administración remota.

#### **1.5. Virtualización y contenedores**

* **Virtualización completa:** VMs con su propio SO. Ej.: VirtualBox, VMware.
* **Paravirtualización:** uso eficiente del hardware. Ej.: Xen, KVM.
* **Virtualización ligera (contenedores):** comparten kernel del post. Ej.: Docker, LXC. Aislamiento basado en namespaces y cgroups (CPU, RAM, red). Seguridad con AppArmor.
* **Orquestación:**
  + Kubernetes: escalado, balanceo, pods.
  + Helm: gestión de paquetes en Kubernetes.
* **Comparativa:** Aislamiento: VM total, contenedor parcial. Recursos: VM más consumo, contenedor más eficiente.Arranque: VM lento, contenedor rápido.

#### **1.6. Seguridad y monitorización**

* **Seguridad:**
  + Control de acceso mediante DAC (Discretionary), MAC (Mandatory) y RBAC (Role-Based), uso de ACLs, autenticación multifactor
  + Cifrado: BitLocker, LUKS, ZFS.
  + Firewalls: iptables, nftables, Windows Defender.
* **Monitorización:**
  + Linux: htop, Prometheus, Grafana.
  + Windows: Sysinternals, Event Viewer.
* Automatización mediante cron, scripts bash/powershell o herramientas de configuración como Ansible y despliegue remoto con Salt Project.

### **2. PARTE DIDÁCTICA**

#### **2.1. Contextualización**

* **Etapa educativa:** Ciclo Formativo de Grado Superior en Administración de Sistemas Informáticos en Red (ASIR). **Módulo profesional:** *Servicios en Red*.

#### **2.2. Objetivos de aprendizaje**

* Comprender la estructura y funciones de sistemas operativos.
* Distinguir entre entornos monousuario y multiusuario.
* Aplicar procedimientos de instalación, configuración, monitorización y seguridad.
* Desplegar servicios mediante máquinas virtuales y contenedores.

#### **2.3. Metodología**

* **Aprendizaje basado en proyectos (ABP):** desarrollo de una red empresarial simulada con dos escenarios operativos.
* **Técnicas didácticas:** simulación de incidencias, resolución por comandos, checklist técnico.
* **Entornos reales:** VirtualBox, Docker, Ubuntu Server, Windows Server.

#### **2.4. Atención a la diversidad y DUA**

* **Nivel III:** entornos preconfigurados, uso de scripts automatizados, apoyo visual.
* **Nivel IV:** videotutoriales, interfaz gráfica de administración, guías paso a paso.
* **DUA:** contenidos accesibles por múltiples vías (CLI, GUI, tutoriales), respuestas prácticas y escritas, retroalimentación inmediata.

#### **2.5. Actividad principal: *“Administra tu sistema”***

* **Objetivo:** configurar y comparar un entorno monousuario y otro multiusuario.
  + Instalación de Windows/macOS (monousuario) y Ubuntu Server (multiusuario).
  + Creación de usuarios, servicios, permisos. Despliegue de contenedor Docker con servicio web.
  + Aplicación de políticas de seguridad y uso de herramientas de monitorización.
* **Producto final:** sistema funcional y documentación técnica comparativa.

# 

# Tema 72: Seguridad en sistemas de red: Servicios, protecciones, estándares avanzados

### **1. Fundamentos de la seguridad en red**

1.1. Importancia de la seguridad en red

* Las redes son vectores de ataque constantes en entornos conectados.
* Riesgos: pérdida de disponibilidad, integridad y confidencialidad.

1.2. Necesidad de protección

* La ciberseguridad comienza en la red: proteger el canal de datos es esencial.

### **2. Servicios de seguridad**

2.1. Autenticación y autorización

* Métodos: contraseñas, autenticación multifactor (MFA), biometría.
* Protocolos: Kerberos, OAuth2, SSO (Single Sign-On).

2.2. Control de acceso

* Modelos: RBAC (control basado en roles), ABAC (basado en atributos).
* Tecnologías: VLANs, NAC (Network Access Control), listas de control de acceso (ACL).

2.3. Cifrado y no repudio

* Herramientas: HTTPS, VPN, cifrado de discos, firmas digitales.

2.4. Auditoría y SIEM

* SIEM: gestión centralizada de eventos e información de seguridad.
* Herramientas: Wazuh, Splunk; detección de anomalías mediante análisis en tiempo real.

### **3. Técnicas de protección**

3.1. Segmentación de red

* Uso de VLANs, microsegmentación y redes definidas por software (SDN).

3.2. Bastionado (hardening)

* Eliminación de servicios innecesarios, refuerzo de configuraciones, automatización con Ansible.

3.3. Prevención de amenazas

* Herramientas: EDR (Endpoint Detection and Response), DNSSEC, bloqueo de direcciones IP.

### **4. Defensa en profundidad**

4.1. Firewalls de nueva generación (NGFW)

* Inspección profunda de paquetes, filtrado por aplicación, bloqueo en tiempo real.

4.2. Sistemas IDS e IPS

* IDS: detección de intrusiones. IPS: prevención activa.

4.3. Copias de seguridad

* Regla 3-2-1: 3 copias, en 2 soportes diferentes, 1 externa.

### **5. Normativa y estándares**

5.1. Estándares técnicos

* ISO 27001, NIST SP 800-53, COBIT, MITRE ATT&CK.

5.2. Legislación vigente

* RGPD, LOPDGDD, ENS (Esquema Nacional de Seguridad), Directiva NIS2.

5.3. Evaluación de riesgos

* MAGERIT y PILAR: análisis detallado de activos, amenazas y vulnerabilidades.

### **6. Amenazas actuales**

* Man-in-the-Middle (MITM): interceptación si el tráfico no está cifrado.
* Ransomware: secuestro de datos mediante cifrado.
* Fallos de configuración en entornos cloud.
* Ataques DDoS: saturación de servicios mediante tráfico masivo.

### **7. Concienciación y formación**

* El usuario como primera línea de defensa.
* Simulacros y campañas de concienciación: phishing, ransomware, ingeniería social.
* Actividades de ciberseguridad: CyberCamp, CTFs, test de impacto.

## **PROPUESTA DIDÁCTICA: “CIBERDEFENSORES EN RED”**

### **A. Contextualización**

* Nivel educativo: 1.º FP Grado Superior en Administración de Sistemas Informáticos en Red (ASIR).
* Módulo: Seguridad y alta disponibilidad.

### **B. Objetivos de aprendizaje**

* Aplicar medidas de protección de red y respuesta a incidentes.
* Identificar amenazas y aplicar estándares y buenas prácticas.
* Fomentar la responsabilidad digital y el trabajo colaborativo.

### **C. Metodología**

* Aprendizaje basado en proyectos (ABP) y gamificación.
* Dinámica de roles: analista SIEM, responsable de red, backup, hardening.
* Aprendizaje activo mediante retos progresivos.

### **D. Actividad principal**

* Simulación de un SOC (Security Operations Center).
* Herramientas: TryHackMe, VirtualBox, Packet Tracer.
* Cada equipo diseña y defiende su infraestructura ante ataques simulados.
* Evaluación continua del rendimiento técnico y organizativo.

### **E. Atención a la diversidad (niveles III y IV)**

* Nivel III: apoyo visual, guías paso a paso, grupos heterogéneos.
* Nivel IV: adaptación de tareas, refuerzo individual, recursos accesibles.

### **F. Diseño Universal para el Aprendizaje (DUA)**

* Representación: vídeos, esquemas, simulaciones.
* Acción y expresión: elección de herramientas, roles diferenciados.
* Implicación: enfoque competitivo, trabajo por equipos, retroalimentación constante.

### **G. Evaluación**

* Rúbricas por competencias técnicas y actitudinales.
* Instrumentos: observación directa, diarios de aprendizaje, checklist de configuración.
* Criterios: efectividad defensiva, trabajo en equipo, resolución de incidentes.

### **H. Conclusión didáctica**

* La ciberseguridad es una competencia transversal y crítica.
* Simular un SOC permite integrar teoría, práctica y conciencia ética.
* El alumnado se convierte en protagonista de su aprendizaje, desarrollando competencias digitales avanzadas.

# Tema 74: Sistemas multimedia

### **1. Definición y contexto**

1.1. ¿Qué es un sistema multimedia?

* Conjunto de tecnologías que integran texto, imagen, audio, vídeo, animación y datos en tiempo real.
* Aplicaciones: educación (pizarras digitales), medicina (imagen diagnóstica), control remoto (drones), entretenimiento (videojuegos, RA).
* Incorporación de inteligencia artificial: reconocimiento facial, generación de voz e imagen.

### **2. Representación digital de medios**

2.1. Imagen

* Formatos RAW, BMP, sin compresión.
* Raster: JPEG, PNG (píxeles, pierden calidad al escalar).
* Vectorial: SVG (formas matemáticas, calidad escalable).
* Canal alfa: gestión de transparencia.

2.2. Audio

* Tasa de muestreo: frecuencia de captura (ej. 44.1 kHz).
* Bitrate: calidad versus tamaño.
* Formatos: WAV (sin compresión), FLAC (sin pérdida), MP3/AAC (con pérdida).

2.3. Vídeo

* FPS: fluidez (30 fps estándar, 60 fps mayor realismo).
* Códec: compresión (H.264); contenedor: empaquetado (MP4).

### **3. Procesamiento multimedia**

3.1. Transformadas

* Fourier: análisis de frecuencias (audio).
* DCT: base de JPEG.
* Wavelets: compresión multiescala (JPEG2000).

3.2. Convoluciones

* Aplicación de filtros a imágenes.
* Base de redes neuronales convolucionales (visión artificial).

### **4. Transmisión multimedia**

* Protocolos adaptativos: HLS, DASH (ajuste de calidad).
* Protocolos en tiempo real: RTMP, RTSP (baja latencia).

### **5. Inteligencia Artificial en multimedia**

5.1. Generación

* DALL·E, Stable Diffusion, voice cloning, NeRF.

5.2. Análisis

* YOLO, DETR (detección objetos).
* CLIP, GPT-4V (relación imagen-texto).

### **6. Herramientas**

* FFmpeg: conversión, edición por línea de comandos.
* OpenCV: visión artificial.
* MediaPipe: detección de gestos en móviles.

### **7. Tendencias futuras**

* Codificación neural, vídeo volumétrico, edge computing, interfaces adaptativas.

### **8. Ética y legislación**

* Deepfakes y manipulación audiovisual.
* Sesgos algorítmicos.
* IA Act (UE): marco legal según nivel de riesgo.

### **9. Conclusión**

* Los sistemas multimedia evolucionan hacia la comprensión y generación inteligentes de contenido.
* Su diseño debe equilibrar eficiencia técnica, ética y usabilidad.

## **PROPUESTA DIDÁCTICA: “ENTRENADOR MULTIMEDIA: CREA UNA APP DE FITNESS INTERACTIVO”**

### **A. Contextualización**

* Nivel educativo: 2.º curso de Grado Superior en DAM.
* Módulo: Multimedia y dispositivos móviles.

### **B. Objetivos de aprendizaje**

* Integrar medios audiovisuales en apps Android.
* Optimizar la compresión y la reproducción multimedia.
* Diseñar experiencias interactivas y accesibles.

### **C. Metodología**

* Aprendizaje basado en proyectos (ABP).
* Trabajo en equipo con división de roles técnicos.

### **D. Actividad principal**

* Proyecto: APP Android de entrenador personal.
* Funcionalidades:
  + Vídeos de ejercicios grabados o de libre uso.
  + Audios de instrucciones y motivación.
  + Temporizador configurable para rutinas.
  + Dinamizador virtual con mensajes automáticos.
  + Modo offline y control de resolución/bitrate.
* Herramientas: Android Studio, FFmpeg, CapCut, OBS Studio.

### **E. Atención a la diversidad**

* Nivel III: plantillas, videotutoriales, apoyo técnico continuo.
* Nivel IV: descomposición de tareas, soporte individualizado.

### **F. DUA**

* Representación: vídeos subtitulados, interfaces intuitivas.
* Expresión: variedad de herramientas y temas.
* Implicación: aplicación real, presentación gamificada.

### **G. Evaluación**

* Rúbricas: integración técnica, diseño, accesibilidad y documentación.
* Instrumentos: presentación oral, prueba funcional, memoria técnica.

### **H. Conclusión didáctica**

* Desarrollo de competencias en programación, tratamiento multimedia y ética digital.
* La app como producto funcional, motivador y aplicable en contextos reales.