2ºDAM - Programación de servicios y procesos

Tema 2 – Programación multiproceso

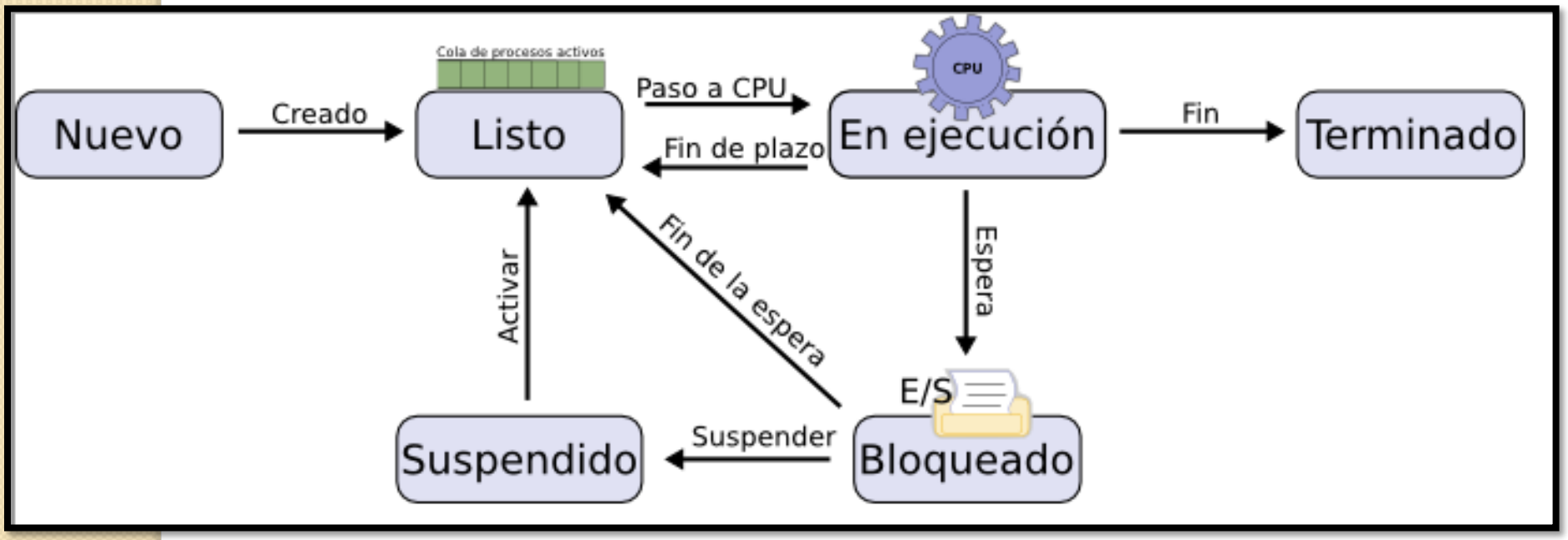
2. Elementos de un proceso

Recordemos los **estados** en el ciclo de vida de un **proceso**:

- **Nuevo**. Proceso nuevo, creado.
- **Listo**. Proceso que está esperando la CPU para ejecutar sus instrucciones.
- **En ejecución.** Proceso que actualmente, está en turno de ejecución en la CPU.
- **Bloqueado**. Proceso que está a la espera de que finalice una E/S.
- **Suspendido.** Proceso que se ha llevado a la memoria virtual para liberar, un poco, la RAM del sistema.
- **Terminado.** Proceso que ha finalizado y ya no necesitará más la CPU.

2.1. Cambios de estados entre procesos

El siguiente gráfico, nos muestra las distintas transiciones que se producen entre uno u otro estado:



Transiciones que se producen entre uno u otro estado de los procesos

2.3. Planificación de procesos por el SO

Hemos visto que un proceso, desde su creación hasta su fin (durante su vida), pasa por muchos estados. **Esa transición de estados, es transparente para el proceso, todo lo realiza el SO**. Desde el punto de vista de un proceso, él siempre se está ejecutando en la CPU sin esperas. Dentro de la gestión de procesos vamos a destacar dos componentes del SO que llevan a cabo toda la tarea: **el cargador y el planificador**.

El cargador es el encargado de crear los procesos.

2.3. Planificación de procesos por el SO

Cuando se inicia un proceso, el **cargador**, realiza las siguientes tareas:

1. **Carga el proceso en memoria principal**. Reserva un espacio en la RAM para el proceso. En ese espacio, copia las instrucciones del fichero ejecutable de la aplicación, las constantes y deja un espacio para los datos (variables) y la pila (llamadas a funciones).

   Un proceso, durante su ejecución, no podrá hacer referencia a direcciones que se encuentren fuera de su espacio de memoria; si lo intentara, el SO lo detectará y generará una excepción (produciendo, por ejemplo, los típicos pantallazos azules de Windows).

2.3. Planificación de procesos por el SO

2. Crea una estructura de información llamada **PCB** (Bloque de Control de Proceso). La información del PCB, es única para cada proceso y permite controlarlo. Esta información, también la utilizará el **planificador**. Entre otros datos, el PCB estará formado por:

- **Identificador del proceso (PID)**. Es un número único para cada proceso, como un DNI de proceso.
- **Estado actual del proceso**: en ejecución, listo, bloqueado, suspendido, finalizando.
- **Espacio de direcciones de memoria** donde comienza la zona de memoria reservada al proceso y su tamaño.

2.3. Planificación de procesos por el SO

- **Información para la planificación**: prioridad, *quantum*, estadísticas, ...

- **Información para el cambio de contexto**: valor de los registros de la CPU, entre ellos el contador de programa y el puntero a pila. Esta información es necesaria para poder cambiar de la ejecución de un proceso a otro.

2.3. Planificación de procesos por el SO

Una vez que el proceso ya está cargado en memoria, será el **planificador** el encargado de **decidir qué proceso se ejecuta y cuánto tiempo se ejecuta**.

El planificador es otro proceso que es parte del SO. La política en la toma de decisiones del planificador se denominan **algoritmo de planificación**.

2.3. Planificación de procesos por el SO

Algoritmos de planificación principales.

Términos a considerar:

- **Tiempo de retorno (Tq):**
  - Tiempo que está un proceso en el sistema.
  - Instante final (Tf) menos instante inicial (Ti).
  - Objetivo: Minimizar.
- **Tiempo de servicio (Ts):**
  - Tiempo dedicado a tareas productivas (CPU, E/S).
  - Ts = T(CPU)+ T(E/S)
- **Tiempo de espera (Te):**
  - Tiempo que un proceso pasa en colas de espera.
  - Te = Tq – Ts
- **Tiempo de retorno normalizado (Tn):**
  - Razón entre tiempo de retorno y tiempo de servicio.
  - Tn = Tq/Ts
  - Indica el retardo experimentado.

2.3.1. Planificación de procesos por el SO Round Robin

Algoritmos de planificación principales:

- **Round-Robin.** Este algoritmo de planificación favorece la ejecución de procesos interactivos. Es aquél en el que cada proceso puede ejecutar sus instrucciones en la CPU durante un quantum. Si no le ha dado tiempo a finalizar en ese quantum, se coloca al final de la cola de procesos listos, y espera a que vuelva su turno de procesamiento. Así, todos los procesos listos en el sistema van ejecutándose poco a poco. La cola de procesos activos (listos) se gestiona con una política FIFO.

2.3.1. Planificación de procesos por el SO Round Robin

- **Ejemplo de planificación round robin**

Tenemos 3 procesos que solicitan **su entrada a la vez**:

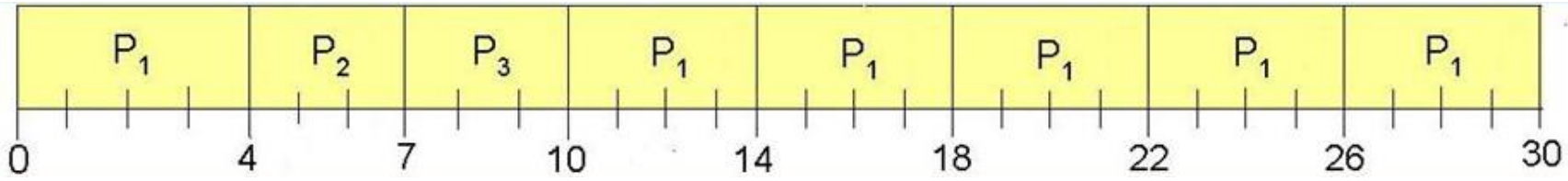
Proceso	Ciclo de CPU
P_1	24
P_2	3
P_3	3

quantum = 4

2.3.1. Planificación de procesos por el SO Round Robin

- Llega el proceso P1,P2,P3 y se *encolan* en orden de llegada. El P1 es el primero en entrar a la CPU, se procesa durante 4 ciclos y vuelve a encolarse. Entonces el siguiente proceso que estaba encolado, P2, entra en la CPU, consume los 3 ciclos que necesita y termina dando paso al P3 que realiza sus 3 ciclos y por último vuelve a procesarse el P1 consumiendo 4 ciclos hasta completar los 24 que necesitaba. Es decir, al finalizar un quantum, el proceso es retirado del CPU y colocado al final de la cola de procesos listos.

Este sería el diagrama de Gantt:



2.3.1. Planificación de procesos por el SO Round Robin

- **Ejemplo de planificación Round-Robin:**
  quantum=2

Procesos	Tiempo de llegada	Tiempo de ejecución (Servicio)
A	0	5
B	1	3
C	2	1

2.3.1. Planificación de procesos por el SO Round Robin

- **Ejemplo de planificación Round-Robin:**
  quantum=2

Procesos/ tiempo	0	1	2	3	4	5	6	7	8	9
A	x	x				x	x		x	
B			x	x				x		
C					x					

Proc/ Tiempo	Llegada	Servicio	Inicio	Fin	Retorno (Fin-Llegada)	Espera (Retorno-Servicio)	Normalizado (Retorno/ Servicio)
A	0	5	0	9	9	4	1,8
B	1	3	2	8	7	4	2,3
C	2	1	4	5	3	2	3

Nosotros utilizaremos esta representación

2.3.2. Planificación de procesos por el SO FCFS

Algoritmos de planificación principales:

- **FCFS.** FCFS (First-Come, First-Served) al igual que Round-Robin, es un algoritmo no apropiativo. Atiende a los procesos por orden de llegada a la cola de procesos activos. A diferencia de Round-Robin, cada proceso se ejecuta hasta que termina, o hasta que hace una llamada de E/S y es interrumpida su ejecución.
  Es un algoritmo que penaliza a los procesos cortos y el tiempo promedio de espera suele ser largo.

2.3.1. Planificación de procesos por el SO FCFS

- **Ejemplo de planificación FCFS:**

Procesos	Tiempo de llegada	Tiempo de ejecución (Servicio)
A	0	5
B	1	3
C	2	1

2.3.2. Planificación de procesos por el SO FCFS

- **Ejemplo de planificación FCFS:**

Procesos/ tiempo	0	1	2	3	4	5	6	7	8	9
A	x	x	x	x	x					
B						x	x	x		
C									x	

Proc/ Tiempo	Llegada	Servicio	Inicio	Fin	Retorno (Fin-Llegada)	Espera (Retorno-Servicio)	Normalizado (Retorno/ Servicio)
A	0	5	0	5	5	0	1
B	1	3	5	8	7	4	2,3
C	2	1	8	9	7	6	7

Nosotros utilizaremos esta representación

2.3.3. Planificación de procesos por el SO Prioridad

- **Por prioridad.** En el caso de *Round-Robin*, todos los procesos son tratados por igual. Pero existen procesos importantes que no deberían esperar a recibir su tiempo de procesamiento a que finalicen otros procesos de menor importancia. En este algoritmo, se asignan prioridades a los distintos procesos y la ejecución de estos se hace de acuerdo a esa prioridad asignada.

2.3.3. Planificación de procesos por el SO Prioridad

- Tenemos varios tipos de algoritmos que se basan en la prioridad de los procesos. También hay que tener en cuenta las siguientes características que puede tener un algoritmo:

•**Apropiativo:** También conocido como **expulsivo**, este tipo de algoritmo nos permite la expulsión de procesos para ejecutar un nuevo proceso, poniendo en la cola de procesos activos al anterior.

2.3.3. Planificación de procesos por el SO Prioridad

- **No Apropiativo:** Este tipo no nos permite la expulsión, por lo que un proceso nuevo no entrará hasta que termine el anterior.

En caso de **empate en prioridad** a la hora de tener que elegir un proceso para ejecutarse en la CPU seguiremos el algoritmo **FCFS**.

Este tipo de algoritmos pueden dar problemas de inanición.

2.3.3. Planificación de procesos por el SO Prioridad (apropiativo)

- **Ejemplo de planificación con prioridad (apropiativo)**

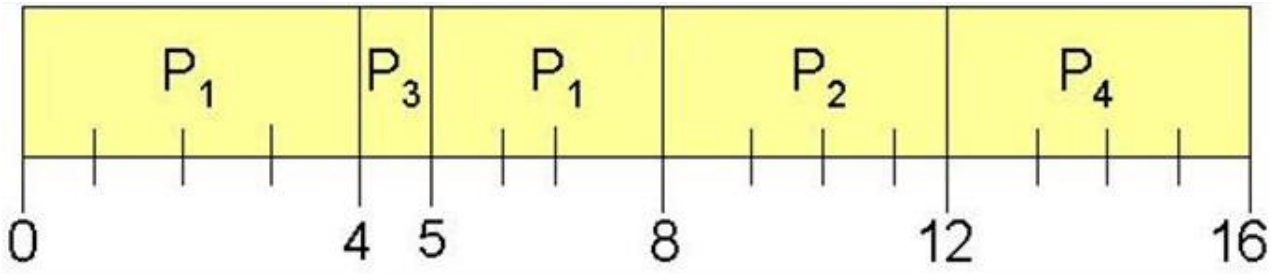
Tenemos 4 procesos que solicitan su entrada de la siguiente manera y tardan y tienen las prioridades siguientes:

Proceso	Tiempo de Llegada	Ciclo de CPU	Prioridad
P_1	0	7	2
P_2	2	4	3
P_3	4	1	1
P_4	5	4	4

2.3.3. Planificación de procesos por el SO Prioridad (apropiativo)

P1 se encuentra en la cola de listos en el momento 0. No hay otros procesos en la cola por lo tanto es despachado al CPU. Al llegar P3 con prioridad mayor, P1 es retirado del CPU con 3 ciclos de CPU faltantes y entra P3 al CPU que se ejecuta hasta su final en el momento 5. En este momento 5 se encuentran P1, P2 y P4 en la cola, P1 tiene mayor prioridad, vuelve al CPU y se ejecuta hasta su final. En el momento 8 se selecciona al proceso P2 con mayor prioridad que P4 y se ejecuta hasta su final momento en el cual se despacha P4 al CPU.

Este sería el diagrama de Gantt:



2.3.3. Planificación de procesos por el SO Prioridad (apropiativo)

- **• Ejemplo de planificación por prioridades (apropiativo):**

Procesos	Tiempo de llegada	Tiempo de ejecución (Servicio)	Prioridad
A	0	5	5
B	1	3	2
C	2	1	1

2.3.3. Planificación de procesos por el SO Prioridad (apropiativo)

- **Ejemplo de planificación por prioridades (apropiativo):**

Procesos/ tiempo	0	1	2	3	4	5	6	7	8	9
A	X					X	X	X	X	
B		X		X	X					
C			X							

Proc/ Tiempo	Llegada	Servicio	Inicio	Fin	Retorno (Fin-Llegada)	Espera (Retorno-Servicio)	Normalizado (Retorno/ Servicio)
A	0	5	0	9	9	4	1,8
B	1	3	1	5	4	1	1,3333
C	2	1	2	3	1	0	1

Nosotros utilizaremos esta representación

2.3.3. Planificación de procesos por el SO Prioridad (no apropiativo)

- **Ejemplo de planificación con prioridad (no apropiativo)**

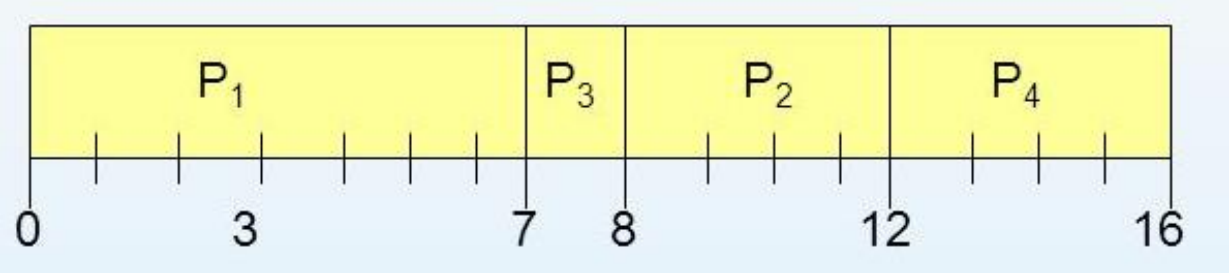
Tenemos 4 procesos que solicitan su entrada de la siguiente manera y tardan y tienen las prioridades siguientes:

Proceso	Tiempo de Llegada	Ciclo de CPU	Prioridad
P_1	0	7	2
P_2	2	4	3
P_3	4	1	1
P_4	5	4	4

2.3.3. Planificación de procesos por el SO Prioridad (no apropiativo)

- P1 se encuentra en la cola de listos en el momento 0. No hay otros procesos en la cola por lo tanto es despachado al CPU. P1 se ejecuta hasta el final. En el momento 7, se encuentran P2, P3 y P4 en la cola. P3 tiene la prioridad más alta por lo tanto es seleccionado y despachado al CPU. Al finalizar P3 en el momento 8, es seleccionado P2 por tener prioridad más alta que P4. Finalmente en el momento 12 es seleccionado el último proceso P4.

Este sería el diagrama de Gantt:



2.3.3. Planificación de procesos por el SO
Prioridad (no apropiativo)

- **Ejemplo de planificación por prioridades (no apropiativo):**

Procesos	Tiempo de llegada	Tiempo de ejecución (Servicio)	Prioridad
A	0	5	5
B	1	3	2
C	2	1	1

2.3.3. Planificación de procesos por el SO
Prioridad (no apropiativo)

- **Ejemplo de planificación por prioridades (no apropiativo):**

Procesos/ tiempo	0	1	2	3	4	5	6	7	8	9
A	X	X	X	X	X					
B							X	X	X	
C						X				

Proc/ Tiempo	Llegada	Servicio	Inicio	Fin	Retorno (Fin-Llegada)	Espera (Retorno-Servicio)	Normalizado (Retorno/ Servicio)
A	0	5	0	5	5	0	1
B	1	3	6	9	8	5	2,6667
C	2	1	5	6	4	3	4

Nosotros utilizaremos esta representación

2.3.4. Planificación de procesos por el SO Colas

- **Múltiples colas.** Es una combinación de los dos anteriores y el implementado en los SSOO actuales. Todos los procesos de una misma prioridad, estarán en la misma cola. Cada cola será gestionada con el algoritmo Round-Robin. Los procesos de colas de inferior prioridad no pueden ejecutarse hasta que no se hayan vaciado las colas de procesos de mayor prioridad.

2.4. Planificación de procesos por el SO Objetivos planificación

En la planificación (scheduling) de procesos se busca conciliar los siguientes objetivos:

- **Equidad.** Todos los procesos deben poder ejecutarse.
- **Eficacia.** Mantener ocupada la CPU un 100 % del tiempo.
- **Tiempo de respuesta.** Minimizar el tiempo de respuesta al usuario.

2.4. Planificación de procesos por el SO Objetivos planificación

- **Tiempo de regreso.** Minimizar el tiempo que deben esperar los usuarios de procesos por lotes para obtener sus resultados (están formados por una serie de tareas de las que el usuario sólo está interesado en el resultado final. Ejemplo: enviar a imprimir documentos).
- **Rendimiento.** Maximizar el número de tareas procesadas por hora.

2. 5. Planificación de procesos por el SO Ejercicios

- **Ejercicio 1: round robin.**

Realiza el diagrama de ejecución de procesos y calcula los tiempos de retorno, espera y normalizado.

Quantum: 2.

Procesos	Llegada	CPU
A	0	8
B	1	5
C	2	4
D	5	4
E	3	7
F	6	4

2. 5. Planificación de procesos por el SO Ejercicios

- **Ejercicio 2: planificación con prioridad (no apropiativo)**

  Realiza el diagrama de ejecución de procesos y calcula los tiempos de retorno, espera y normalizado.

Proceso	Instante de llegada	Tiempo de CPU	Prioridad
A	0	8	5
B	3	4	7
C	6	2	9
D	10	3	8
E	15	6	1
F	24	4	5

2. 5. Planificación de procesos por el SO Ejercicios

- **Ejercicio 3: planificación con prioridad (apropiativo)**

  Realiza el diagrama de ejecución de procesos y calcula los tiempos de retorno, espera y normalizado.

Proceso	Instante de llegada	Tiempo de CPU	Prioridad
A	0	2	4
B	1	4	3
C	3	4	2
D	5	1	1
E	6	2	3