UNIDAD 1 – SISTEMAS INFORMATICOS Y HARDWARE

¿Qué es la informática?

Información + automática = Informática.

Misión → Desarrollar métodos que permitan el procesamiento, almacenamiento y transmisión de información en formato digital.

Desarrollo histórico

Origen de la idea primigenia, nos tenemos que remontar cientos de años antes del nacimiento de Cristo → Abaco chino (primer instrumento realiza operaciones semiautomáticas).

A partir de la era de la electromecánica y la electrónica → Aparición del "primer ordenado".

Años 40 – 50

Z1, ENIAC, EDVACK, MARCK II, UNIVAC.

Construidos a partir de válvulas de vacío.

Tarjetas perforadas para introducir datos y programas. Se codifica la información (datos) e instrucciones (programas) en forma de perforaciones según un código binario.

- Z1 → 1983, Konrad Zuse, primer computador electromecánico programable, primer ordenador. 1.000 kg y velocidad 1 Hz.
- ENIAC → 1946, Eckert y Mauchly, realiza operaciones matemáticas que ayudan al cálculo de trayectorias. 27.000 kg y velocidad 5 KHz. Ocupa 200 m^2.

Años 50 – 60

Transistores. Reducción de peso, tamaño, consumo eléctrico y coste.

Mayor velocidad en las operaciones (velocidad a microsegundos).

Primeros lenguajes de programación (COBOL y FORTRAN).

Años 60 – 70

Circuito integrado (chip/microchip).

PDP-8 → Primer miniordenador.

Primeros sistemas operativos y se permite la renovación de periféricos.

Años 70 – 80

Procesador o microprocesador.

Concepto de microcomputador u ordenador personal (IBM PC 1981).

Nacimiento Apple y Microsoft → Sistemas operativos.

MS-DOS y PC-DOS

Años 80 – 90

Desarrollo inteligencia artificial → Dotan a las máquinas de capacidad de autoaprendizaje y resolución de problemas de forma similar a la inteligencia, razonamiento y cognición humanas.

Velocidades de GHz.

Nanotecnología.

Redes de área amplia (WAN) y desarrollo masivo de internet y WWW.

Años 2000 en adelante

Reducción del tamaño de los componentes. Comunicación móvil de banda ancha.

Mejora de WWW con Web 2.0 y redes sociales.

Redes fijas de muy alta velocidad.

Movimiento tecnológico Marker y DIY.

Aplicaciones en domótica, robótica, impresión 3D…

Open source de autopilotos para vehículos aéreos no tripulados (drones).

Aparición del Internet de las cosas o IoT.

Aplicaciones de geolocalización.

Representación de la información

Señal analógica → Puede tomar cualquier valor dentro de un intervalo continuo (altura persona, intensidad corriente o voltaje).

Señal digital → Puede tomar valores enteros discretos (3 estados de un semáforo o interruptor).

Los ordenadores solo pueden procesar información digital. El sistema empleado para la transmisión de la información → **Sistema numeración binario o código binario** (secuencia bits, valores 0 y 1). Es más fácil emitir y codificar información, recibirla y decodificarla, almacenarla o procesarla, en 2 estados.

El lenguaje binario es conocido como lenguaje máquina.

1B	8 bits
1KB	1.024 B
1MB	1.024 KB
1GB	1.024 MB
1TB	1.024 GB
1PB	1.024 TB

1 byte = 8 bits → Representan hasta 256 unidades de informática.

El código estándar que empleamos para traducir número y letras del alfabeto latino, anglosajón u otros al sistema binario → Código ASCII o tabla ASCII.

Estructura del ordenador

Principal función → Procesar datos, mediante 3 fases:

- Recibir datos entrada.
- Procesar esos datos.
- Generar una salida presentando un resultado.

Hardware → Parte física. Placa base, procesador, memoria, cableado, tarjetas expansión… Según la arquitectura de Von Neumann:

1.- Unidad central de proceso

Elemento principal. Coordina, controla y realiza todas las operaciones del sistema. Microprocesador o procesador.

Unidad de Control (UC) → Gobierna las actividades de un ordenador y se encarga de interpretar las instrucciones, determinar la correcta secuencia de ejecución.

Unidad Aritmético-Lógica (UAL) → Encarga cálculos aritmético y lógicos. Calculadora muy avanzada integrada dentro de CPU.

2.- Memoria

Conjunto de circuitos integrados (chips) y dispositivos que permiten recuperar (lectura) y almacenar (escritura) datos.

Memoria cache → Pequeño tamaño y gran velocidad. Almacena instrucciones y datos a los que el procesador accede más frecuentemente (principios localidad espacial y localidad temporal). Varios niveles de cache (L1, L2, L3...). Jerarquía de memoria.

Memoria RAM → Memoria de acceso aleatorio. Volátil, la información se destruye al apagar el ordenador. Cargan programas y datos. Muy rápida (no tanto como la cache).

Memoria ROM → Solo lectura, no volátiles, no se pueden borrar ni modificar. BIOS (software que localiza los elementos básicos para cargar el sistema, se encuentra en una pequeña memoria ROM en la placa base).

Memorias auxiliares → Dispositivos externos (discos duros, CD's, memorias USB...). No volátiles, de gran capacidad de almacenamiento.

1. Disco duro: Sistema de grabación electromagnética. El tiempo de acceso depende de su ubicación en el disco. Discos duros de estado sólido (SSD) son de mayor velocidad, menor ruido y menos sensibles a golpes, vida útil inferior (chips de memoria Flash).
2. CD – DVD – BD: Disco óptico de tecnología láser con datos de solo lectura. Son memorias tipo ROM.
3. Memoria USB: Memoria Flash. Pequeño y portable. Buena velocidad. No fiabilidad aceptable a largo plazo.
4. Almacenamiento cloud (nube): Almacenamiento en servidores de alta velocidad gestionados, custodiados y protegidos por empresas externas, garantizar servicio 24/7/365, accesible desde la web y múltiples aplicaciones y dispositivos.

3.- Periféricos

Elementos a través de los cuales se introduce o extraer información.

ENTRADA	SALIDA	MIXTOS
Teclado	Monitor	Modem
Ratón	Impresora	Router
Joystick	Plotter	Pantalla táctil
Lector código barras	Altavoces	Impresor multifunción
Escáner		
Micrófono		
Tableta digitalizadora		

Lenguajes de programación

Lenguaje formal diseñado para crear programas informáticos. Formado por un conjunto de símbolos, una sintaxis u orden correcto de los elementos y expresiones, y una semántica. Mediante ellos, se escriben programas.

Programación → Proceso por el cual se escribe, prueba, depura, interpreta o compila y se mantiene el código fuente de un programa informático.

Lenguajes de bajo nivel

Lenguajes ensambladores, reglas nemotécnicas (ADD, SUB, MUL…) usado por el humano pero que a su vez lo hace fácilmente traducible al lenguaje máquina.

VENTAJAS	INCONVENIENTES
Crear programas más eficientes (banca, automoción, aeronáutica…)	Muy dependiente de la máquina, cada maquina tiene su propio lenguaje.
Controlar la maquina con tanto detalle como quiera. El programador tiene control completo sobre el funcionamiento interno del programa y su ejecución.	Las instrucciones son elementales o básicas. Requiere un gran esfuerzo.
	Programador ha de conocer perfectamente el hardware.

Lenguajes de alto nivel

Más próximos al lenguaje humano. Cada instrucción corresponde a varias o muchas instrucciones en lenguaje máquina. HTML, Java, JavaScript, PHP, Cobol…

VENTAJAS	INCONVENIENTES
Mayor independencia de la máquina.	El programador no conoce en detalle lo que realiza internamente.
Programación más fácil y se cometen menos errores.	No son tan rápidos ni eficientes.
Incluyen rutinas de uso frecuente.	

Ambos lenguajes (bajo y alto nivel) tiene que ser posteriormente traducidos al lenguaje máquina.

- Traducción se hace con anterioridad a la ejecución del programa → Lenguajes compilados (ASP, C++, C#, .NET…). Cuando ejecutamos un programa con extensión .exe.
- Si la traducción se hace al vuelo (momento de ejecutarse) → Lenguajes interpretados (Java, PHP, Python…), se van interpretando línea a línea.

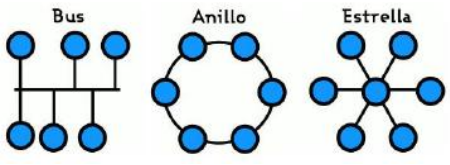
Redes de ordenadores e internet

Red o interconexión de ordenadores → Proporciona un sistema de transmisión de datos que permite compartir recurso, como unidades de disco, datos, impresoras y programas, independientemente de donde se encuentren lo usuarios

Tipos de redes

Redes LAN → Redes de área local, uso privado (casas, pequeñas empresas, colegios…). Rápida interconexión y transmisión de datos.

- Topología en bus o línea: Ordenadores se conectan uno tras otro a la línea de transmisión. La red solo se ve afectada por los fallos en línea.
- Topología en anillo: La línea de transmisión se cierra a través de las conexiones en los equipos, el fallo en un equipo causa un falle de red.
- Topología en estrella: Más empleada. Se conectan a un nodo común (router), gestiona las comunicaciones entre ellos. Vulnerable al fallo del nodo central.



Redes WAN → Redes de área amplia, conectan ciudades, países o continentes. Mas lentas (imperceptibles).

Elementos principales de una red:

Servidores: Genera y ofrecen algún servicio a los demás, y ordenadores clientes, que acceden a sus servicios.

- Redes clientes-servidor centralizan el acceso a recursos y tienen uno o más servidores para atender a los clientes.
- Redes Peer to Peer (P2P) son redes donde no existe jerarquía y los servicios están distribuidos, cada ordenador es un servidor y cliente de los demás.

LENGUAJES DE PROGRAMACIÓN

Programas: Secuencias ordenes/instrucciones que le indican a un dispositivo electrónico/informático, mediante lenguaje de programación, que hacer y en qué orden, sin ambigüedad y con finalidad concreta.

Lenguaje de programación: Lenguaje formal (símbolos, sintaxis y semántica) para crear programas informáticos.

Al **programar**, implementan algoritmos, métodos para la resolución de problemas. Dan de ser:

- Precisos: Sin ambigüedades o casos sin contemplar.
- Ordenados: instrucciones dispuestas en un orden a realizar.
- Finitos: Con un final.

Lenguaje maquina: 0 / 1.

Lenguaje de bajo nivel: Se representan 0 y 1 con lenguaje mnemónico (lenguaje ensamblador).

Lenguaje de alto nivel: Más próximos al humano (Java, PHP, Python).

Proceso de traducción de un programa en alto o bajo nivel a lenguaje máquina.

- Antes de ejecutar el programa (compilación). Un programa compilador traduce el código fuente, escrito por el programador, a un fichero ejecutable compuesto de 0 y 1. C y C++.
- Mientras se ejecuta el programa (interpretación). Un programa interprete traduce el código fuente, escrito por el programador, en tiempo real e instrucción a instrucción, leyéndola y convirtiéndola a 0 y 1 y ejecutándola… PHP y Python.

Conversión binario-decimal

Desarrollamos el numero como la suma de: Digito multiplicado por la base del sistema de numeración (2) elevada a la posición (comenzando por posición 0 de D→I).

Conversión decimal-binario

Dividimos entre la base sucesivamente hasta que el cociente sea menor que la misma. El numero binario equivalente será la secuencia del ultimo cociente seguida del último resto, del penúltimo resto…. Hasta el primer resto.

Sistema numeración hexadecimal

Sistema de numeración posicional que tiene como base el 16, 16 dígitos diferentes van del 0 al 9 seguidos de letra A (equivalente a 10_{10}) a la F (equivalente a 15_{10}).

Nos permite indicar grandes secuencias de numero binarios de manera equivalente y abreviada (indicar direcciones de memoria, operaciones de CPU…).

Cada digito en hexadecimal equivale a 4 dígitos (bits) en binario y, por tanto, con 2 dígitos en hexadecimal podemos expresar 8 dígitos binarios (1 byte).

UNIDAD 2 – COMANDOS LINUX

Comandos: Instrucción que se envía al SO, a través de una interfaz de línea de comandos.

Sintaxis: "$nombre_comando -parametros arg1 arg2... argN"

Comandos de ayuda → **"man":** muestra el manual/ayuda del comando que le pasemos como parámetro. Consultar funcionamiento del resto de comandos.

Comandos básicos

"ls": Muestra listado con ficheros y directorios de un directorio/carpeta determinada. Si no indicamos parámetros, muestra el contenido del directorio actual.

- Parámetro -l: Información mas detallada.
- Parámetro -a: Elementos ocultos del sistema.
- Parámetro *: Listado recursivo, mostrara el contenido de las carpetas contenidas en el directorio listado.

Caracteres comodines:

- **"*"** para sustituirlo por cualquier cadena de 0 o + caracteres. "ls h*" – Listar ficheros y directorios que empiecen por y tengan a continuación 0 o + caracteres.
- **"?"** para sustituirlo por cualquier carácter individual. "ls h?????" – Listar ficheros que empiecen por h y tengan exactamente 5 caracteres más.

"cd": Desplazarnos a un directorio.

- "cd nombre_directorio"

*De forma relativa, para subir al directorio superior/padre, utilizar "cd .." (si utilizas "cd ." te quedas en la carpeta actual).

"mkdir": Crear un directorio vacío.

"rmdir": Borrar un directorio vacío.

"rm": Borrar un fichero.

- Borrar varios ficheros de un directorio → "*".
- Borrar directorios y subdirectorios forzando el borrado → "-rf".

Para crear un fichero:

- "touch nombre_fichero".
- "cat > nombre_fichero + "intro" + "texto_fichero" + "Ctrl+D".

"cp": Copiar un fichero → "cp fichero_origen fichero_destino".

*Copiar varios ficheros origen a la vez a un mismo directorio destino → "cp origen1 origen2 origen3 destino"

Copiar todos los ficheros de un directorio → "*".

Copiar los subdirectorios de manera recurrente → "-r".

"mv": Mover directorios y ficheros. Similar a "cp" pero el resultado elimina cada origen una vez copiado en destino.

CONTENIDO DE FICHEROS

Redirección

En vez de mostrar el contenido de un directorio en pantalla, quiero redireccionarlo a un fichero para guardar directamente esta salida en el fichero. "ls > fichero" → "ls > salida.txt".

La redirección ">" sobrescribe (destruye) el fichero destino. Si queremos añadir contenido al final de un fichero existente ">>" → "ls >> salida.txt".

"echo": Muestra por salida estándar, la cadena de texto que recibe como parámetro → "echo "texto"".

- Introduce un salto de línea después del texto. Si quiere evitarlos "echo -n "texto"".

"cat": Muestra el contenido de uno/varios ficheros en salida estándar → "cat texto.txt".

*Tanto "echo" como "cat" permiten el uso de redirección de la salida.

Visualizar el contenido de varios ficheros, indicando rutas separadas por un espacio → "cat salida.txt texto.txt".

"wc": Cuenta líneas ("-l"), palabras ("-w") o caracteres ("-c") de un fichero.

Tuberías → **"|":** Combinar varios comandos para ejecutarlos en un único mandato. La salida/resultado del 1er comando se envían a la entrada del 2do, y así sucesivamente.

"ps": Con "aux", nos muestra todos los procesos que hay ejecutándose en el sistema.

- Proceso: Elemento dinámico que representa una instancia de un programa en ejecución.

"kill": Finalizar un proceso en ejecución. Finalizar un proceso concreto → "kill -n".

"killall": Finalizar todos los programas de un tipo → "killall nombre_proceso".

"grep": Búsqueda de cadena de caracteres. Si generamos un fichero con una serie de nombres, podemos filtrar aquellas líneas que contengan una cadena concreta → "grep "cadena" ruta".

- Si queremos encontrar todas las cadenas que aparezcan en los ficheros de nuestro directorio actual → "grep "Jose" ./*".
- Si quisiéramos buscar dentro de los subdirectorios (búsqueda recursiva), con parámetro "-r".

"find": Buscar ficheros/carpetas en una ruta determinada.

"head": Acompañado de un numero natural como parámetro "-n", muestra las **"n primeras líneas"** de un fichero → "cat nombres.txt | head -3".

"tail": Similar a "head", nos muestra las **"n ultimas líneas"** de un fichero.

- "cat nombres.txt | tail -2".
- "cat nombres.txt | head -3 | tail -1".

AWK: Procesar patrones en líneas de texto.

- Acceder a datos con formato tabla.
- Acceder a datos separados por un delimitador.

Se considera un lenguaje por su potencialidad y versatilidad.

Leer ciertas columnas de una tabla → "awk '{print $numero_columna_1, ..., $numero_columna_n}'".

Ejemplo → Ejecutamos "ls -l". Para que el comando **awk** muestre el valor de las **columnas 3 y 9**:

- "$ ls -l | awk '{print $3, $9}'.

Nos permite localizar coincidencias en el contenido de ficheros, dado un patrón de búsqueda. El patrón de búsqueda se denomina "**expresión regular**" → Denominación más genérica utilizada en otros ámbitos, y en detalle, es una secuencia de caracteres.

Sintaxis → "$ grep [opciones] [expresión regular] [archivos]".

- [opciones]: Indicar parámetros para personalizar uso de comandos.
- [expresión regular]: Definir el patrón búsqueda.
- [archivos]: Expresar los archivos (ruta) donde se realizará la búsqueda.

Buscar en el contenido de un fichero

Buscar cadena de texto en un fichero conocido → "$ grep "texto" fichero.txt" o "$ cat fichero.txt | grep "texto"".

*El comando devolverá las líneas del fichero que contengan alguna coincidencia con el patrón de búsqueda, la cadena "texto".

Opciones podemos indicar como parámetros:

- "-i": No tendrá en cuenta mayúsculas y minúsculas → "$ cat fichero.txt | grep -i "texto"".
- "-v": Devuelve resultado inverso, es decir, las líneas que no cumplan el patrón → "$ cat fichero.txt | grep -v "texto"".
- "-n": Numero de línea donde sucede la coincidencia → "$ cat fichero.txt | grep -n "texto"".
- "-W": Encuentra coincidencia solo si el patrón coincide con una palabra completa → "$ cat fichero.txt | grep -w "texto"".

*Se pueden combinar → "$ cat fichero.txt | grep -iw "texto"".

Expresión regular

Son una secuencia de caracteres que conforma un patrón de búsqueda. Permite definir una serie de aspecto con mayor nivel de detalle.

- "^": Indica que dicha cadena debe aparecer, al comienzo de una línea del fichero → "$ cat fichero.txt | grep "^texto"".
- "$": Igual que el anterior, pero al final del fichero → "$ cat fichero.txt | grep "texto$"".
- ".": Sustituye a cualquier carácter (similar al "?") → "$ cat fichero.txt | grep "t...o"".
- []: Lo que va dentro, se entiende como un conjunto de posibilidades que deben cumplir el resultado:
  - "$ cat fichero.txt | grep "[a-z]*texto"" → Busca una cadena formada por una letra desde la a hasta la z, seguida de la palabra "texto". Encontraría línea de fichero, si contiene → **Contexto o pretexto.**
  - "$ cat fichero.txt | grep "U[0-9]" → Busca una cadena formada por la letra "U" seguida de un numero de 0 a 9. Encontraría la línea del fichero si contiene → **"U1" o "U2".**
  - "$ cat fichero.txt | grep "[a-t]*texto"" → "*" indica que la posible letra se repite de 0 a n veces, en este caso acotando el conjunto desde la a hasta la t, y después va seguida la palabra "texto". Encontraría línea del fichero con → **Contexto, pretexto, texto...**
  - "$ cat fichero.txt | grep "U[0-1]\{1,2\}" → Busca una cadena formada por "U", seguida de 1 o 2 números acotados entre 0 y 1. Encontraría línea del fichero con → **"U0" o "U1" o "U10" o "U01".**

Buscar en múltiples ficheros → "grep [opciones] [expresión regular] [ruta_archivos]".

[ruta_archivos], puede ser:

- Todos los archivos de la carpeta actual (ruta relativa) → "./*".
- Archivos que empiezan por "lib", de la carpeta padre (ruta relativa) → "../lib*".
- Archivos texto de carpeta de usuario (ruta absoluta) → "/home/usuario/*txt".
- Cualquier otra ruta absoluta o relativa.

[opciones], puede ser:

- "-r": Búsqueda recursiva, para que busque también en los subdirectorios.
  - "$ grep -r "texto" ./*" → Busca cadena "texto" en todos los ficheros de la carpeta actual y en todos los subdirectorios. Nos dará la ruta de los ficheros en los que encuentre la cadena "texto" y, dentro de cada fichero, las líneas en las que ha localizado la coincidencia.

Ejemplo combinado

En base a "fichero.txt" que nos proporcionan y guarda datos personales.

Muestra el nombre y apellidos (columnas séptima y octava respectivamente).

Socios que contengan en sus datos el patrón siguiente:

- 5 dígitos numéricos entre 0 y 9, seguidos de una de las letras "L" o "H".

"$ cat fichero.txt | grep "[0-9]\{5}[]LH}" | awk '{print $7, $8}'"