## Introducción

**Computadora**: es un dispositivo electrónico capaz de recibir instrucciones y ejecutarlas. Estas instrucciones son dadas por un usuario, por medio de una interfaz que presenta el sistema operativo lo cual hace más sencilla la comunicación entre el usuario y la máquina. El sistema operativo interpreta y ejecuta con los recursos que dispone. La computadora está compuesta por hardware y software. Puede procesar grandes cantidades de datos, hacer cálculos complejos, comunicarse con otros dispositivos y seres humanos.

**Software**: todo lo que interpreta y ejecuta los deseos del usuario.

* **Sistema**: el software de sistema le procura al usuario y programador adecuadas interfaces de alto nivel, controladores, herramientas y utilidades de apoyo que permiten el mantenimiento del sistema global. Incluye entre otros: sistemas operativos, controladores de dispositivos, herramientas de diagnóstico, herramientas de corrección y optimización, servidores, utilidades.
* **Programación**: es el conjunto de herramientas que permite al programador desarrollar programas de informática. Incluyen: editores de texto, compiladores, intérpretes, enlazadores, depuradores, entornos de desarrollo integrados (IDE)
* **Aplicación**: permite a los usuarios llevar a cabo una o varias tareas específicas, en cualquier campo de actividad susceptible de ser automatizado o asistido, con especial énfasis en los negocios. Incluye: aplicaciones para Control de sistemas y automatización industrial, aplicaciones ofimáticas, software educativo, empresarial, Bases de datos, telecomunicaciones, videojuegos, software de diseño asistido (CAD)

**Hardware**: son los recursos físicos que usa la computadora para ejecutar las instrucciones.

* **De entrada**: componentes que permiten el ingreso de información desde alguna fuente externa. Proveen el medio fundamental para transferir hacia la computadora (al procesador) información desde alguna fuente.
* **De salida**: Son aquellos que permiten dar salida a la información resultante de las operaciones realizadas por la CPU. Los más comunes de este grupo son los monitores, las impresoras, las consolas. y los altavoces.
* **Internos**: componentes físicos que forman parte del dispositivo principal, siendo inseparable de este. Ejemplos de hardware interno: Placa base, CPU, RAM, GPU, HDD, SDD.

**Programar**: ordenarle a una computadora qué hacer, cuándo hacerlo y cómo.

**Servidores**

* Servidor web: almacena documentos HTML, imágenes, archivos de texto, escrituras, y demás material web y distribuye este contenido a clientes que lo piden en la red.
* Servidor de base de datos: provee servicios de base de datos a otros programas u otras computadoras y aquellas computadoras (servidores) dedicadas a ejecutar esos programas.
* Servidor de archivos: es el que almacena varios tipos de archivos y los distribuye a otros clientes en la red.

**Interfaces de usuario**

* Interfaz de línea de comandos (CLI): interfaces alfanuméricas (intérpretes de comandos) que solo presentan texto.
* Interfaz gráfica de usuario (GUI): permiten comunicarse con la computadora de forma rápida e intuitiva representando gráficamente los elementos de control y medida.
* Interfaz nativa de usuario (NUI): pueden ser táctiles, de forma similar a si se accionara un control físico; pueden funcionar mediante reconocimiento del habla, como, por ejemplo, Siri; o mediante movimientos corporales.

**Historia de la computadora**

* 1122 a.c en China: invención del ábaco, que es un instrumento sencillo para realizar operaciones aritméticas.
* 1642 d.c en Francia: Blaise Pascal inventa la pascalina para ayudar a su padre con cálculos de su negocio. Fue la primera calculadora mecánica.
* 1801 d.c en Francia: Joseph Jacquard inventa las tarjetas perforadas como forma de guardar información para diseños de tela complejos.
* Entre 1833 y 1842 en Gran Bretaña: Charles Babbage intentó crear la primera computadora. Aparece el primer algoritmo de la matemática Ada Lovelace.
* 1936 en el Reino Unido: Alan Turing inventó la máquina de Turing. Logra ejecutar programas almacenados. Se creó con la intención de descifrar los mensajes alemanes en la segunda guerra.
* 1943 en USA: se crearon las primeras computadoras conocidas como ENIAC, operada por las chicas del refrigerador, consideradas las primeras programadoras junto a Ada Lovelace.
* 1945 en Hungría: Von Neumann diseñó una arquitectura que establece las pautas actualmente utilizadas de cómo armar una computadora para optimizar sus recursos.
* 40’s: primera generación de computadoras
* 60’s: transistor - chip
* 70’s: microchip
* 80’s: Windows, Apple, Linux

Era cognitiva: según IBM, desde 2011 ingresamos a la era cognitiva donde encontramos tecnología como las computadoras cuánticas que pueden ejecutar en 200 segundos cálculos complejos que las computadoras convencionales más rápidas de hoy en día les llevaría unos 10 mil años.

## La terminal

**IDE**: conjunto de herramientas diseñadas para facilitarnos la creación y el desarrollo de nuestros programas o aplicaciones

**Terminal**: también denominada Interfaz de Línea de Comandos (CLI). Nos permite darle instrucciones a la computadora a través de la ejecución de comandos, que son órdenes que el usuario le proporciona a un sistema informático mediante texto.

Los comandos son más rápidos de ejecutar que en una GUI, y es más directo. Podemos lograr comandos muy complejos de manera muy sencilla. Al utilizar GUI, por detrás se está ejecutando un comando. Los comandos sguen funcionando de la misma manera en el paso del tiempo, da la ventaja de compatibilidad que permite trabajar con cualquier versión. No tiene distracciones.

**Comandos**: son instrucciones codificadas para ser interpretadas por un sistema operativo. Se envían a la computadora a través de la terminal. Pueden tener argumentos y opciones.

**Ruta absoluta**: es la ruta completa de un archivo o directorio. Utiliza la barra / al comienzo.

**Ruta relativa**: damos por sentado que la indicación la damos desde donde estoy parado. No utiliza la barra / al comienzo.

## GIT

**Git**: es un software de control de versiones pensado para la eficiencia y confiabilidad del mantenimiento de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Es una tecnología que permite hacer un backup completo de nuestros archivos para que estén constantemente actualizados. Permite compartir los archivos con el resto del equipo o colaboradores. Con Git podemos tener un historial completo de las versiones de un archivo sin necesidad de crear varias copias del mismo

**Repositorio local**: es el que tiene todos los archivos (que hayas guardado en él) en la computadora. Se crea con el comando git init.

**Identificarse**: debemos decirle a Git quienes somos.Para identificar el usuario usamos git config user.name “nombre de usuario”. Este comando no muestra texto adicional en la terminal. Para identificar email usamos el comando git config user.email “email”. Si a estos dos comandos le agregamos --global estamos indicando que será esta nuestra identificación para todos los repositorios.

**Agregar archivos al repositorio**: cuando creamos nuevos archivos o modificamos los existentes y queremos agregarlos al repositorio debemos usar el comando git add “nombre del archivo”o git add . para agregar todos los archivos del directorio. Con este comando hacemos que Git haga seguimiento de dichos archivos o actualice los archivos en el repositorio local. El comando git status nos da información sobre el estado del repositorio, por ejemplo si hay archivos sin agregar, si hay archivos con modificaciones o si está todo listo para un commit.

**Commits**: son los paquetes que nos van a permitir ir haciendo un seguimiento de los cambios que vamos realizando, dado que cada uno de ellos tiene una timestamp, o fecha de creación, y un autor. Los commits van a ser nuestro historial de cambios que se fueron haciendo en el proyecto. Para hacer un commit usamos el comando git commit -m "mensaje" en donde podemos especificar cuales son las modificaciones que agregamos al archivo. Debemos agregar los archivos antes de hacer un commit. El commit no sube las cosas a Github, a la nube. Solo los commitea en el repositorio local.

## Github

**Github**: aplicación estable, segura, rápida y eficiente. Es un lugar en la nube donde podés alojar archivos de tu proyecto de manera gratuita. Una vez creada la cuenta podemos comenzar a crear repositorios, donde se irán almacenando archivos del proyecto y a través del cual se puede hacer seguimiento del mismo. A cada proyecto le corresponde un repositorio. Los repositorios que viven en Github son repositorios remotos. Cada persona del equipo va a tener una copia del repositorio en su computadora, que son los repositorios locales.

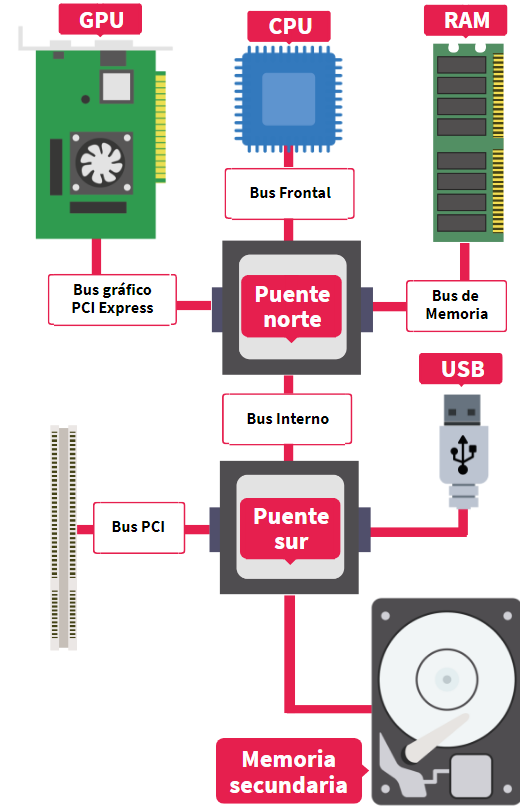
Subiendo archivos: usamos el comando git push origin master. Este repositorio le indica a Git que suba los archivos del repositorio local al remoto, en la nube. “Origin” se refiere a que es el remoto y “master” o “main” se refiere a que es la rama principal del proyecto. Tiene que estar todo commiteado.

Ramas: es una línea paralela en las que puedes agregar nuevas funcionalidades sin tener que modificar la línea original del proyecto.

Bajando archivos: usamos el comando git clone URL\_exacta (donde se encuentran los archivos) para crear una copia exacta de todos los archivos existentes en el repositorio remoto. Se hace solo una vez en cada computadora.

Cuando lo que se quiere es actualizar los archivos pero ya se hizo antes el git clone debemos usar el comando git pull origin master (o el nombre de la rama). Siempre haremos un git pull antes de hacer un push para asegurarnos de no pisar nada cuando subamos los cambios.

Si subimos versiones diferentes al mismo tiempo, Git hace un merge de ambas y las compatibiliza. Pero si estuvimos trabajando en el mismo archivo en el mismo lugar, Git no siempre sabe solucionar este problema. En este caso, Git nos indicará si tenemos conflictos y cómo solucionarlos. Pero en caso de que eso suceda, debemos corregirlo manualmente y hacer un nuevo commit. Por esto siempre debemos hacer un pull antes de un push. Si en este pull nos tira el error de que no puede fusionar ciertas líneas, las debemos trabajar manualmente. Entre los signos menor <<<<< y los signos igual ==== vemos lo que tenemos en nuestro archivo, y debajo, después de los signos igual y antes de los signos mayor >>>>> veremos lo que nos trae desde github. Luego de hacer la fusión manual debo hacer un nuevo add y un nuevo commit y finalmente un push.



## Memorias

**Memoria principal**:

* Se utiliza para almacenar datos o información de forma temporal.
* El procesador puede acceder directamente a los datos almacenados.
* Puede ser de tipo volátil o no volátil.
* Su capacidad es limitada. Actualmente su capacidad puede llegar hasta los 64 gigabytes.
* El acceso a la memoria principal se realiza a través del bus de datos.
* Mayor velocidad que la memoria secundaria.
* Mayor costo que la memoria secundaria.
* La CPU puede acceder a la memoria RAM a través del single channel (se utiliza una única señal a un ancho de banda y frecuencia determinada) o dual channel (permite el acceso simultáneo a dos módulos de memoria; para ello, todos los módulos de memoria deben tener la misma capacidad, velocidad, frecuencia, latencia y fabricante).

**Memoria secundaria:**

* Se puede almacenar información de manera permanente.
* El procesador no puede acceder a los datos de forma directa. Estos deben primero copiarse en la memoria principal para que el procesador pueda leerlos.
* Siempre son de tipo no volátil.
* Puede guardar una gran cantidad de datos e información. Su capacidad llega hasta los terabytes.
* A la memoria secundaria únicamente puede accederse a través de los buses de entrada y salida.
* Su velocidad es menor que la primaria.
* Su costo es menor que la primaria.

**Memoria caché**: se sitúa entre la CPU y la memoria RAM. La CPU copia en ella los datos más relevantes que va a utilizar de la memoria RAM para acceder a ellos más rápidamente. Se divide en tres niveles (L1, L2 y L3) que obedecen a un orden de jerarquía establecido por cercanía al procesador, velocidad y capacidad.

**Dispositivos magnéticos** (Cinta magnética, Diskette, Discos duros): emplea un sistema de grabación magnética para almacenar información. Está formado por uno o más discos que giran a velocidad constante.

**Dispositivos de estado sólido** (Flash, Pendrive, SSD): no posee partes móviles y que permiten la escritura y lectura en múltiples posiciones en la misma operación mediante pulsos eléctricos.

**Unidades ópticas** (CD, DVD, Blu-ray): los datos almacenados pueden ser guardados o leídos a través de un láser.

**Características de las memorias RAM**

* **Velocidad**: MHz
* **Capacidad**: es la cantidad de datos que se pueden almacenar en una RAM. La capacidad se mide en gigabytes (GB).
* **Latencia**: es la cantidad de ciclos de reloj que transcurren entre una petición y su respuesta
* **Voltaje**: hace referencia a la energía consumida por el módulo de RAM.

En Dual channel:

* Las velocidades se suman > Si la velocidad de cada módulo es de 1600 Mhz, la velocidad total será de 3200 Mhz.
* La capacidad se suma > Si cada módulo tiene una capacidad de 8 GB, la capacidad total será de 16 GB.

**Tipos de RAM**:

* FPM (fast page mode): es un tipo de memoria RAM que espera durante todo el proceso de localización de un bit de datos por columna y fila; y luego lee el bit antes de comenzar con el siguiente. La velocidad máxima de transferencia es de unos 176 Mbps.
* SDR (single data rate): es una forma completa de memoria de acceso dinámico síncrono. Tiene tiempos de acceso de entre 25 y 10 ns (nanosegundos) y están en módulos DIMM (módulo de memoria dual en línea) de 168 contactos.
* R (rambus): La memoria dinámica de acceso aleatorio rambus es una forma completa de RDRAM. Este tipo de chips de RAM funciona en paralelo, lo que le permite alcanzar una velocidad de datos de 800 MHz o 1.600 Mbps. Genera mucho más calor al funcionar a tan altas velocidades.
* V (video): Es la memoria RAM optimizada para adaptadores de vídeo. Tienen dos puertos para que los datos de vídeo puedan escribirse al mismo tiempo que el adaptador de vídeo lee regularmente la memoria para refrescar la pantalla actual del monitor.
* EDO (extended data output): No espera a que finalice el procesamiento del primer bit para continuar con el siguiente. En cuanto se localiza la dirección del primer bit, la EDO RAM comienza a buscar el siguiente.
* DDR: Lanzada en el año 2000, aunque no empezó a usarse hasta casi 2002. Operaba a 2.5V y 2.6V y su densidad máxima era de 128 Mb (por lo que no había módulos con más de 1 GB) con una velocidad de 266 MT/s (100-200 MHz)
* DDR2: lanzada hacia 2004, funcionaba a un voltaje de 1.8 voltios, un 28% menos que DDR. Se dobló su densidad máxima hasta los 256 Mb (2 GB por módulo). Lógicamente la velocidad máxima también se multiplicó, llegando a 533 MHz.
* DDR3: El lanzamiento de esta memoria se produjo en 2007 y supuso toda una revolución porque aquí se implementaron los perfiles XMP. Para empezar los módulos de memoria operaban a 1.5V y 1.65V, con velocidades base de 1066 MHz pero que llegaron mucho más allá, y la densidad llegó hasta a 8 GB por módulo.
* DDR4: Lanzada en 2014. Se reduce el voltaje hasta 1.05 y 1.2V, aunque muchos módulos operan a 1.35V. La velocidad se ha visto notablemente incrementada, pero su base comenzó en los 2133 MHz. Actualmente hay módulos de 32 GB.
* DDR5: Lanzada a mediados del 2020 llega a anchos de banda de hasta 6.4 Gbps en sus modelos iniciales, es la primera memoria DDR de doble canal en un solo chip. Su frecuencia base es de 4800 MHz y, además, su consumo baja por la clásica reducción de voltaje, esta vez a 1.1 V. Su capacidad de almacenamiento máxima en un módulo de memoria es de 128 GB.

**Registros de la CPU**: memorias de muy alta velocidad que se utiliza en los procesadores para acceder a información importante de manera rápida. La CPU tiene 5 registros internos.

* PC: Program counter
* IR: Instructions register
* MAR: Memory address register
* MDR: Memory data register
* Accumulator

**Fragmentación**: espacio que queda desperdiciado al momento de usar los métodos de **partición** de memoria.

* **Externa**: se genera cuando, durante el reemplazo de procesos, quedan huecos entre dos o más procesos de manera no contiguos y cada hueco no se puede ocupar con algún proceso de la lista de espera. Quizás, si unimos todos los huecos, sí sea espacio suficiente, pero se requeriría de un proceso de desfragmentación de memoria o compactación para lograrlo.
* **Interna**: es generada cuando se reserva más memoria de la que el proceso va realmente a usar. Se debe de esperar a la finalización del proceso para que se libere el bloque completo de la memoria.

**Segmentación**: es otra técnica de gestión de memoria que pretende acercarse más al punto de vista del usuario. Los programas se desarrollan en torno a un núcleo central desde el que se bifurca a otras partes o se accede a zonas de datos. Desde este punto de vista, un programa es un conjunto de componentes lógicos de tamaño variable o un conjunto de segmentos, es decir, el espacio lógico de direcciones se considera como un conjunto de segmentos, cada uno definido por su tamaño y un número. La segmentación de un programa la realiza un compilador y en ella cada dirección lógica se expresará mediante dos valores: número de segmento (s) y desplazamiento dentro del segmento (d).

**Paginación:** es una técnica de gestión que permite asignar la memoria de forma discontinua. Con este fin, se divide la memoria en trozos de tamaño fijo llamados armazones o frames y la lógica en bloques de igual tamaño denominados páginas. El sistema operativo mantiene internamente una tabla de páginas donde se relaciona cada página cargada en memoria con el frame que la contenga, es decir, su dirección inicial en memoria real. El sistema operativo analizará cada nuevo trabajo que se disponga a entrar para conocer el número de páginas que ocupa y buscará en su lista de frames libre un número igual de ellos. Si estos existen, cargará en ellos las páginas del programa y construirá la correspondiente tabla de páginas, actualizando la lista de frames libres. Cada trabajo en memoria tendrá su propia tabla de páginas apuntada por el bloque de control del proceso. De esta manera, se logra evitar la fragmentación externa ya que cualquier frame libre es asignable a un trabajo que lo necesite. Por otro lado, seguirá existiendo fragmentación interna puesto que, los trabajos no ocuparán un tamaño múltiplo del tamaño de la página.

## Sistemas Operativos

El sistema operativo es el soporte lógico que controla el funcionamiento del equipo físico. Es un conjunto de programas y funciones que ocultan los detalles del hardware, ofreciendo una vía sencilla y flexible de acceso al mismo.

**Funcionalidades** del sistema operativo / **recursos administrados**:

* Gestionar la memoria de acceso aleatorio y ejecutar las aplicaciones, designando los recursos necesarios.
* Administrar la CPU, gracias al algoritmo de programación.
* Direccionar las entradas y salidas de datos (a través de drives), por medio de los periféricos de entrada y salida.
* Administrar la información para el buen funcionamiento de la PC.
* Dirigir las autorizaciones de uso para el usuario.
* Administrar los archivos.

**Clasificación**:

* Según el usuario:
  + Monousuario (Windows hasta Me, DOS)
  + Multiusuario (Windows a partir de XP, Unix, Linux, Mac OSX)
* Según la gestión de tareas:
  + Monotarea (DOS)
  + Multitarea (Windows, Unix, Linux, Mac OSX)
* Según la gestión de recursos:
  + Centralizado: solo permite utilizar los recursos de un solo ordenador (Windows, Linux, Mac OSX, Unix)
  + Distribuido: permite ejecutar los procesos de más de un ordenador al mismo tiempo (Novell Netware, Windows Server, Cisco IOS, Unix, Linux)
* Según la estructura interna:
  + Monolítica (VMS, Linux, Multics, Windows hasta Me)
  + Jerárquica (Unix, Multics)
  + Máquina virtual (Microsoft Hyper-V, VMware, VirtualBox, QEMU, Kernel-Based Virtual machine)
  + Cliente-servidor

Generaciones:

* Generación 0: computadoras electrónicas digitales sin SO
* Generación 1: Tubos de vacío y tableros enchufables; máquinas enormes que ocupaban cuartos enteros
* Generación 2: Transistores y sistemas de lote; tarjetas perforadas.
* Generación 3: Circuitos integrados ( CI ) y multiprogramación; IBM 360
* Generación 4: Computadoras personales y LSI; MS-DOS y UNIX.

**Kernel**: es la capa fundamental de un sistema operativo, es el encargado de comunicar y administrar los recursos de la computadora, como la RAM o el uso del procesador.

**Tipos de kernel**:

* **Monolítico**: es el más veloz ya que se comunica con llamadas al sistema. El 70% del kernel no es utilizado. Si un sistema falla, todo el núcleo falla
* **Microkernel**: más lento debido a que se comunica con paso de mensajes. Es más fácil agregar nuevas funcionalidades. Requiere más líneas de código.
* **Kernel híbrido**: En esencia es un microkernel con más código “no esencial”, pero menor al de un monolítico puro. Agiliza la velocidad de un microkernel. Compatible para gran variedad de dispositivos.
* **Nanokernel**: El código es aún más reducido que en microkernel, pero más difícil de crear. Todos los servicios se comunican con paso de mensajes. Fácil modificación del sistema operativo.

**Llamadas al sistema:** son la manera en la cual un programa solicita una acción al sistema operativo con el que interactúa. Es el punto de enlace entre el modo usuario y el modo privilegiado del sistema operativo. Lo que permite a las aplicaciones utilizar recursos de hardware.

Los sistemas operativos diferencian las acciones que puede realizar un usuario (modo usuario) de las que no ya que algunas pueden llegar a ser muy dañinas para el sistema operativo, por lo que solo debe controlarlas el mismo (modo privilegiado).

**Clasificación de llamadas al sistema**:

* Gestión de control: supervisa el inicio, creación, detención y finalización de los procesos.
* Gestión de archivos: incluyen la creación, eliminación, apertura, cierre, escritura y lectura de archivos.
* Gestión de dispositivos: administra los recursos disponibles, como el almacenamiento.
* Gestión de información: asegura la puntualidad e integridad de la información.
* Comunicación entre procesos: coordina la interacción entre los distintos procesos y aplicaciones.

## Procesos

### 

Un proceso es la ejecución de un programa o instrucción. Es un conjunto de operaciones que componen un programa, los cuales a la hora de ejecutarse se reparten la utilización del procesador para realizar su tarea.

El cambio de proceso consiste en interrumpir la ejecución de un proceso y pasar a ejecutar otro proceso. El cambio de contexto consiste en el cambio de modo que se produce durante un cambio de proceso. Para cambiar el proceso de A por B, deja de ejecutar el proceso en curso (A), pasa de modo usuario a modo supervisor, luego el núcleo estudia si el proceso (B) está listo para ejecución..

Los procesos no se “almacenan” en la memoria principal ya que son efímeros.

La computadora ejecuta las tareas de acuerdo al plan de acción diseñado por el sistema operativo.

Se crean de manera interactiva por el usuario o en segundo plano (llamadas al SO).

Cuando un proceso no puede resolverse inmediatamente, se crean procesos hijos, los cuales realizan subtareas para que el proceso padre cumpla su objetivo. Un proceso padre puede tener varios procesos hijos, pero los procesos hijos tienen un único padre.

Existen 2 tipos de procesos que se ejecutan de manera concurrente:

* **Procesos independientes**: tienen total autonomía, no pueden afectar ni ser afectados por otros procesos en ejecución. Tienen todos los recursos que necesitan para poder terminar. No se comunican con otros procesos.
* **Procesos cooperativos**: pueden afectar y ser afectados por otros procesos. Trabajan en función de la disponibilidad y los recursos de otros procesos. Existen 2 motivos para compartir recursos:
  + Información compartida: algunos procesos carecen de información suficiente, por lo que deben consultar la compartida para poder ejecutarse.
  + CPU eficiente y veloz: gracias a la información compartida, el CPU trabaja de manera más eficiente y veloz. Esto da lugar a la **modularidad**: ejecución independiente y simultánea de varios pasos de una tarea.

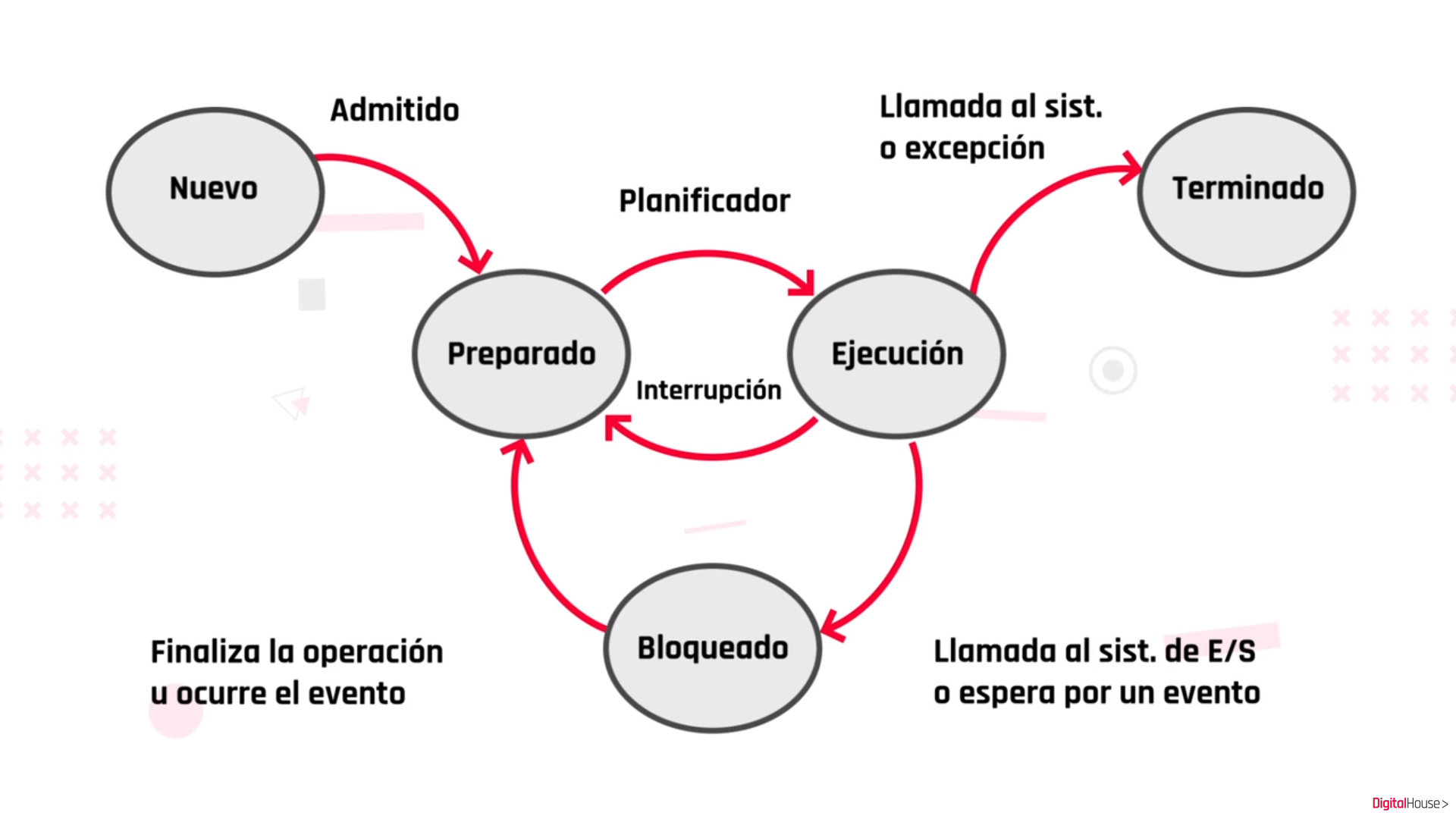
Las tareas de un proceso pueden depender de la finalización de otras tareas. Esto puede ocasionar errores cuando un recurso está bloqueado por la ejecución de otra tarea.

**Mecanismos de comunicación entre procesos** (IPC = Inter Process)

* **Memoria compartida**: se establece una porción de memoria que será compartida por diversos procesos para que puedan intercambiar información. Es el kernel el que establece los permisos.
  + Ventaja: es más económica que usar un multiprocesador.
* **Paso de mensajes/señales**: La comunicación entre los procesos se hace mediante operaciones explícitas de envío y recepción, siendo el kernel el intermediario.
  + Ventajas: no existen errores como la exclusión mutua; son compatibles con cualquier arquitectura de computadora.

**Ciclo de vida/Estados de los procesos**:

* Nuevo: cuando un proceso se crea.
* Listo (o preparado): cuando está en condiciones de ser ejecutado.
* Ejecución cuando su turno de usar el procesador comenzó.
* Bloqueado: cuando está esperando que un proceso o recurso pueda ser utilizado.
* Terminado (o salida): cuando ha sido ejecutado y su ciclo de vida finalizó.



Cuando un proceso está utilizando un recurso lo bloquea de manera que los demás no lo puedan usar. Cuando un proceso se está ejecutando y aparece una llamada de espera, este pasa a una lista de bloqueados y permanece ahí hasta que un proceso diferente le envía la señal de avance. Así, el proceso que estaba bloqueado pasa a una lista de espera para usar la CPU.

**Área o sección crítica**: acceso a un recurso compartido que no debe ser accedido por más de un proceso o hilo en ejecución. Se necesita un mecanismo de sincronización en la entrada y salida de la sección crítica para asegurar la utilización en exclusiva del recurso, por ejemplo un semáforo, monitores, el algoritmo de Dekker y Peterson, los candados.

Se requiere de una buena planificación de procesos para que la cola de procesos no colapse ni se produzca una **inanición** (a un proceso o hilo se le deniega siempre el acceso a un recurso compartido, por lo que la tarea no puede finalizar nunca).

La **planificación** de procesos consiste en las políticas y mecanismos que poseen los sistemas operativos actuales para realizar la gestión del procesador. Su objetivo es dar un buen servicio a todos los procesos que existan en un momento dado en el sistema.

**Técnicas o políticas de planificación de procesos:**

* **FIFO** (First In, First Out): se asigna la CPU al proceso que la solicita primero.
* **SJF** (Shortest Job First): se asigna la CPU al proceso con el menor tiempo de ejecución hasta que dicho proceso finaliza.
  + Puede provocar inanición.
* **SRTF** (Shortest Remaining Time First): si un proceso se está ejecutando y llega uno más corto que lo que queda del actual, se interrumpe y se ejecuta el nuevo proceso. Una vez que se termina, se retoma la ejecución del proceso anteriormente interrumpido.
  + Es muy eficiente
  + Presenta mayor sobrecarga.
  + Puede ser injusta ya que un proceso corto puede echar a uno largo que esté haciendo uso del procesador y que además esté terminando.
* **Round Robin**: existe una porción o quantum de tiempo establecido. Se ejecutan los procesos a medida que llegan a la fila de espera hasta que se termina su ejecución o el quantum se cumple (lo que suceda primero). Si el proceso no finalizó, vuelve a la cola de espera.
* Retroalimentación multinivel
* Planificación por comportamiento
* **Colas múltiples**: los procesos se asignan a diferentes colas, cada una con una planificación diferente, de acuerdo a sus necesidades.

**Criterios a tener en cuenta en un algoritmo de planificación:**

* **Eficiencia**: se refiere al porcentaje de tiempo en que el procesador está siendo utilizado, que debe ser el mayor posible.
* **Tiempo de procesador**: es el tiempo que un proceso está utilizando el procesador sin contar el tiempo que se encuentra bloqueado por operaciones de entrada/salida.
* **Tiempo de respuesta**: es la velocidad con que el ordenador da la respuesta a una petición. Depende mucho de la velocidad de los dispositivos de entrada y salida.
* **Tiempo de servicio**: es el tiempo que tarda en ejecutarse un proceso, donde se incluye el tiempo de carga del programa en memoria, el tiempo de espera en la cola de procesos separados, el tiempo de ejecución en el procesador y el tiempo consumido en operaciones de entrada/salida
* **Rendimiento**: es el número de trabajos o procesos realizados por unidad de tiempo, que debe ser lo mayor posible.
* **Tiempo de ejecución**: es idéntico al tiempo de servicio menos el tiempo de espera en la cola de procesos separados; es decir, es el tiempo teórico que necesitaría el proceso para ser ejecutado si fuera el único presente en el sistema.
* **Tiempo de espera**: es el tiempo que los procesos están activos, pero sin ser ejecutados, es decir, los tiempos de espera en las distintas colas.

**Hilos de ejecución**

Los procesos pueden dividirse en secuencias de tareas o hilos, los cuales son porciones de código que pueden ejecutarse de forma simultánea en cooperación con otros subprocesos. Son divisiones lógicas, no físicas. De esta manera, múltiples hilos pueden existir dentro de un proceso, ejecutándose de forma concurrente, compartiendo recursos y memoria, y aumentando la eficiencia.

Se tarda menos tiempo en crear un hilo nuevo en un proceso ya existente a crear un nuevo proceso.

**Los procesos no comparten recursos cuando se ejecutan, los hilos sí**. Los hilos pueden bloquear recursos y negarle el acceso a otros hilos, por lo que se requiere de una buena sincronización.

Procesadores mononúcleo:

* Capacidad de respuesta menor
* Comportamiento predecible
* No presentan errores que podrían presentarse en los multihilos
* Los problemas de bloqueo de recursos se reducen considerablemente

Procesadores multinúcleo:

* Excelente capacidad de respuesta
* Trabajo en paralelo
* Sincronización compleja
* Comportamiento difícil de predecir

1 - ¿Un script puede estar compuesto por un único comando? VERDADERO

2 - Git es una tecnología desarrollada para rastrear y registrar cambios en cualquier tipo de archivo de computadora, especialmente cuando se trata de archivos utilizados por más de una o dos personas. VERDADERO

3 - Un núcleo no puede trabajar con más de un proceso a la vez. VERDADERO (puede ejecutar diferentes subprocesos de un mismo proceso)

4 - Opción 1: El kernel es una aplicación que corre sobre el sistema operativo.

Opción 2: Es una parte del sistema operativo que asigna o quita recursos de hardware. (CORRECTA)

Opción 3: Es un método...

5 - Pueden coexistir múltiples hilos de un proceso.

Opción 1: Sí, comparten recursos y memoria. (CORRECTA)

Opción 2: Sí, comparten...

Opción 3: No, ...

6 - ¿Quién interactúa en la comunicación por memoria compartida? EL KERNEL, memoria primaria, memoria secundaria