Explicación práctica 4

lunes, 9 de octubre de 2023 15:45

Proceso → parte de programa + grande

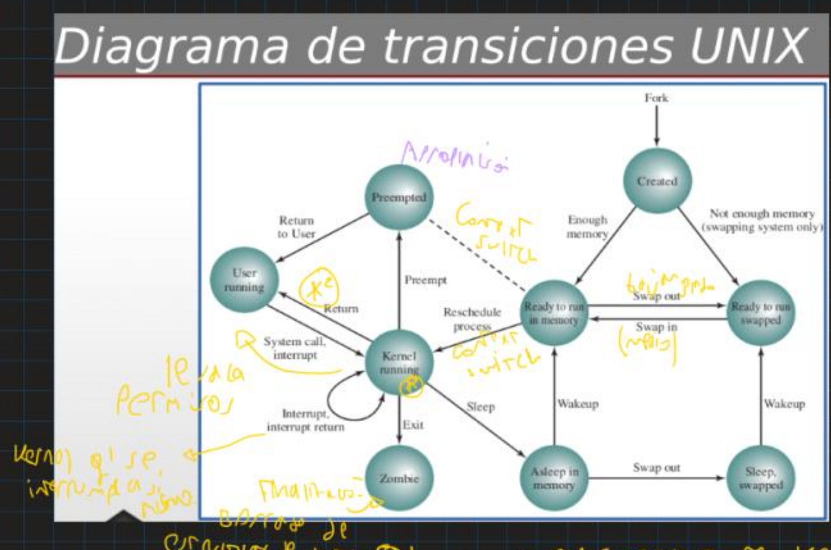
Ligados a CPU → ejecutan solo ráfagas del CPU

E/S Bound → combinan ráfagas de CPU con E/S.

Programa
- → estático
- → Reside en disco
- → No tiene PC
- → Existe desde q se ejecuta hasta borrado.

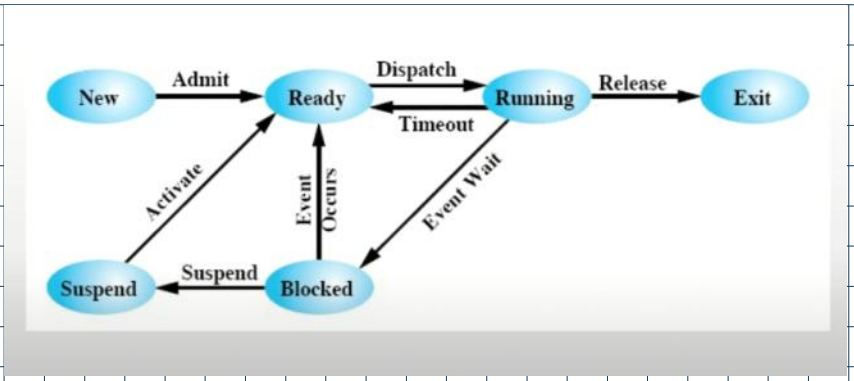
Proceso
- → dinámico
- → tiene PC (solo puede estar y reanudar dsp)
- → Ciclo de vida es desde q se ejecuta hasta fin.

PCB
- → 1 x proceso
- → primero en crearse cuando se hace un fork, último en borrarse



⊗ Arranca a ejecutarse en modo kernel porque tiene q' hacer context switch. cuando termina de cargar todo pasa a modo user y se ejecutan instrucciones del proceso.

(x²) Devuelve a kernel x interrupciones.



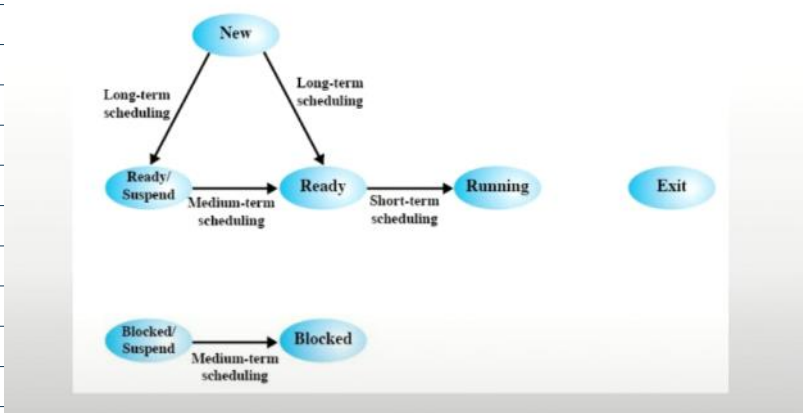
planificación → decidir según los recursos disponibles qué procesos se pueden cargar en RAM y cuáles se ejecutan y en qué orden

- – tiempo de respuesta
- + rendimiento
- uso eficiente

Long term → Admite a mem (controla grado de multiprogram)
De estado nuevo a Ram

medium term → Manda a swap entre disco y mem.
– grado de multip.

Short term → qué proceso se va a ejecutar de ready a running



Tiempos de los procesos

- **Retorno**: tiempo que transcurre entre que el proceso llega al sistema hasta que completa su ejecución
- **Espera**: tiempo que el proceso se encuentra en el sistema esperando, es decir el tiempo que pasa sin ejecutarse (**TR - Tcpu**)
- **Promedios**: tiempos promedio de los anteriores

Planificadores → Apropiativos → Cuando proceso se empieza a ejecutar no lo para nadie (salvo cuando sale el proceso mismo)

Apropiativos → Proceso puede ser interrumpido y llevado a reposo
- Hay mucho context switch
- No hay starvation.

FIFO

- First come first served
- Cuando hay que elegir un proceso para ejecutar, se selecciona el mas viejo
- No favorece a ningún tipo de procesos, pero en principio prodíamos decir que los *CPU Bound* terminan al comenzar su primer ráfaga, mientras que los *I/O Bound* no

SJF

- Shortest Job First
- Política *nonpreemptive* que selecciona el proceso con la ráfaga más corto
- Calculo basado en la ejecución previa
- Procesos cortos se colocan delante de procesos largos
- Los procesos largos pueden sufrir *starvation* (inanición)

RR

- Round Robin
- Politica basada en un reloj
- **Quantum (Q)**: medida que determina cuanto tiempo podrá usar el procesador cada proceso:
  - Pequeño: overhead de *context switch*
  - Grande: Aumenta → FIFO
- Cuando un proceso es expulsado de la *CPU* es colocado al final de la *Ready Queue* y se selecciona otro (*FIFO circular*)

- Existe un "contador" que indica las unidades de CPU en las que el proceso se ejecuto. Cuando el mismo llega a 0 el proceso es expulsado
- El "contador" puede ser:
  - Global
  - Local → *PCB*
- Existen dos variantes con respecto al valor inicial del "contador" cuando un proceso es asignado a la CPU:
  - **Timer Variable**
  - **Timer Fijo**

- Cada proceso tiene un valor que representa su prioridad → menor valor, mayor prioridad
- Se selecciona el proceso de mayor prioridad de los que se encuentran en la *Ready Queue*
- Existe una *Ready Queue* por cada nivel de prioridad
- Procesos de baja prioridad pueden sufrir *starvation* (inanición)
  - Solución: permitir a un proceso cambiar su prioridad durante su ciclo de vida → **Aging o Penalty**
- Puede ser un algoritmo **preemptive** o no

- Shortest Remaining Time First
- Versión *preemptive* de *SJF*
- Selecciona el proceso al cual le resta menos tiempo de ejecución en su siguiente ráfaga.
- ¿A qué tipos de procesos favorece?

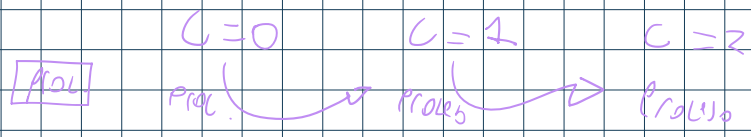
Algoritmos de planificación - CPU + I/O

- Ciclo de vida de un proceso: uso de CPU + operaciones de I/O
- Cada dispositivo tiene su cola de procesos en espera → un scheduler por cada cola
- Se considera I/O independiente de la CPU (DMA, PCI, etc.) → uso de CPU y operaciones de I/O en simultaneo

- Orden de aplicación:
  - Orden de llegada de los procesos
  - **PID** de los procesos
- Siempre se mantiene la misma politica

Actual mente

- Schedulers actuales → combinación de algoritmos vistos
- La *ready queue* es dividida en varias colas (similar a prioridades)
- Los procesos se colocan en las colas según una clasificación que realice el sistema operativo
- Cada cola posee su propio algoritmo de planificación → **planificador horizontal**
- A su vez existe un algoritmo que planifica las colas → **planificador vertical**
- Retroalimentacion → un proceso puede cambiar de una cola a la otra




```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void) {
        int c;
        pid_t pid;
        printf("Comienzo.:\n");
        for (c = 0; c < 3 ; c++ )
        {
                pid = fork();
        }
        printf("Proceso\n");
        return 0;
}
```

Veces son la palabra "Proceso" aparecen al final de la sig