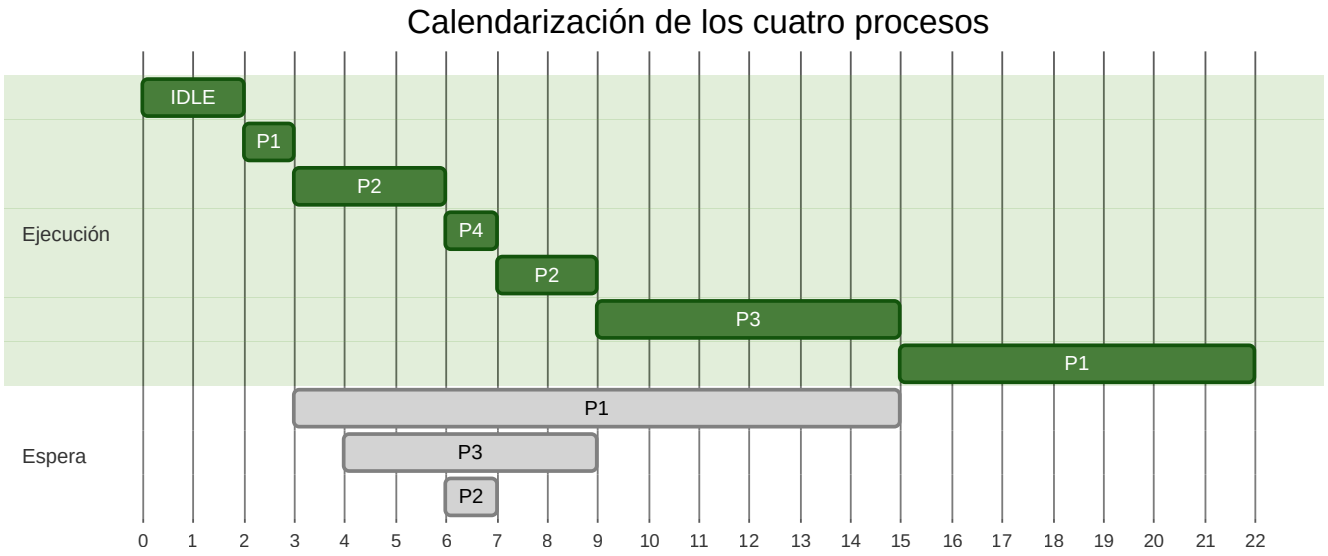
Sistemas Operativos

Tarea 2

Alumno: **Edgar Quiroz**

1. ¿Qué sucede si el quantum tiene una duración a la latencia del despachador?
   Los procesos que realizan un cambio de contexto para otro proceso agotaría su tiempo de procesador.
   Y para todos los procesos, como la latencia es lo más breve posible, su tiempo de procesador sería muy poco.
   Así que en general, la mayoría de los proceso no podría hacer muchas operaciones y el desempeño general del sistema
   se alentaría considerablemente.
2. Dada la siguiente tabla y suponiendo que los procesos son calendarizados con *SJB*, ¿en qué instante se va a empezar
   a ejecutar el proceso P_3?

Proceso	**Llegada**	**Burst**
P_1	2	8
P_2	3	5
P_3	4	6
P_4	6	1

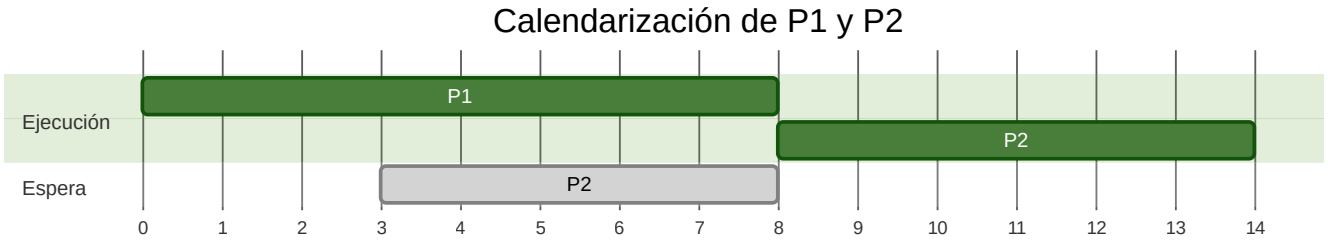


Por lo que P_3 se empieza a ejecutar en el instante 9.

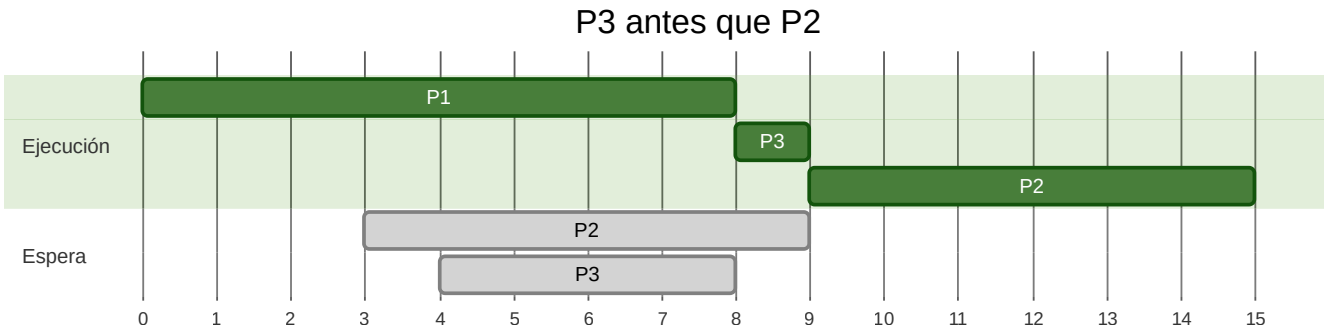
3. Considera la siguiente tabla

Proceso	**Llegada**	**Burst**
P_1	0	8
P_2	3	6
P_3		

Suponiendo que el segundo proceso en llegar es P_2 y se usa la calendarizacioń *SJB*, ¿es posible que P_3 sea el
segundo proceso en ejecutarse?
Explique y llene lo parámetros de ser posible.
El diagrama de Gantt para esos dos procesos sería



En el instante 8, P_2 se empieza a ejecutar porque tiene el *burst* mínimo, con 6.
Si en ese momento hubiera algún otro proceso con un *burst* más pequeño, sería ejecutado antes de P_2,
independientemente de que P_2 lleve más tiempo en espera.
Por ejemplo,



Que corresponde a la tabla de procesos

Proceso	**Llegada**	**Burst**
P_1	0	8
P_2	3	6
P_3	4	1

Se cumple que P_3 llega después que P_2 y que se ejecute antes.

4. Considera la siguiente tabla

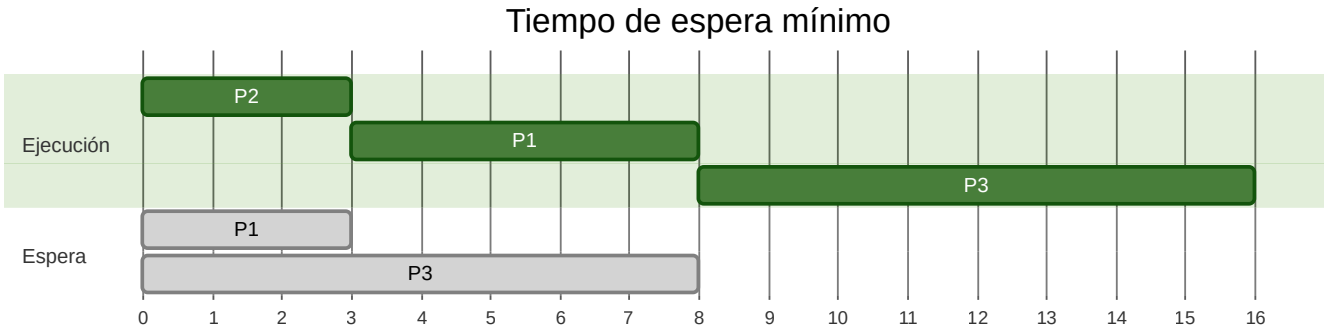
Proceso	**Burst**
P_1	5
P_2	3
P_3	7

Ordena los procesos para minimizar el tiempo de espera suponiendo que todos los procesos llegan al tiempo 0.
Supongamos que la ejeucición de procesos es preemtive. Esto es que toda la ejecución de un proceso se debe realizar
en un sólo bloque.
Como hay tres procesos, un proceso no esperará, otro proceso esperará al anterior, y el último esperará a los dos
anteriores.
Por lo que el tiempo de espera promedio en general sería

$$3\overline{T} = 2b_1 + b_2$$

Donde b_i es el burst del i-ésimo proceso en ejecutarse.

Como se quiere minimizar esta expresión, hay que elegir el valor de b_i más pequeño para b_1 y el segundo más
pequeño para b_2, que corresponde a *SJB*.
Por lo que el orden quedaría como



Con $3\overline{T} = 11$.

5. ¿Qué tipo de calendarización da preferencia a los procesos *IO-Bound*?
   Usando una cola multinivel con feedback, se tiene que los procesos con *burst* largos serán penalizados siendo
   transferidos a colas de menor prioridad.
   Por lo que en general un proceso *IO-Bound* tendría mejor posición en las colas que un proceso con uso intenso de
   CPU.
   Así que se podría considerar que este método da preferencia a los procesos *IO-Bound*.
6. Considera la siguiente tabla

Proceso	**Burst**
P_1	10
P_2	3
P_3	2
P_4	1

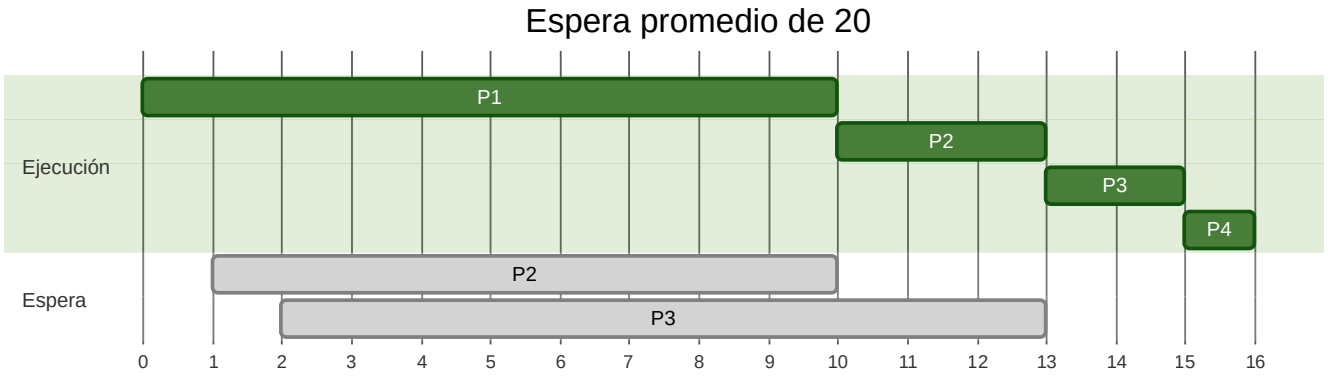
   Suponiendo que se utiliza la calendarización *FCFS*, ¿es posible que el tiempo de espera promedio sea 5 si P_1
   llega primero?
   Suponiendo que todos los procesos llegan en el tiempo 0, los procesos P_2, P_2, P_4 tendría que esperar 10
   unidades para que termine P_1.
   Entonces, para cualquier orden

$$4\overline{T} \geq 0 + 10 + 10 + 10 = 30 > 20$$

Por lo que si todos los procesos llegan al tiempo 0, no sería posible tenen un tiempo de espera promedio de 5.
Por otro lado, si los procesos no tienen que llegar forzosamente al tiempo 0, con el siguiente orden

Proceso	**Burst**	**Llegada**
P_1	10	0
P_2	3	1
P_3	2	2
P_4	1	15

Que corresponde a la siguiente ejecución



Se tiene que sólo P_2 y P_3 esperan, con 9 y 11 respetivamente.
Esto da un tiempo de espera promedio de $4\overline{T} = 0 + 9 + 11 + 0 = 20$, que es un tiempo promedio de 5.

7. ¿Qué proceso modifica se encarga del envejecimiento durante la hambruna?
   El envejecimiento se realiza cada cierta cantidad fija de tiempo. El proceso que esté ejecutándose cuando pase el
   tiempo necesario se encargará de llevar acabo el envejecimiento y de cambiar de proceso de ser necesario.
8. ¿Es conveniente que algún proceso tenga el parámetro *nice* al máximo?
   Sí, pues es necesario tener siempre un proceso que se encargue de todas las necesidades periódicas del sistema.
   Aún si todos los procesos duermen, debería haber un proceso encargado. Para que sólo esté activo en ese caso, debería tener su parámetro *nice* al máximo.
9. Explica que es la inversión de prioridades y da un ejemplo.

Sean los procesos P_1, P_2 con prioridades $P_1^p < P_2^p$.
Dígamos que P_2 requiere de un recurso R que P_1 está usando.
Como P_2 no puede acceder al recurso, se bloquea, y deja a P_1 en ejecución.
Entonces, aunque P_2 tenga mayor prioridad, su ejecución es interrumpida por P_1. Es como si las prioridades
estuvieran invertidas.

10. ¿Hay alguna alternativa para que en una cola multinivel sin feedback se pueda ejecutar un proceso de una cola de
prioridad menor antes que uno de una cola prioridad mayor?
Un mecanismo para esto es *time_slicing*. Esto consiste en asignar cierto
porcentaje fijo del procesador a cada cola.
Entonces, el primer elemento de una colo de prioridad inferior podría
ejecutarse antes que el último proceso de la cola de mayor prioridad.