Hardware	Software
La parte física del ordenador que podemos ver y tocar: teclado, ratón, monitor, discos duros, placa base...	La parte encargada de dar instrucciones al hardware, hacer ejecutar la computadora y almacenar los datos necesarios para los programas.

El **Software** se puede dividir en múltiples categorías según diferentes criterios: Basadas en el tipo de trabajo que realizan

- **De sistema**: programas que permiten el funcionamiento del hardware y sirven de base para otros programas (Windows, Linux...)
- **De aplicación**: Software diseñado para realizar tareas específicas para el usuario (Word, Excel...)
- **De programación**: Herramientas para desarrollar otros programas (Visual Studio, Eclipse...)
- **De uso específico**: programas especializados en una función concreta (software de gestión médica, software contable...)
- **Multimedia**: programas para crear, editar o reproducir audio, video e imágenes (Photosop, VLC, Audacity...)

Basadas en el método de distribución

- **Shareware**: software de prueba que se puede usar gratis por un tiempo limitado o con funciones reducidas. Luego se debe pagar para desbloquear la versión completa
- **Freeware**: programas gratuitos que se pueden usar sin límite de tiempo, aunque los derechos de autor siguen siendo del creador (Adobe Acrobat Reader, Skype...)
- **Adware**: software que se distribuye gratis pero muestra publicidad para financiarse (algunas apps móviles gratuitas)

Teniendo en cuenta la licencia

- **Software libre**: permite al usuario usar, modificar y distribuir el programa libremente. Generalmente con licencias como GNU GPL (GIMP, Linux...)
- **Software propietario**: su código fuente no está disponible y tiene restricciones de uso y distribución. Normalmente requiere pago (Microsoft Office, Photosop...)
- **Software de dominio público**: programas sin restricciones de copyright; cualquiera puede usarlos, modificarlos y redistribuirlos libremente

El Ecosistema del Desarrollo de Software

Conceptos Fundamentales El desarrollo de software requiere comprender la interacción entre hardware y software, así como los diferentes tipos de programas que podemos crear Cada categoría de software tiene un propósito específico y utiliza herramientas particulares para su desarrollo y distribución ==Importante==: El software es la parte lógica que da la vida al hardware, permitiendo que los ordenadores realicen tareas útiles para los usuarios

Licencias de Software: Libre y propietario

Una licencia es un contrato entre el desarrollador de un software y el usuario final. En él se especifican deberes y derecho de ambas partes. El desarrollador es el que especifica que tipo de licencia distribuye

Software Libre

El autor de la licencia concede libertades al usuario, entre ellas están:

- Libertad para usar el programa con cualquier fin
- Libertad para saber cómo funciona el programa y adaptar el código a nuestras propias necesidades
- Libertad para poder compartir copias a otros usuarios
- Libertad para poder mejorar el programa y publicar las modificaciones realizadas

Software Propietario

Este software no nos permitirá acceder al código fuente del programa y de forma general nos prohibirá la redistribución, la reprogramación, la copia o el uso simultáneo en varios equipos

Restricciones Principales	Variantes
- Sin acceso al código fuente - Prohibida la redistribución - No se permite reprogramación - Restricciones de copia - Limitaciones de uso simultáneo	- Freeware - Shareware

Software de Dominio Público

Este software no pertenece a ningún propietario y carece de licencia, por lo que todo el mundo lo puede utilizar. Características clave:

- Sin propietario
- Sin licencia
- Uso universal
- Código fuente disponible

Licencia GPL

la licencia que más se usa en el software libre es la licencia GPL (GNU) que nos dejara usar y cambiar el programa, con el único requisito que se hagan públicas las modificaciones realizadas. Permite:

- Usar el programa
- Cambiar el programa
- Distribuir copias Único Requisito:
- Las modificaciones realizadas deben hacerse públicas

Concepto de programa Informático

Un programa informático es un grupo de instrucciones que están escritas en un lenguaje de programación sobre el que se aplican una serie de datos para resolver un problema. Es decir, el ordenador necesita que esté en lenguaje máquina y para ello tendremos que usar un compilador.

Una vez hecho esto tendremos que procesar todas las instrucciones pasándolas a la memoria principal

Programa y componentes del sistema informático

Necesitamos recursos hardware del ordenador (procesador, RAM dispositivos E/S...) si queremos iniciar un programa. Las instrucciones para inicializar el programa se cargan en la memoria principal y se ejecutarán en la CPU

Componentes principales de la CPU

Unidad de control (UC): Se encarga de interpretar y ejecutar las instrucciones que se almacenan en la memoria principal y, además, genera las señales de control necesarias para ejecutarlas

Unidad Aritmético-Lógica(ALU): Es la que recibe los datos y ejecuta operaciones de cálculo y comparaciones, además de tomar decisiones lógicas (verdaderas/falsas), pero siempre supervisada por la Unidad de Control

Los registros: Son los que almacenan la información temporal, almacenamiento interno de la CPU

Registros de la Unidad de Control

Contador de programa (CP)	Registro de Instrucciones (RI)	Registro de dirección de memoria (RDM)	Registro de Intercambio de memoria (RIM)
Contendrá la dirección de la siguiente Instrucción para realizar	Es el que contiene el código de la Instrucción, se analiza dicho código. Consta de dos partes: el código de la operación y la dirección de memoria en la que opera	Tiene asignada una dirección correspondiente a una posición de memoria que va a almacenar la información mediante el bus de direcciones	Recibe o envía, según si es una operación de lectura o escritura, la información contenido en la posición apuntada por el RDM

Componentes adicionales de control

Decodificador de instrucción (DI): Extrae y analiza el código de la instrucción contenida en el RI, y además, genera las señales para que se ejecute correctamente la acción

El Reloj: Marca el ritmo de DI y nos proporciona unos impulsos eléctricos con intervalos constantes a la vez que marca los tiempos para ejecutar las instrucciones

El Secuenciador: Son órdenes que se sincronizan con el reloj para que ejecuten correctamente y de forma ordenada las instrucciones

Fases de ejecución de instrucciones

Cuando ejecutamos una instrucción podemos distinguir dos fases: **1. Fase de búsqueda**: Se analiza la instrucción en la memoria principal y se envía a la UC para poder procesarla **2. Fase de ejecución**: Se ejecutan las acciones de las instrucciones

Para que podamos realizar operaciones de lectura y escritura en una celda de memoria, se utilizan el RDM, RIM y el DI. El Decodificados de Instrucciones es el encargado de conectar la celda RDM con el registro de intercambio RIM, la cual posibilita que la transferencia de datos se realice en un sentido u otro según sea de lectura o escritura

Código fuente, código objeto y código ejecutable; máquinas virtuales

En la etapa de diseño construimos las herramientas de software capaces de generar un código fuente en lenguaje de programación. Estas pueden ser los diagramas de flujo o el pseudocódigo. La etapa de codificación es la encargada de generar el código fuente y pasa por diferentes estados

Tipos de código

1. Código fuente: Es el código realizado por los programadores usando algún editor de texto o herramienta de programación. Posee lenguaje de alto nivel y no se puede ejecutar directamente desde el ordenador **2. Código objeto**: Es el código que se crea tras realizar la compilación del código fuente. Este código es una representación intermedia de bajo nivel **3. Código ejecutable**: Este código se obtiene tras unir el código objeto con varias librerías para que así pueda ser ejecutado por el ordenador

Compilación

Es el proceso a través del cual se convierte un programa en lenguaje máquina a partir de otro programa de computadora escrito en otro lenguaje. La compilación se realiza a través de dos programas: **compilador** y **enlazador** ![[Pasted image 20251020132311.png]]

Fases del compilador

![[Pasted image 20251020132353.png]]

Análisis del código

Análisis léxico: Se lee el código obteniendo unidades de caracteres llamados tokens **Análisis sintáctico**: Recibe el código fuente en forma de tokens y ejecuta el análisis para determinar la estructura del programa **Análisis semántico**: Revisa que las declaraciones sean correctas, los tipos de todas las expresiones, los tipos de todas las expresiones...

Generación y optimización

Generación de código intermedio: Después de analizarlo todo, se crea una representación similar al código fuente para facilitar la tarea al traducir al código **Optimización de código**: Se mejora el código intermedio anterior para que sea más fácil y rápido su interpretación **Generación de código**: Se genera el código objeto

Máquinas Virtuales

Son un tipo de software capaz de ejecutar programas como si fuese una máquina real. Esto nos permite crear entornos aislados y multiplataforma para la ejecución de aplicaciones

Clasificación de máquinas Virtuales

Máquinas virtuales de sistema: Nos permiten visualizar máquinas con distintos sistemas operativos en cada una. Ej: VMware, VirtualBox... **Máquinas virtuales de proceso**: Se ejecutan como un proceso normal dentro de un S.O y solo soporta un proceso. Se inician cuando lanzamos el proceso y se detienen cuando este finaliza. El objetivo es proporcionar un entorno de ejecución independiente del hardware y del S.O y permitir que el programa sea ejecutado de la misma forma en cualquier plataforma. Ej: máquina virtual de Java

Máquina virtual de Java

Los programas que se compilan en lenguaje Java son capaces de funcionar en cualquier plataforma (Unix, MAC, Windows...). Esto se debe a que el código no lo ejecuta el procesador del ordenador sino la propia Máquina Virtual de Java

Funcionamiento de la JVM

01	02	03	04
Código fuente	**Compilación**	**Bytecode**	**Ejecución**
El código estará escrito en archivos de texto planos con la extensión Java	El compilador javac generará uno o varios archivos siempre que no se produzcan errores y tendrán la extensión .class	Este fichero .class contendrá un lenguaje intermedio entre el ordenador y el S.O y se le llamará bytecode	La Java VM coge y traduce el bytecode en código binario para que el procesador de nuestro ordenador sea capaz de reconocerlo. Los ficheros .class podrán ser ejecutados en múltiples plataformas

![[Pasted image 20251023131846.png]]

Tareas y Consideraciones de la JVM

Tareas que puede realizar la máquina virtual Java:

- La reserva de espacio para objetos creados y liberar aquella memoria que no se usa
- Comunicación con el sistema en el que se ejecuta la aplicación para varias funciones
- Observar que se cumplen las normas de seguridad para las aplicaciones Java

Desventajas Son más lentos que los lenguajes ya compilados, debido a la capa intermedia. No es una desventaja demasiado crítica

Clasificación de los lenguajes de programación

Podemos clasificar los lenguajes de programación basándonos en los siguientes criterios

Según su nivel de abstracción	Según la forma de ejecución	Según el paradigma de programación
Desde lenguajes de bajo nivel hasta alto nivel	Compilados o interpretados	Imperativos, funcionales, lógicos, estructurados y orientados a objetos

Clasificación según su nivel de abstracción

1. **Lenguaje de bajo nivel**: Lenguaje de máquina: Es el lenguaje de más bajo nivel por excelencia, el que entiende directamente la máquina. Utiliza lenguaje binario y los programas específicos para cada procesador. Lenguaje ensamblador: Al lenguaje de máquina le sigue este. Es complicado de aprender y es específico para cada procesador. Cualquier programa escrito en este lenguaje tiene que ser traducido al lenguaje de máquina para que se pueda ejecutar. Se utilizan nombres nemotécnicos y las instrucciones trabajan directamente con registros de memoria física

2. **Lenguajes de nivel medio**: Posee características de ambos tipos de nivel, tanto del nivel bajo como del alto, y se suele usar para la creación de sistemas operativos. Ej C
3. **Más fáciles de aprender y usar**: Este tipo de lenguaje es más fácil a la hora de aprender, ya que para usarlo lo hacemos con palabras que solemos utilizar. El idioma que se suele emplear es el inglés y para poder ejecutar lo que escribamos necesitaremos un compilador para que traduzca al lenguaje máquina las instrucciones. Este lenguaje es independiente de la máquina ya que no depende del hardware del ordenador Ej: C++, Java, COBOL...

Clasificación según la forma de ejecución

Lenguajes compilados: Al programar en alto nivel hay que traducir ese lenguaje a lenguaje de máquina a través de compiladores. Los compiladores traducen desde un lenguaje fuente a un lenguaje destino. Devolverá errores si el lenguaje fuente está mal escrito y lo ejecutará si el lenguaje destino es ejecutable por la máquina. Ej: C, C++, C#...

Lenguajes interpretados: Son otra variante para traducir programas de alto nivel. En ese caso nos da la apariencia de ejecutar directamente las instrucciones del programa fuente con las entradas proporcionadas por el usuario. Cuando ejecutamos una instrucción se debe interpretar y traducir al lenguaje de máquina. Ej: PHP, JavaScript, Python...

Compiladores vs Intérpretes

El compilador es, de forma general, más rápido que un intérprete al asignar las salidas. Sin embargo, al usar el intérprete evitaremos tener que compilar cada vez que hagamos alguna modificación. El lenguaje Java usa tanto la compilación como la interpretación

Clasificación según el paradigma de programación

El paradigma de programación nos detalla las reglas, los patrones y los estilo de programación que usan los lenguajes. Cada lenguaje puede usar más de un paradigma, el cual resultará más apropiado que otro según el tipo de problema que queramos resolver

Lenguajes imperativos: Establecen cómo debe manipularse la información digital presente en cada memoria o cómo se debe enviar o recibir la información en los dispositivos. Ej: lenguaje de máquina y ensamblador Algunos ejemplos de estos lenguajes son: Java, C, C#, Pascal. Casi todos los lenguajes de desarrollo de software comercial son imperativos

Programación estructurada --> Programación modular --> Programación orientada a objetos

Lenguajes funcionales: Formados por definiciones de funciones junto con argumentos que se aplican. Apenas se usan para el software comercial Ej: Miranda, Haskell

Lenguajes lógicos: basados en el concepto de razonamiento, ya sea de tipo deductivo o inductivo. El sistema es capaz de hacer razonamientos a partir de una base de datos consistente en un conjunto de entidades, propiedades de esas entidades y relaciones entre entidades. Los programas suelen tener forma de base de datos, formados por declaraciones lógicas que podremos consultar. La ejecución será en forma de consultas hacia esa base de datos. El lenguaje lógico más importante es Prolog, preparado para sistemas expertos

Lenguajes estructurados: Utilizan las tres construcciones lógicas nombradas anteriormente y resulta fácil de leer. El inconveniente de estos programas estructurados es el código, que está centrado en un solo bloque, lo que dificulta el proceso de hallar el problema Cuando hablamos de programación estructurada nos estamos refiriendo a programas creados a través de módulos (pequeñas partes más manejables que al unirse hacen que el programa funcione). Cada uno posee una entrada y salida. Aunque cada uno trabaje de forma independiente, deben de estar perfectamente comunicados

Lenguajes orientados a objetos: definido por un conjunto de objetos en vez de por módulos como hemos visto anteriormente. Estos objetos están formados por una estructura de datos y por una colección de métodos que interpretan esos datos. Los datos que se encuentran dentro de los objetos son sus atributos y las operaciones que se realizan sobre los objetos cambian el valor de uno o más atributos

Ventajas	Desventajas
- Facillidad para reutilizar el código - Trabajo en equipo - Mantenimiento del software	- Dificultad para programar en este lenguaje (poco intuitivo) Ej: C++, Java, Ada...