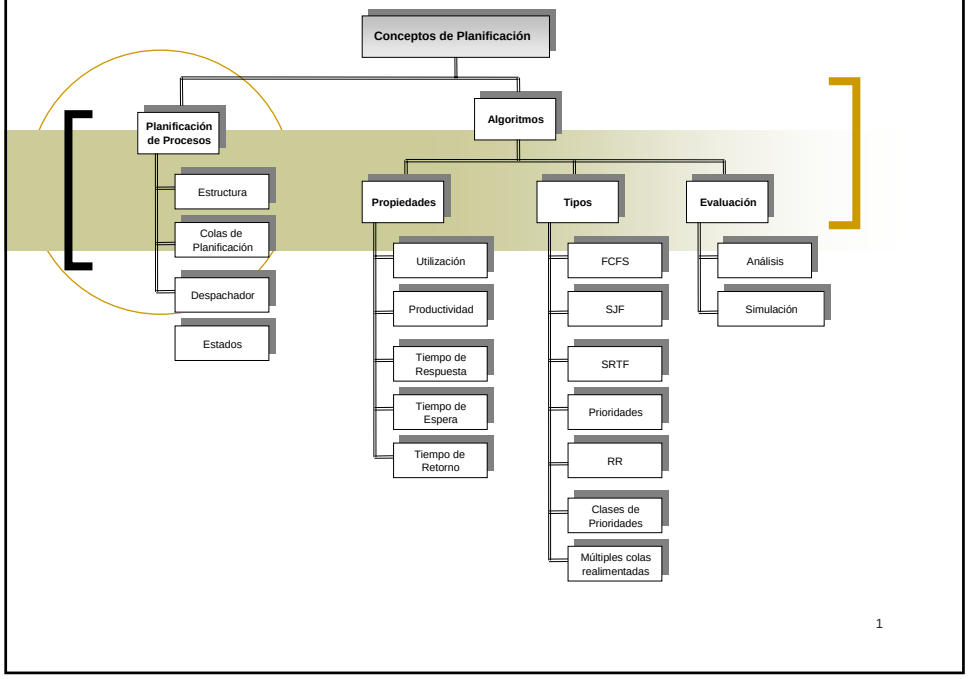


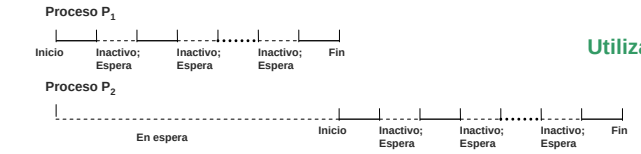
1

Conceptos de Planificación

- El objetivo de la multiprogramación es que en todo momento haya un proceso en ejecución (maximiza la utilización)
- **Planificación**: forma o criterio que se sigue a la hora de decidir que proceso debe entrar en ejecución.
- La tarea de planificación es la más crítica de un SOTR.
- Ventajas de la multiprogramación:
  - Aumento de utilización de CPU (% de actividad de la CPU)
  - Mayor productividad (cantidad de trabajo por u.t.)
- La ejecución de un proceso consiste en una alternancia entre **ráfagas de CPU** y **ráfagas de E/S**

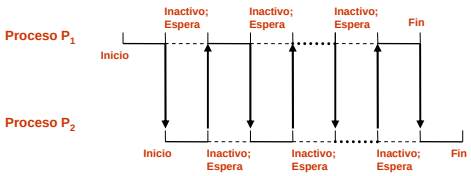
Multiprogramación: Ejemplo

- Un sistema con dos procesos P1 y P2. Cada proceso se ejecuta durante 1 seg. y espera otro seg. Este esquema se repite 60 veces.



Ejecución de procesos sin multiprogramación

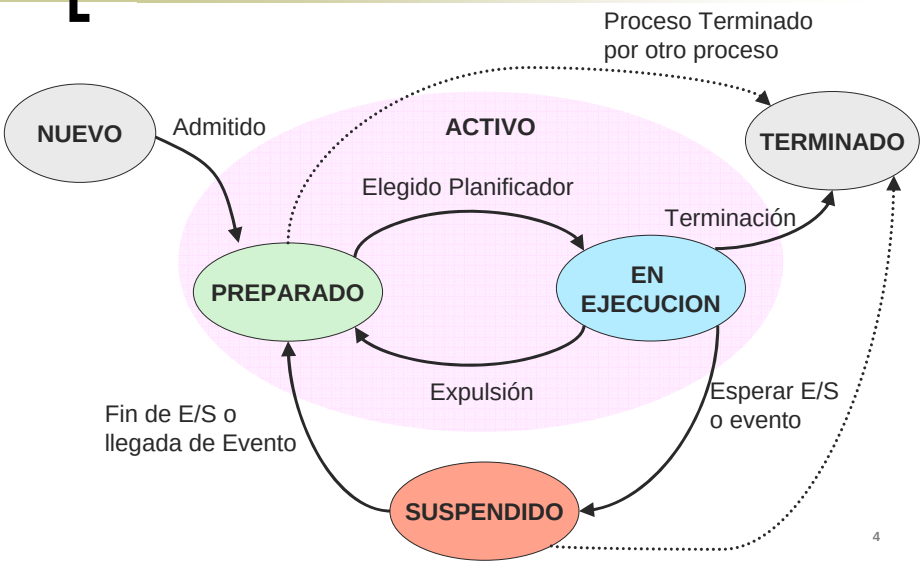
Utilización: 50%



Ejecución de procesos con multiprogramación

Utilización: 100%

Estados de un Proceso

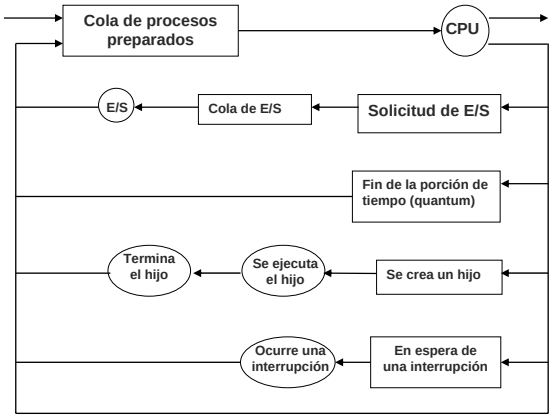


Colas de Planificación (I)

- Un proceso tendrá más o menos posibilidad de entrar en ejecución dependiendo del estado en que se encuentre.
- Es necesario mantener una relación de los procesos que se encuentran en cada estado ⇒ ***Colas de planificación***
- Cuando un programa desea entrar en el sistema se coloca en una *cola de trabajos* a esperar que se le asigne memoria.
- Cuando a un trabajo se le asigna memoria entra en la *cola de procesos* ***preparados***
- Cuando un proceso realiza una operación de E/S pasa a una ***cola de dispositivo*** asociada al dispositivo en el que realiza la operación de E/S

Colas de Planificación (II)

- Una representación común para analizar la planificación de procesos en el *diagrama de colas*



El planificador (I)

- Planificador: parte del SO que se encarga de tomar la decisión de qué proceso entra en ejecución
- Algoritmo de planificación: criterio que utiliza el planificador para designar el proceso que entra en ejecución
- Objetivos de un buen planificador:
  - Equidad
  - Eficiencia (100% utilización)
  - Minimizar el tiempo de espera
  - Aumentar el rendimiento (máximo número de trabajos por u.t.)

El Planificador (II)

- Problemas de un planificador:
  - Alcanzar todos los objetivos provoca contradicciones
  - El comportamiento de los procesos es único e impredecible
- El SO debe evitar que un proceso “monopolice” el uso del procesador
- El SO debe ejecutar cada cierto tiempo el planificador. Si el planificador es capaz de quitar a un proceso el procesador, la planificación denomina ***expulsiva*** *(preemptive)*
- Si cuando un proceso consigue el procesador ya no lo cede hasta que termina, se dice que la planificación es ***no expulsiva***

Estructura de la Planificación

- Las decisiones de la planificación se pueden efectuar en una de las cuatro circunstancias siguientes:
  - **(1).** Un proceso pasa de estado de ejecución a estado suspendido (en espera)
  - **(2).** Un proceso pasa de estado de ejecución a estado preparado (listo)
  - **(3).** Un proceso pasa de estado suspendido (en espera) a estado preparado (listo)
  - **(4).** Cuando termina un proceso
- En el primer y último caso no hay opción en términos de planificación (***no expulsivo***)
- En el resto de casos (**2,3**) es el planificador quien retira el uso de la CPU al proceso mediante una política ***expulsiva***

El despachador (*Dispatcher*)

- El planificador simplemente DECIDE que proceso sale o entra al procesador
- El **despachador** se encarga de entregar o quitar el control de la CPU a un determinado proceso
- Tareas del despachador
  - Cambiar de contexto
  - Cambiar a modo usuario
  - Saltar a la posición adecuada del programa de usuario para reiniciar la ejecución.
- Características del despachador:
  - Ser lo más rápido posible

Algoritmos de Planificación (I)

- Los distintos algoritmos de planificación tienen propiedades diferentes y pueden favorecer o perjudicar a un tipo u otro de procesos.
- Para comparar los algoritmos de planificación se han propuesto varios criterios:
  - ***Utilización de la CPU***: mantener la CPU tan ocupada como sea posible (maximizar)
  - ***Rendimiento (Productividad)***: número de procesos que se completan por unidad de tiempo (maximizar)
  - ***Tiempo de retorno***: tiempo transcurrido desde que se presenta el proceso hasta que se completa (minimizar)
  - ***Tiempo de espera:*** tiempo que un proceso pasa en la cola de procesos listos esperando la CPU (minimizar)
  - ***Tiempo de respuesta:*** tiempo que tarda un proceso desde que se le presenta una solicitud hasta que produce la primera respuesta (minimizar)

Algoritmos de Planificación (II)

- Es deseable maximizar la utilización de CPU y la productividad, y minimizar los tiempos de retorno, de espera y de respuesta
- Puesto que conseguir todo lo anterior es imposible (contradictorio) lo que se desea es llegar a un compromiso entre todos los criterios de forma que se optimice el promedio.
- Algoritmos:
  - Por orden de llegada (**FCFS**) ("First Come First Served")
  - Prioridad al trabajo más breve (**SJF**) ("Shortest Job First")
  - Prioridad al que resta menos tiempo (**SRTF**) ("Shortest Remaining Time First")
  - Planificación por prioridades (estáticas o dinámicas)
  - Planificación circular o "Round Robin" (**RR**)
  - Planificación con clases de prioridades
  - Planificación con múltiples colas realimentadas

Algoritmo FCFS ("First Come First Served")

- La CPU se asigna a todos los procesos en el mismo orden en que lo solicitan
- Propiedades
  - No optimiza: el tiempo de espera, retorno, rendimiento. Muy variables en función del orden de llegada y de la duración de intervalos de CPU
  - Optimiza: utilización
  - Efecto convoy: los trabajos largos retrasan a los cortos
  - No adecuado para sistemas interactivos
  - Muy fácil de implementar (cola FIFO)
  - *No expulsivo*

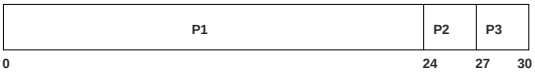
Algoritmo FCFS: Ejemplo

Proceso	Instante de llegada	Tiempo de CPU
P1	0	24
P2	0	3
P3	0	3

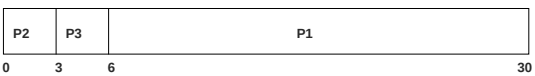
Consideremos los procesos P1, P2 y P3 cuyo comportamiento se muestra en la tabla adjunta

- **Caso 1: orden de llegada P1, P2, P3. Tiempo medio de espera (0 + 24 + 27)/3 = 17**
- **Caso 2: orden de llegada P2, P3, P1. Tiempo medio de espera (6 + 0 + 3)/3 = 3**

Caso 1: Orden de llegada P1, P2, P3



Caso 2: Orden de llegada P2, P3, P1



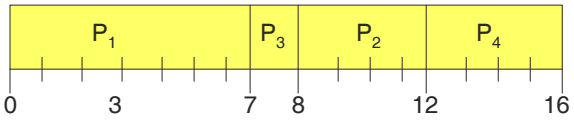
Algoritmo SJF ("Shortest Job First")

- Este algoritmo da prioridad al proceso que va a necesitar menos tiempo de CPU (mejora el tiempo medio de espera)
- Funcionamiento:
  - Asocia a cada proceso un tiempo aproximado de utilización de CPU
  - Asigna la CPU al proceso con menor tiempo asociado
  - Cuando un proceso consigue la CPU la conserva hasta que decide liberarla (**no existe expulsión**)
- Inconvenientes
  - Estimación del tiempo de utilización de CPU por parte de un proceso (a veces se modela con técnicas estadísticas)

Algoritmo SJF ("Shortest Job First") Ejemplo

Procesos	Llegada	Tiempo CPU (ms)
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (*no expulsivo*)



- Tiempo de espera medio = (0 + 6 + 3 + 7)/4 = 4

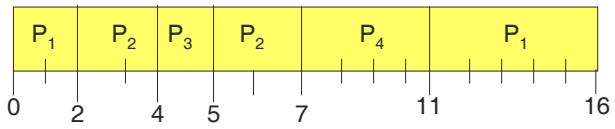
Algoritmo SRTF ("Shortest Remaining Time First")

- Da prioridad al proceso que le resta menos tiempo de CPU para terminar (**variante del SJF con expulsión**)
- Optimiza la media del tiempo de espera y rendimiento
- Funcionamiento:
  - Los procesos llegan a la cola y solicitan un intervalo de CPU
  - Si dicho intervalo es inferior al que le falta al proceso en ejecución para abandonar la CPU, el nuevo proceso pasa a la CPU y el que se ejecutaba a la cola de preparados.
- Inconvenientes:
  - El intervalo de CPU es difícil de predecir
  - Posibilidad de **inanición**: los trabajos largos no se ejecutarán mientras hayan trabajos cortos

Algoritmo SRTF ("Shortest Remaining Time First") Ejemplo

Procesos	Llegada	Tiempo CPU (ms)
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SRTF (*expulsivo*)



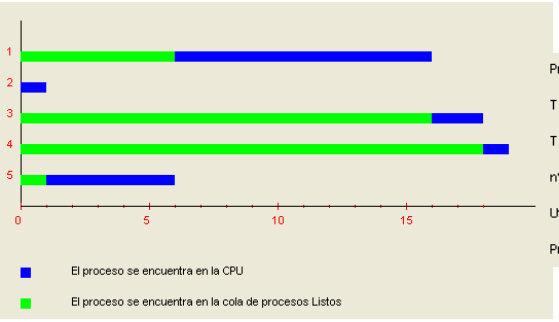
- Tiempo de espera medio = (9 + 1 + 0 +2)/4 = 3

Planificación por prioridades

- Se asocia a cada proceso un número entero llamado ***prioridad*** de acuerdo con algún criterio.
- Se asigna la CPU al proceso con mayor prioridad
- Variantes:
  - Algoritmos con ***expulsión*** o ***sin expulsión***
  - Prioridades estáticas o dinámicas
    - **Estáticas:** se asigna antes de la ejecución y no cambia
    - **Dinámicas:** cambia con el tiempo
- Propiedades:
  - Con prioridades estáticas aparece el problema de ***inanición***: los procesos con baja prioridad no se ejecutan nunca (poco equitativo)
  - El problema anterior se soluciona con la *actualización de prioridades* (dinámicas): la prioridad de un proceso aumenta con el tiempo de espera

Planificación por prioridades Ejemplo

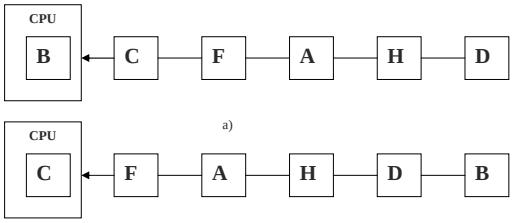
Procesos	Tiempo CPU	Prioridad (Inversa)
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2



Procesos	1	2	3	4	5
T.de Retorno	16	1	18	19	6
T.de Espera	6	0	16	18	1
n°de Ciclos	19				
Utilización	1.0				
Productividad	3				

Turno Rotatorio o Round Robin (RR) (I)

- Es de los más utilizados, sencillos y equitativos.
- A cada proceso se le asigna un intervalo de tiempo llamado cuanto o ***quantum***.*(de 10 a 100ms)*
- Un proceso se ejecuta durante ese **cuanto** de tiempo. Si cuando acaba el cuanto no ha terminado su ejecución, se le expulsa de la CPU dando paso a otro proceso.
- Si un proceso termina antes del cuanto, se planifica un nuevo proceso



a)

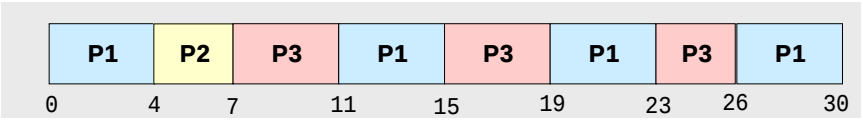
b)

Turno Rotatorio o Round Robin (II)

Quantum q=4

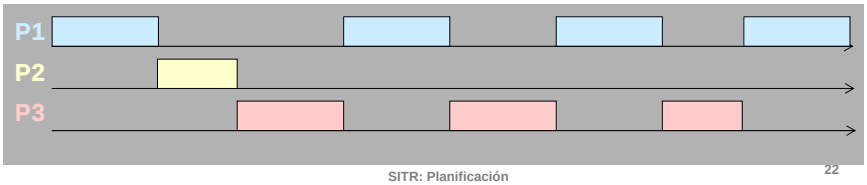
Procesos	T. Llegada	Duración
P1	0	16
P2	0	3
P3	0	11

Diagrama de Gantt



Cronograma por procesos

Tiempo medio Espera =(14+4+15)/3=11

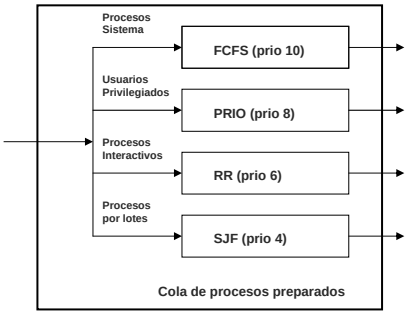


Turno Rotatorio o Round Robin (III)

- Valor del "*quantum*" de tiempo
  - Para **q grandes**: el algoritmo degenera en un algoritmo FCFS.
  - Para **q pequeños**: q ha de ser grande respecto al tiempo necesario para el cambio de contexto, sino la sobrecarga introducida es muy alta.
  - **Regla práctica**: El 80% de los intervalos de CPU han de ser inferiores al "*quantum*" de tiempo.
- Si hay ***n*** procesos en la cola de listos y el quantum es ***q***, cada proceso recibe **1/*n*** del tiempo de CPU. Ningún proceso espera más de **(*n*-1)*q*** unidades de tiempo.
- Propiedades
  - Equitativo
  - Fácil de implementar
  - Normalmente el tiempo de retorno medio es mayor que en **SJF**, pero el tiempo de respuesta es mejor

Planificación con Clases de Prioridades

- Los procesos se clasifican en distintos grupos: sistema, interactivos, tiempo real,...
- La cola de procesos preparados consiste en varias colas donde cada cola tiene su propio algoritmo y además, existe un algoritmo entre colas (p.e. **RR** con ***q*** elevado)

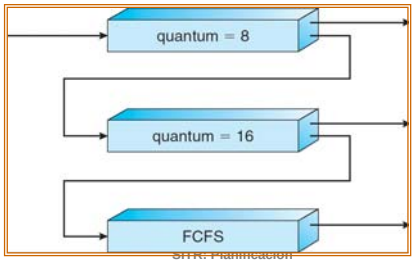


Múltiples Colas Realimentadas

- Existen diferentes colas de procesos preparados.
- Cada cola posee una política de planificación y una prioridad asignada
- Un proceso puede cambiar de cola de acuerdo a un esquema de actualización de prioridades
  - Los procesos con tiempo de espera acumulado elevado pasan a una cola de nivel superior
  - Los procesos con tiempo de utilización de CPU elevados son degradados a una cola inferior

Múltiples Colas Realimentadas Ejemplo

- Tenemos tres colas:
  - Q_0 – RR con quantum 8 ms
  - Q_1 – RR con quantum 16 ms
  - Q_2 – FCFS
- Planificación
  - Un proceso que entra en la cola de procesos listos ingresa en la cola Q_0 . Cuando obtiene la CPU se le asignan 8 ms. Si no termina su ráfaga de CPU en ese tiempo se pasa a Q_1.
  - En Q_1 se asignan 16 ms de CPU al proceso. Si no termina en ese tiempo es expulsado y colocado en la cola Q_2.



Planificador Windows XP

Clases de Prioridad (procesos)

Modificadores (hilos)

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

- El algoritmo es de Colas Multinivel con Realimentación. Cada prioridad tiene asociada una cola con planificación RR.
- Prioridades 0-15 variables, 16-31 fijas (tiempo real).
- A los hilos que agotan su quantum se les reduce la prioridad. Cuando un hilo pasa de suspendido a listo se aumenta su prioridad.

Planificador Linux

- Se usan dos algoritmos: tiempo compartido y tiempo real
- Tiempo compartido
  - Prioridad basada en créditos – el proceso con más créditos es el siguiente en tomar la CPU
  - Los créditos se reducen cuando ocurre una interrupción de reloj
  - Cuando el crédito es 0, se escoge otro proceso
  - Cuando todos los procesos tienen crédito 0 se asigna de nuevo crédito para todos los procesos
    - Basado en factores como prioridad e historia
- Tiempo real
  - Tiempo real blando
    - Cumple el estándar Posix.1b – dos clases
    - FCFS y RR
    - El proceso de mayor prioridad siempre se ejecuta primero

Evaluación de Algoritmos

- Para seleccionar un algoritmo
  - Seleccionar un criterio (utilización, t. de respuesta, t. espera, t. retorno, ...)
  - Estudiar la adaptación del algoritmo a esos criterios.
- Métodos de evaluación de algoritmos
  - Evaluación analítica
    - Mediante modelo determinista
    - Mediante modelos de colas
  - Simulación