**Gestión de procesos:**

**Proceso:** Programa en ejecución. Consta de imagen de memoria que consta de código ejecutable, variables, stack, heap y librerías dinam, las estáticas son parte del texto.

Necesita: Registros de la CPU(PC,SP,IR) se guardan en la BCP (Se conoce como estado del procesador),Entorno(datos globales), Metadatos datos del proceso que produce el SO están en el Kernel(BCP(guarda identidad del proceso), tabla de paginas, mapa E/S(almacena info sobre descripciones entradfa/salida)), el disco donde se guarda el código ejecutable.

-Tienen dos stack uno en zona del user (stack de proceso)y otro en el kernel(stack de kernel) en Intel. Esto hace no tener que hacer cambio de contexto

Ciclo de vida de un proceso

**Creación**:

Se crea por petición de otro proceso. PARTES NECESARIAS:Asignar imagen de memoria, Seleccionar un BCP libre en la tabla del Kernel, cargar el ejecutable y los datos estáticos en la memoria del proceso, crear el stack y el heap.

Mientras esto dure el proceso esta en estado nuevo y cuando termina pasa a estado listo.

**Activación:**

Consiste en poner el proceso en ejecución y de ello se ocupa el **Dispatcher**(comp del Kernel). Para activarlo se recuperan el estado del procesador que están almacenados en la BCP. El Kernel entra en ejecución en modo supervisor, guarda el contexto del proceso que sale y rescata el nuevo de la BCP del proceso seleccionado y cambia el PC. **Todo este proceso se llama cambio de contexto(Muy costoso en tiempo). Con memoria virtual la tabla de paginas cambia y se recarga la TLB.**

Interrupción:

La interrupción es una fase crítica ya que activa el Kernel y pone la CPU en modo supervisor(por evento de HW o int SW). El Kernel primero trata la interrupción, salva los registros del procesador q ve el user en el stack del user del proceso en ejecución. El Kernel tras tratar la interrupción puede bloquear el proceso debido a la interrupción o expulsarlo de la cola(estado listo) y el scheduler decide que proceso entra a la CPU O puede recuperar el contexto del proceso, invocar la instrucción de retorno de interrupción y se regresa al entorno de user en el punto de antes de la interrupción.

Finalización:

El proceso acaba cuando dentro de el se termina (return() o excepción división por 0, etc), pero no deja de existir sino que se debe liberar la BCP o l tabla de memoria, esto se dice q poasa a aestado Zombie.

--No confundir cambio de contexto con paso de user a supervisor(interrupción y levantar al Kernel).La primera conlleva la segunda pero la segunda no siempre viene dada por la primera. En XV6 se define la structura trapframe que almacena el estado del procesador en un cambio de contexto y en una interrupción.

**Interrupción:**

Diagrama

Descripción generada automáticamenteEn las interrupciones la atención se divide en dos partes, la primera (Top Half) el Kernel enmascara todas las demás interrupciones y reconoce la actual, programa un tasklet, forma especial de código que se encola en el propo Kernel para su ejecución posterior en el Bottom Half la mayor parte del código esta en el tasklet q si se puede interrumpir por si hay que tratar interrupciones más importantes.

Cosas q intervienen en interrupción en IntelX86:

-HW genera interrupción-Circuito PIC8259A(recibe las interrupciones y dice cual es la mas importante)-CPU salva el contexto, bloquea las demás, reconoce la actual obtiene el vector de interr y desbloquea las demás-ejecuta la rutina de interrupción y al acabar envia una señal EOI al PIC8259A para indicarlo-restaura contexto.

Interrupción de SW: Sirve para invocar los servicios del SO. Registros si 6omenos parámetros (%eax, %ebx, %ecx,,%esi, %edi, %ebp) 6omas parámetros se pasa puntero donde están los valores ambas devuelven %eax.

**Procesos especiales:**

Proceso Servidor: esta en bucle infinito pendiente de recibir peticiones y atenderlas.

Demonio: arranca al inicio del sistema, si acaba de forma anormal el sistema lo vuelve a arrancar(inmortal). Se ejecutan en el background.

Proceso de sistema que se ejecutan como modo privilegiado. Crean un servicio especial durante el arranque, tiene acceso a todo el mapa de memoria y son flujos de ejecución independientes dentro del SO.

**Threads:**

Llamados procesos ligeros. Los threads de un mismo proceso comparten código, datos (heap incluido) y recursos del SO y tienen su propio PC, registros y stack.Puede haber conflictos al acceder a los mismos datos desde dos threads diferentres. Cambio de contexto de thread es más rápido q el de un proceso. Solo se lava el contexto del thread.

Planificación:

El tiempo de ejecución es independiente del planificador. Planif a largo plazo: decide en que orden se ejecutan las tareas en un gran sistema(batch).Planif a medio sirve para decidir que procesos van a bloquearse. Planif a corto decide que proceso listo entra en la CPU para ejecutarse. El procedimiento para cargar un nuevo proceso en la CPU se llama dispatching.

KPI:

T ejecución, T espera(tiempo q el proceso esta listo esperando a entrar en la CPU), T respuesta(T desde q el proc esta listo hasta que responde a una petición del user), penalización(t res/tej).

Tick son interrupciones periódicas en la CPU para gestionar los procesos. Quantum es el tiempo máximo que puede ejecutarse un proceso de forma continua habiendo otros en la cola.

En los sistemas no expulsivos, el proceso completa su quantum con independencia de los que estén esperando en cola. En los expulsivos, el proceso puede ser expulsado aunque no hay

terminado su quantum si hay a la espera otro de mayor prioridad.

Algoritmos de planif

FCFS (FirstComeFirstServed)—Concede la CPU al q mas tiempo lleva en la cola y se ejecuta hasta terminar no quantum.

RoundRobin:--Hay quantum, loq significa q funciona como un FCFS pero al acabar el quantum va al final de la cola.

Prioridad: Starvation(CPU se consume por los de mayo prioridad y los de menos no se ej tanto)

Linux O(1): primer planif de linux, con diferentes colas de prioridad. Tiempo real (0-99)y procesos normales(100-139)

Linux CFS: En vez de cola de prioridad utiliza un árbol de prioridades.

Xv6 tiene un RoundRobin sin prio y con quantum de 100ms

BSD (descendiente de Unix) tiene el algoritmo ULE

Tanto Unix como Windows usan elthread como unidad de planificación

**operación de expulsar al proceso que ocupa la CPU y concedérsela a otro🡪Activación**