**3.1 CONCEPTO DE PROCESO:**

* Los sistemas operativos manejan diversas actividades de la CPU, como ejecutar trabajos en lotes o programas de usuario.
* Los procesos son programas en ejecución que incluyen el código del programa, la actividad actual (representada por el contador de programa y los registros del procesador), la pila del proceso y la sección de datos.
* Un programa se convierte en un proceso cuando se carga en memoria.
* Pueden existir múltiples procesos asociados con el mismo programa, cada uno con su propio conjunto de recursos.
* Un proceso puede cambiar de estado a medida que se ejecuta, incluyendo nuevo, en ejecución, en espera, preparado y terminado.

**3.1.1 El proceso:**

* Un proceso es más que simplemente el código de un programa; también incluye actividad actual, pila del proceso, sección de datos y memoria asignada dinámicamente.
* Un proceso se activa cuando se carga un archivo ejecutable en memoria.
* Pueden existir múltiples procesos asociados con el mismo programa, cada uno con su propio conjunto de recursos y estado de ejecución.

**3.1.2 Estado del proceso:**

* Un proceso puede estar en diferentes estados, como nuevo, en ejecución, en espera, preparado y terminado.
* Estos estados representan la actividad actual del proceso.
* Cada proceso tiene un único estado en un momento dado, pero puede haber múltiples procesos preparados y en espera simultáneamente.

**3.1.3 Bloque de control de proceso (PCB):**

* Cada proceso se representa mediante un PCB que contiene información como el estado del proceso, contador de programa, registros de la CPU, información de planificación de la CPU, gestión de memoria, información contable y estado de E/S.
* El PCB sirve como repositorio de información específica del proceso que puede variar entre procesos.

**3.1.4 Hebras:**

* Inicialmente, un proceso tenía una sola hebra de ejecución.
* En los sistemas operativos modernos, un proceso puede tener múltiples hebras de ejecución, lo que le permite realizar varias tareas simultáneamente.
* Esto amplía las capacidades de los procesos para realizar múltiples actividades al mismo tiempo.

**3.2 PLANIFICACIÓN DE PROCESOS:**

* La multiprogramación tiene como objetivo ejecutar varios procesos simultáneamente para maximizar la utilización de la CPU.
* Los sistemas de tiempo compartido alternan la CPU entre diferentes procesos para permitir la interacción de los usuarios con los programas en ejecución.
* El planificador de procesos selecciona un proceso disponible para ejecutar en la CPU, asegurando que solo un proceso se ejecute a la vez en sistemas monoprocesador.

**3.2.1 Colas de planificación:**

* Los procesos se colocan en una cola de trabajo a medida que entran en el sistema, y los preparados se mantienen en una cola de procesos preparados.
* Existen colas adicionales para procesos en espera de E/S, cada dispositivo tiene su propia cola.
* Un diagrama de colas puede representar las relaciones entre las colas de trabajo y los recursos que atienden a estas colas.

**3.2.2 Planificadores:**

* Los procesos se mueven entre diferentes colas de planificación durante su tiempo de vida.
* Los planificadores a largo plazo y a corto plazo seleccionan procesos para la ejecución en la CPU.
* El planificador a largo plazo controla el grado de multiprogramación y puede seleccionar procesos basándose en su limitación de E/S o de CPU.

**3.2.3 Cambio de contexto:**

* Las interrupciones causan cambios en el contexto de la CPU, lo que obliga al sistema operativo a cambiar entre distintos procesos.
* El cambio de contexto implica guardar el estado del proceso actual y restaurar el estado de otro proceso.
* El tiempo dedicado al cambio de contexto es tiempo desperdiciado, pero su velocidad varía según el hardware y el sistema operativo utilizado.

**3.3 OPERACIONES SOBRE LOS PROCESOS:**

En la mayoría de los sistemas operativos, los procesos pueden ejecutarse concurrentemente y crearse y eliminarse dinámicamente. Se requiere un mecanismo para la creación y terminación de procesos.

**3.3.1 Creación de procesos:**

* Los procesos pueden crear otros procesos nuevos mediante una llamada al sistema específica.
* El proceso creador se denomina proceso padre, y los nuevos procesos son sus hijos, creando así un árbol de procesos.
* Cada proceso es identificado por un identificador de proceso (PID), que es un número entero único.
* Se puede obtener un listado de procesos en UNIX mediante el comando ps.
* Un proceso necesita recursos para sus tareas, y cuando crea un subproceso, este puede obtener recursos del sistema operativo o del proceso padre.
* Un proceso padre puede pasar datos de inicialización al proceso hijo.
* Hay dos posibilidades en términos de ejecución y espacio de direcciones del nuevo proceso: continuar ejecutándose concurrentemente con el padre o esperar a que los hijos se completen, y ser un duplicado del proceso padre o cargar un nuevo programa.
* En UNIX, se puede crear un proceso hijo con la llamada al sistema fork() y reemplazar su espacio de direcciones con un nuevo programa usando la llamada al sistema exec().
* En Windows, los procesos se crean con la función CreateProcess(), que carga un programa específico en el espacio de direcciones del proceso hijo durante su creación.

**3.3.2 Terminación de procesos:**

* Un proceso termina ejecutando su última instrucción y pidiendo al sistema operativo que lo elimine con la llamada al sistema exit().
* Un proceso puede terminar otro proceso mediante la llamada al sistema TerminateProcess() en Windows.
* Un padre puede terminar la ejecución de uno de sus hijos por diversas razones, como el exceso de uso de recursos o la finalización de la tarea asignada.
* Algunos sistemas, como VMS, no permiten que un hijo continúe existiendo si su proceso padre se ha completado, lo que se conoce como terminación en cascada.

**VIDEO:**

* La CPU, un recurso crítico en los sistemas informáticos, enfrenta el desafío de manejar múltiples procesos simultáneamente, ya que típicamente hay más procesos que CPUs disponibles.
* La gestión eficiente de la CPU se vuelve crucial en entornos con múltiples procesos compitiendo por su utilización.
* Para abordar esta problemática, se introduce el concepto de multiprogramación, que permite alternar la ejecución de varios procesos en una sola CPU, lo que da la impresión de paralelismo.
* El cambio de contexto es esencial en la multiprogramación, ya que implica la transición entre la ejecución de diferentes procesos, donde la información del proceso en ejecución se guarda y se carga la del siguiente proceso.
* En entornos de multiprocesadores reales, existe la posibilidad de un verdadero paralelismo, donde varios núcleos de CPU pueden ejecutar procesos simultáneamente.
* La gestión de la memoria también es un aspecto crítico en la ejecución de múltiples procesos, ya que cada proceso consume recursos de memoria y necesita un espacio adecuado para su ejecución.
* Los procesos se crean mediante llamadas al sistema, que pueden ocurrir al inicio del sistema, de manera interactiva o en segundo plano, o como respuesta a ciertas acciones del usuario.
* Se utilizan llamadas al sistema específicas, como fork y exec en entornos Unix, para crear y ejecutar nuevos procesos.
* El proceso de creación de procesos implica la duplicación de un proceso existente (fork) seguido de la asignación de un nuevo programa para ejecutar (exec).
* Los procesos pueden finalizar voluntariamente mediante llamadas al sistema, como exit, o involuntariamente debido a errores o señales de terminación del sistema operativo.