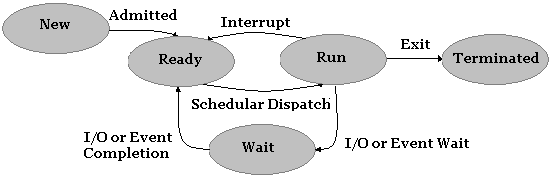
**Tutorial 25**

Ahora que podemos ejecuta procesos en memoria, ya podemos empezar a planificar poder cargar mas procesos a memoria y poder manejarlos independientemente. Para esto, será necesario implementar mas funciones a nuestro sistema operativo, como semáforos, mutexes, pipes y un planificador de procesos. Un proceso puede tener 4 estados: Nuevo, En Ejecucion,Listo, Bloqueado y Terminado.



Sin embargo, a la hora de ejecutar múltiples procesos en un sistema operativo nos llegamos a encontrar con varios problemas. Un problema muy común se llama acceso al área crítica. Esto es cuando dos o más procesos comparten un nombre de variable en memoria por ejemplo o desde el punto de vista del CPU, tiene información guardad en los registros (EAX, EBX, ECX, EDX, etc). Cuando el proceso A guarda una información en el registro EAX por ejemplo y luego el planificador quita el proceso A del CPU y pone el B y este viene y cambia este registro y luego el planificador quita el proceso B y pone nuevamente el proceso A, ya el registro EAX no será el mismo que el proceso A coloco, pudiendo a continuación dar un problema al comparar el valor de la variable con algo no esperado por ejemplo. Para solucionar este problema de área critica, los sistemas implementan algo que se llama mutex o exclusión mutua, que significa que un proceso puede tomar un recurso y bloquearlo para que los demás procesos no tomen control sobre ese recurso compartido hasta que el proceso lo haya dejado de utilizar y este libre para que otro proceso haga uso de el. También debemos asegurarnos que las instrucciones de bloqueo sean atómicas, que se hagan en una sola operación y esta operación no pueda ser interrumpida por otro proceso.

Otra cosa necesaria para poder controlar los procesos es poder implementar IPC o Inter Process Communication. Esto se puede hacer mediante “pipes” nombrados para pasar información entre varios procesos, implementar paso de mensajes entre procesos para enviar y recibir data y también memoria compartida entre procesos.

Luego de tener todos estos conceptos listos e implementados en nuestros programas de usuario y kernel, es necesario poder definir un algoritmo de planificación de procesos que evite la inanición de los programas al requerir tiempo de ejecución en el procesador. Existen varios algoritmos como por ejemplo: el primero proceso en aparecer es el primer proceso puesto a ejecutarse hasta que termine pero esto deja a los demás procesos en un estado de inanición que puede resultar con un tiempo variable por lo cual no es muy conveniente. También existen planificadores por colas de prioridades, sin embargo, una cola de procesos de prioridad alta puede o no que deje a procesos en una cola de prioridad mas baja caer en inanición si pasan mucho en el CPU. El mas seguro de todos es poder implementar un planificador usando Round Robin como algoritmo de cambio y que de a cada proceso un quantum de tiempo en CPU. Esto asegura que todos los procesos se ejecuten en un tiempo definido y así poder ejecutarlos a todos y evitar la inanición.