**Funcionamiento de sistema operativo**

**Gestión del procesador por el sistema operativo: conceptos básicos:** un **proceso** es un programa que inicia su ejecución, nace en el momento en que se inicia su ejecución y mueren el momento en que finaliza esa ejecución o en que se cancela por algún motivo. Un **programa** es sencillamente un conjunto de instrucciones almacenadas pasivamente una unidad de disco o en memoria sin embargo un **proceso** es ya un programa activo que está realizando las tareas de procesamiento o tratamiento de la información que indica sus instrucciones.

**Tipos de procesamientos**

* **Procesamiento interactivo:** el usuario interactúa directamente con el computador respondiendo éste de forma inmediata por ejemplo cundo trabajamos con un navegador web, damos la dirección de la página web y el computador de forma casi inmediata nos da la respuesta.
* **Procesamiento por lotes:** en los grandes computadores, o cuando los programas ocupan mucho tiempo de CPU, los trabajos a ejecutar se introducen en disco formando una cola (lote de trabajos) en donde cada uno de ellos espera a que le toque su turno de ejecución. el sistema operativo selecciona del lote de trabajos pendientes el o los programas a ejecutar en cada momento. Esto se denomina procesamiento por lotes o **batch.**

**El Sistema Operativo gestiona la ejecución de procesos en:**

* **Monoprogramacion:**  en la monoprogramación se ubica en la memoria principal un programa (además del S.O) y no se inicia la ejecución de otro programa hasta que finaliza totalmente el que se encuentra en ejecución. por ejemplo, si tenemos la CPU cargado con el sistema operativo, proceso 1 y proceso 2, además de operaciones de entrada y salida como impresora, flash USB, disco. En primer lugar, se ejecuta un módulo del sistema operativo que carga en memoria principal el proceso 1y se inicia la ejecución de este proceso hasta que llega un momento en que realizan una llamada al sistema operativo para realizar una operación de entrada/salida y el sistema operativo lanza a ejecutar en sus programas correspondiente que atiende esa petición de entrada/salida y se efectúa la entrada/salida que en este caso puede ser lectura de disco y mientras se efectúa esta lectura el proceso 1 queda bloqueado y no puede continuar su ejecución entonces la CPU queda sin usarse hasta que acabe la lectura de disco y cuando acabe esa lectura se hace otra llamada al sistema operativo para dar paso a la continuación del proceso 1 que pude terminar su ejecución o ser pausado por otra llamada al sistema operativo para realizar una impresión (dejando sin utilizar la CPU) , cuando acabe la impresión seguirá por terminar la ejecución del proceso 1 y el planificador a largo plazo del sistema operativo consulta en la cola de trabajos si hay otros procesos o trabajos pendientes a ejecutar y da paso al proceso 2 continuando este de forma similar al proceso 1 ya mencionado.

**La monoprogramación tiene varios inconvenientes como:** la memoria casi siempre queda parcialmente ocupada (cabrían más procesos en ella), se desaprovecha la memoria principal ya que se cargan de a un proceso aunque quepan más. Por otra parte, siempre que hay una operación de Entrada/salida la CPU no se utiliza, es decir, se desaprovecha el procesador, también los periféricos se utilizan uno a uno por eso la monoprogramación no es eficiente.

* **Multiprogramación:** el sistema operativo carga en memoria principal(MP) todos los de la cola de trabajo que quepan en ella, aunque pude quedar un mínimo espacio ya que ningún otro proceso cabe en esa partición de memoria. El planificador de trabajos a corto plazo del sistema operativo asigna el procesador sucesivamente a los procesos en memoria de forma que el procesador queda aprovechado al máximo. Un ejemplo sería que tenemos en el procesador el sistema operativo y proceso 1, proceso 2, proceso 3 y en los procesos de entrada/salida tenemos impresora, flash USB, disco. Como en la monoprogramación el sistema operativo da inicio al proceso 1 y antes de terminar su ejecución se hace una llamada al sistema operativo para poder realizar un proceso de entrada/salida en lo cual el proceso 1 queda bloqueado y da inicio a la operación entrada/salida que podría ser lectura de disco pero al quedar libre el procesador hasta que concluya la lectura de disco da paso a la ejecución del proceso 2 que también puede ser interrumpido por una llamada al sistema operativo para una operación de entrada/salida como una impresión el procesador queda libre por lo que comienza con la ejecución del proceso 3 que puede ser interrumpido por un proceso de entrada/salida como la lectura de datos de la memoria USB y el proceso 3 queda bloqueado pero la lectura en disco concluye y se desbloquea el proceso 1 e inicia su ejecución hasta terminar o ser interrumpido por otra llamada al sistema operativo para realizar un proceso de entrada/salida por lo cual seguiría por tratar de terminar la ejecución del proceso 2 que había quedado bloqueado. Como se puede ver, el procesador queda siempre ocupado y no se desaprovecha igual que los periféricos.

**Beneficios de la multiprogramación:**  aprovecha los tiempos muertos del procesador que se tenían con la monoprogramación se aprovecha los tiempos muertos de los periféricos, todos los procesos van avanzando poco a poco. Se dice que se ejecutan **concurrentemente.** Esta multiprogramación se puede denominar multiprogramación clásica. En esta pasa ejecutar otro proceso cuando se bloquea el que este en ejecución se dice que el sistema operativo es **no apropiativo**. Hay otros sistemas de programación más evolucionados y en ellos el sistema operativo puede interrumpir la ejecución de un proceso como por ejemplo cuando llegue a la cola un programa de mayor preferencia en este caso se lo denomina al sistema operativo **apropiativo**. Un tipo de sistema operativo que se utiliza con mucha frecuencia se denomina **de tiempo compartido** (**time sharning**) en estos sistemas operativos a cada proceso se le asigna una ranura de tiempo por ejemplo una ranura de 20ms(milisegundos) cuando pasa ese tiempo se interrumpe su ejecución. La interrupción del primer proceso lanza a ejecutar un módulo planificador a corto plazo que selecciona el siguiente proceso preparado o no bloqueado residentes en la memoria. De esta manera se aprovecha al máximo todos los recursos y da la sensación de que todos los procesos avanzan simultáneamente pero en realidad avanzan concurrentemente. Por ejemplo en el procesador tenemos el sistema operativo, el proceso 1 2 3 4 y tres periféricos de entrada/salida entonces el sistema operativo da inicio al proceso 1 pasa la ranura de tiempo de como ya mencionamos de 20ms y se bloquea para pasar a llamar al sistema operativo que da lugar al inicio del proceso 2 y cuando pasa 20 ms arranca el proceso 3 que se quedara bloqueado al pasar 20 ms y arranca el proceso 4 ,pasa su ranura de tiempo y arranca el proceso 1 así sucesivamente o cuando arranque el proceso 2 pude quedar bloqueado por una llamada al sistema operativo para dar lugar a un proceso de entrada/salida por lo que seguirá con el proceso 3 y así sucesivamente hasta terminar de ejecutar todos los procesos.

Cuando se interrumpe un proceso para ejecuta otro hay que realizar un **cambio de contexto: . Bloque de control de proceso (BCP):**  el sistema operativo por cada proceso mantiene en la memoria principal un BCP, que contiene la información tal como: identificadores (del proceso), copia de los contenidos de la CPU, y estado del proceso.

**. cambio de contexto:**  para interrumpir la ejecución de un proceso para pasar a ejecutar otro, el sistema operativo debe efectuar un cambio de contexto que consiste en salvar en el BCP los contenidos de los registros de procesador, incluyendo palabra de estado, puntero de pila y contador de programa del proceso que se interrumpe. y a continuación restaura en el procesador los valores anteriores del proceso que estaba interrumpido o bloqueado, esto también lo hace desde el BCP.

**Conceptos de hebra o hilo(thread)**

El sistema operativo UNIX introdujo el concepto de hilo que incluyen los sistemas operativos modernos: un proceso puede descomponerse en tareas o hilos diferentes que se pueden ejecutar en el tiempo concurrentemente. Cada hilo o hebra tiene su propio estado,prioridad,BCP,etc. Por ejemplo, la **interfaz de órdenes de Windows:** cada ventana de carpeta lleva asociado un hilo independiente de ejecución (puede estar haciendo concurrentemente 2 copias de ficheros en ventanas diferentes)

**Gestión de la memoria por el sistema operativo**

**Direcciones virtuales y físicas:** un programa maquina es un conjunto ordenado de instrucciones en código máquina. Las direcciones del programa se denominan direcciones virtuales o lógicas o relativas(DV). Por otra parte, los programas se cargan en memoria principal a partir de una determinada posición de memoria, en función de las zonas libres, esta dirección inicial se denomina **dirección base (DB)**. Las direcciones reales donde se carga el programa se denominan **direcciones físicas(DF)**. Se puede obtener con facilidad la **dirección física** (DF)sumando **la dirección base** (DB)más la **dirección virtual** (DV) las transformaciones de direcciones virtuales a físicas se suelen realizar con ayuda de un registro base que se carga previamente con la dirección base que se encuentra en la propia CPU o en unos circuitos específicos denominados **unidad de gestión de memoria(MMU)**. En la actualidad, la MMU suele incluirse en el mismo chip del microprocesador

En **multiprogramación** el sistema operativo asigna espacio de memoria a los programas o procesos, de acuerdo con las zonas libres de memoria existente en cada momento. Dependiendo de las posibilidades del sistema operativo, la asignación puede realizarse de distintas formas:

* **Particiones estáticas:** la memoria se divide en ciertos números de particiones o zonas, cada una de las cuales contendrá un proceso. La dirección base de cada proceso es la dirección de inicio de cada partición. El tamaño de las particiones que puede ser no uniforme es un parámetro del sistema operativo. El sistema operativo mantiene una **tabla de particiones o procesos** en la que cada fila corresponde a una partición, conteniendo información tal como: estado(listo , activo o bloqueado), si está libre o no (0 indica que está ocupado el proceso y 1 indica que esta libre) y luego contiene el BCP para transformar la direcciones virtuales a físicas en cada acceso de memoria la MMU consulta la **tabla** **de particiones** para saber la partición en que se encuentra el proceso y calcular su dirección base. El **planificador de trabajo** una vez que una partición está libre, hace que se introduzca el programa de máxima prioridad que haya en la cola de espera.
* **Particiones dinámicas:** los programas son introducidos por el sistema operativo inicialmente en posiciones consecutivas de memoria, no existiendo por tanto particiones predefinidas. La **gestión de particiones dinámicas** se realiza por medio de una **tabla de procesos** (similar a la tabla de particiones estáticas) en la que cada línea contiene la identificación del proceso, la dirección base, capacidad que ocupa, estado, dirección de su BCP. El sistema operativo también gestiona una **tabla de fragmentos libres** que el **planificado de trabajo** consulta para introducir en memoria principal los fragmentos que quepan en los fragmentos libres y actualiza las dos tablas (tabla de proceso y tabla de fragmentos libre).
* **Reubicación de procesos:** cuando acaba un proceso deja un fragmento libre que solo puede ser ocupado otro igual o capacidad menor. Al ir acabando de ejecutarse los programas, el número de fragmentos libres crecerá llegado un momento en que el porcentaje de memoria aprovechado es muy reducido porque los procesos ubicados en memoria de forma consecutiva por ejemplo proceso1,23,34,52,77 al ir terminando no lo hacen de forma consecutiva y quedan espacios o fragmentos libres por partes distintas de la memoria y puede que no haya ningún programa de esa capacidad es decir que sea mayor a los fragmentos libres que hay y que solo podría entrar si se juntaros esos espacios libres de forma consecutiva. Pero este problema se resuelve haciendo una compactación que agrupa a todos los fragmentos libres, reubicando los programas en ejecución y una vez unidos todos los fragmentos libres, existe la posibilidad de introducir nuevos procesos.
* **Segmentación:**  la memoria se aprovecha mejor cuando más pequeños fuesen los procesos o hilos o unidades a almacenar en ellas. La **gestión de memoria con segmentación** consiste en considerar como unidades de almacenamiento **segmentos**, en lugar de procesos o hilos. Los segmentos tienen menor capacidad que los procesos o hilos que los contienen. La segmentación permite que ciertos procesos puedan compartir código (rutinas, etc.) o datos comunes sin necesidad de estar duplicados en memoria principal (estos segmentos se denominan segmentos reentrantes). La **gestión de la segmentación** la realiza el sistema operativo, como las particiones dinámicas, solo que cada partición en vez de corresponder a un programa completo corresponde a un segmento, de esta manera el sistema operativo mantiene por cada proceso una **tabla de segmentos** donde se especifica su capacidad y la dirección base de cada segmento. Para transformar las direcciones virtuales a físicas en cada acceso a memoria la MMU consulta la tabla de segmentos para ver si la dirección sobrepasa sobrepasa el límite y para obtener la dirección base correspondiente.
* **Paginación:** los programas se consideran divididos en zonas consecutivas o **páginas**(procesos)**.** Cada página es de tamaño fijo, preestablecido como un parámetro del sistema operativo usualmente los tamaños que se suelen seleccionar van de 512B a 64KB por lo general. Cada página se identifica dentro del programa con un numero correlativo (0,1,2,3,4). La memoria principal se estructura en **marcos de páginas** (memoria principal) de igual capacidad que las páginas del programa ya que cada marco de página va a albergar una página de programa. Cada marco de páginas al igual que ocurre con las páginas también se identifican con un numero correlativo (00,01,02,03, etc.). **gestión de la paginación:** el fundamento de la paginación está en que no es necesario que el programa completo se almacene en posiciones consecutivas de memoria sino que las páginas se almacenan en marcos de páginas libres, independientemente de que estén o no contiguos. El sistema operativo mantiene por cada proceso una **tabla de página** que indica el marco de página donde se encuentra cada página del proceso.
* **Memoria virtual:** presenta una memoria principal aparente mayor a que la memoria física real y permite ejecutar programas de capacidad superior a la memoria que físicamente tiene el computador y por otra parte permite también aumentar el número de procesos en la memoria principal en la **implementación de la memoria virtual** en primer lugar se mantiene en disco el proceso completo, que se considera por la CPU dividido en páginas. Únicamente se cargan en la memoria principal las páginas que en ese momento intervengan en el proceso una decisión que tiene que tomar el sistema operativo es decidir que paginas deben estar en memoria y hay dos criterios el primero es el **criterio de carga memoria** que decide que páginas se cargan en memoria el otro criterio es: **criterio de remplazo de página** que considera que páginas se deben sacar de memoria principal para introducir otra y que se hace mediante algoritmos de sustitución de páginas conocidos como FIFO,NRU,algoritmo de reloj,etc.