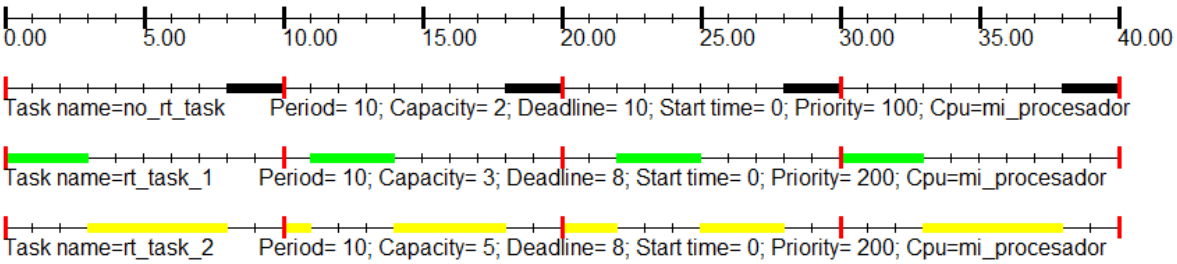
Core

Se le define el tipo de scheduler (planificador) y un quantum. El planificador en este ejercicio es POSIX 1003.1b/Highest Priority First, que básicamente hace que las tareas de prioridad más alta se ejecuten primero (para desempates, ver pág. 2).

Tarea

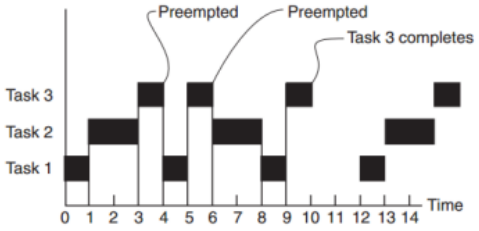
Para crearlas debe existir un core, un procesador (que utilice dicho core) y un espacio de direcciones o memoria. En una tarea se definen los siguientes parámetros[1]:

- Período: plazo de ejecución de la tarea (cada cuanto se vuelve a activar).
- Prioridad: valor que lee POSIX para el scheduling, entre 1 y 255.
- Capacidad: es la duración de la tarea, para cada período de la misma.
- Deadline: tiempo máximo para que finalice la tarea (no debe superarlo).
- Start time: delay inicial para cada activación de la tarea.
- Planificador: para tareas que tienen una misma prioridad (Round Robin o FIFO).



[1] Más detalles: https://www.eecs.umich.edu/courses/eecs498-brehob/Labs/Lab5.pdf

Task	Execution Time	Period	Priority
T1	1	4	High
T2	2	6	Medium
T3	3	12	Low



Round Robin

En POSIX, cuando hay dos tareas de misma prioridad para ejecutar, Round Robin asigna un quantum (tiempo de uso de la CPU) a cada una. Vemos el caso con Quantum 3. En el primer período, la tarea *rt_task_1* se ejecuta 3 segundos y luego le pasa el turno a *rt_task_2*. Al llegar a los 6 segundos, le tocaría a la 1, pero ya finalizó (capacity 3), entonces sigue ejecutándose *rt_task_2* hasta terminar en el segundo 8.

Cabe aclarar que el Quantum se reinicia a 0 cada vez que supera el máximo.

Tarea	Inicio	Fin	Quantum
RT_TASK_1	t = 0	t = 3	-
RT_TASK_2	t = 3	t = 8	5 % 3 = 2

En el segundo período (t = 10), empieza a ejecutarse *rt_task_2* porque aún le restaba un Quantum para llegar a 3. Entonces, se ejecuta por 1 segundo, y le pasa el turno a *rt_task_1*, que se ejecuta por 3 segundos, momento en el que finaliza y le pasa el turno nuevamente a *rt_task_2*, que se ejecuta durante 4 segundos hasta terminar.

Tarea	Inicio	Fin	Quantum
RT_TASK_2	t = 10	t = 11	-
RT_TASK_1	t = 11	t = 14	-
RT_TASK_2	t = 14	t = 18	4 % 3 = 1

Nuevamente, en el tercer período (t = 20), se ejecuta primero *rt_task_2* durante 2 segundos hasta completar el Quantum. Luego le pasa el turno a *rt_task_1*, que se ejecuta 3 segundos y finaliza, y continúa *rt_task_2* por 3 segundos y termina.

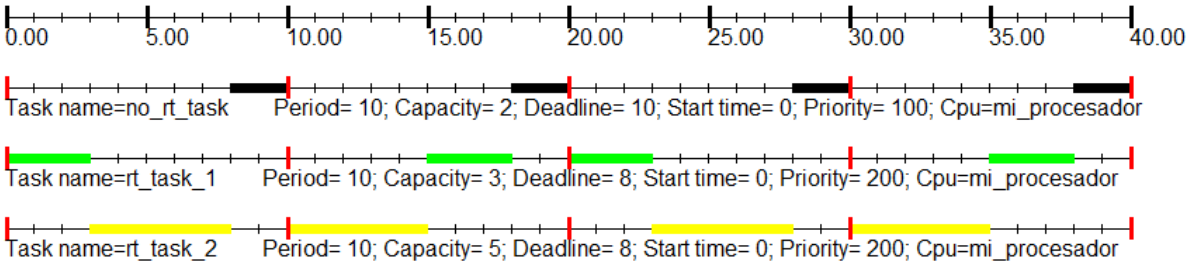
Tarea	Inicio	Fin	Quantum
RT_TASK_2	t = 20	t = 22	-
RT_TASK_1	t = 22	t = 25	-
RT_TASK_2	t = 25	t = 28	-

En el cuarto período (t = 30), ahora se ejecuta primero *rt_task_1*, ya que la otra completó su Quantum, repitiéndose la situación del primer período (t = 0).

FIFO

First In First Out (FIFO) is, in some ways, the simplest scheduling algorithm. As the name suggests, this policy is based on a queue where the first process that enters the queue is the first process that is executed. Each process runs to completion (and no process can be preempted) before the scheduler executes the next process in the queue. And because there is no prioritization in FIFO, systems using this policy often have trouble meeting deadlines, even in systems with low total CPU utilization.

A common variation of FIFO is a priority-based version of FIFO. This is generally implemented with preemption. The idea is that higher priority tasks are always run first but when two tasks both have the same priority, the one that arrived first is run.

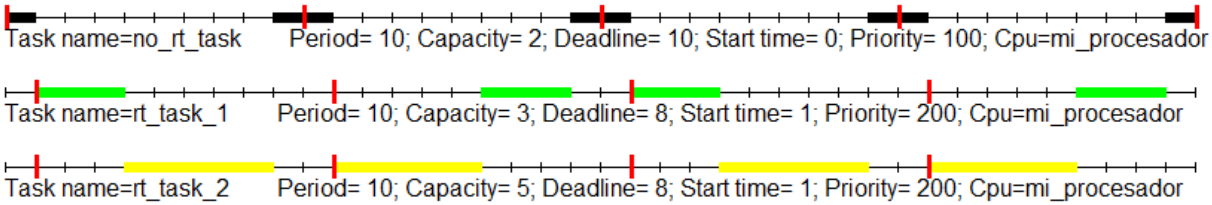


En el caso del POSIX, aplica el segundo párrafo. Sin embargo, este ejercicio tiene la particularidad de que las tareas siempre coinciden en instante de activación, ya que el período es el mismo en todos los casos. Por intuición, Cheddar asume que en

un primer caso la tarea *rt_task_1* llegó un ínfimo instante antes que *rt_task_2* en t = 0, y luego lo contrario en t = 10, y así indefinidamente (la última en ejecutarse arriba antes).

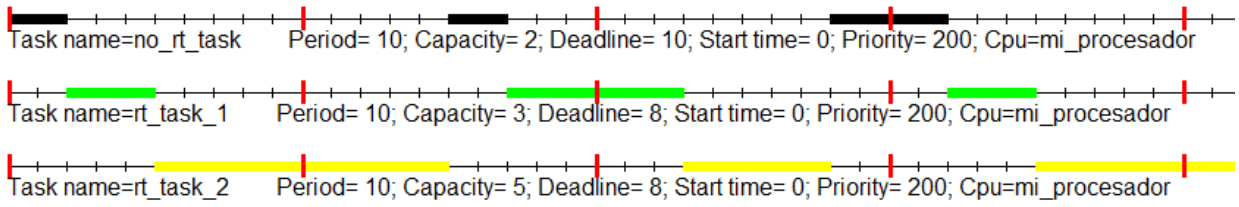
Apropiación en FIFO

Si cambiamos el Start Time de las tareas de tiempo real a 1, vemos que existe apropiación por parte de éstas cuando *no_rt_task* se encuentra en ejecución:



FIFO para activaciones simultáneas

Cambiando la prioridad de *no_rt_task* para evaluar qué sucede cuando hay 3 tareas que arriban a la vez, podemos observar que FIFO permite que la última tarea que se encontraba ejecutándose siga teniendo la "adquisición" de la CPU (t = 10).



Consideraciones generales

We want to know if these tasks are schedulable. First, if the total CPU utilization were greater than 1, we'd know the tasks were not schedulable. Next, we examine the critical instant— the time when all tasks try to start at the same time. In a static scheduling algorithm, this is the worst-case situation.

Como dice el párrafo anterior, lo primero que hay que observar para saber si las tareas se pueden planificar o no es calcular el factor de utilización de CPU (entre 0 y 1).

$$Uso\ CPU = \sum_{i=1}^{n} \frac{Tiempo\ de\ ejecución\ de\ la\ tarea\ i}{Plazo\ o\ período\ de\ la\ tarea\ i}$$

Si pasa esta condición, entonces se analiza el momento cuando todas las tareas deben comenzar, es decir, la activación de estas coincide en un mismo tiempo t.

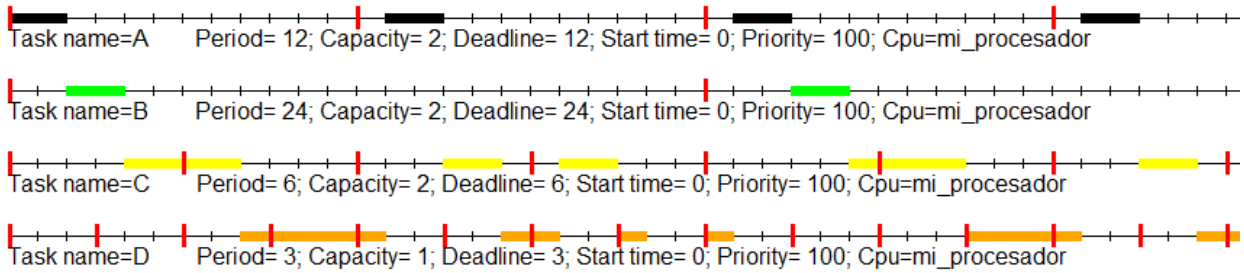
En el ejercicio 3, el uso de CPU es 100%, y vimos que existe planificación.

Ejercicio 1

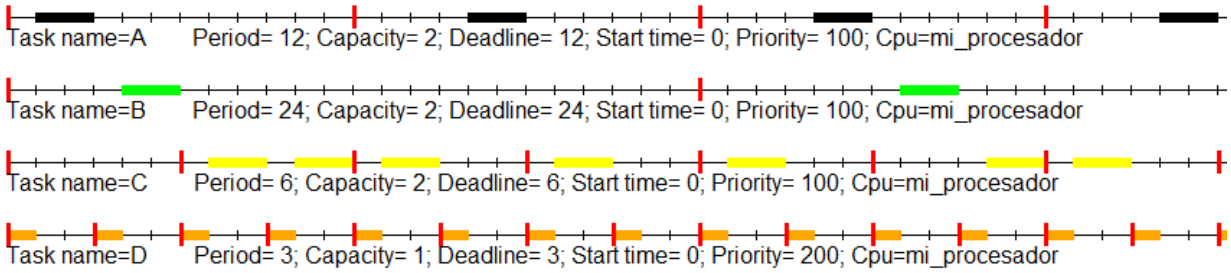
En este caso, el uso de CPU es 2/12 + 1/12 + 4/12 + 4/12 = 11/12 = 91.7%, entonces puede existir alguna planificación viable por el momento.

Tarea	Tiempo de ejecución	Período/Plazo
A	2	12
B	2	24
C	2	6
D	1	3

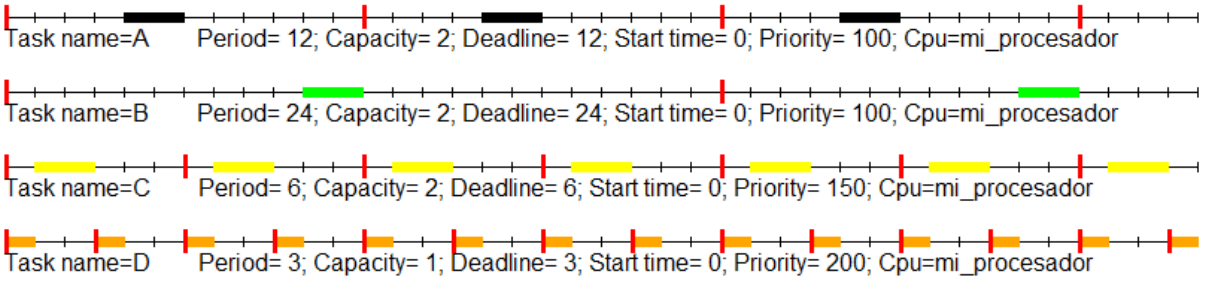
Intento 1: misma prioridad a todos, FIFO. **Problema: venció el plazo de D.**



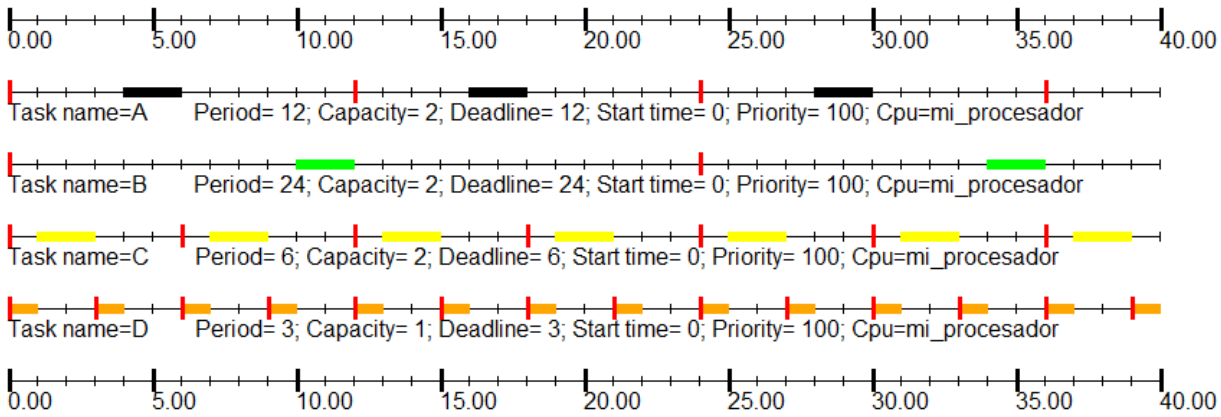
Intento 2: prioridad para D, FIFO. **Problema: venció el plazo de C.**



Intento 3: Prioridad(D) > Prioridad(C) > Prioridad(A) = Prioridad(B), FIFO. **Es viable.**



Una solución más elegante es usar un core con política de planificación Earlist Deadline First (EDF), de modo que siempre ejecuta primero las tareas de menor plazo.



Resultó entonces un período principal de 24, y un secundario de 3.

Ejercicio 2

En este caso, el uso de CPU es 2/20 + 5/20 + 7/20 + 4/20 = 18/20 = 90%, entonces puede existir alguna planificación viable por el momento.

Tarea	Tiempo de ejecución	Período/Plazo
A	1	10
B	3	12
C	7	20
D	1	5

Siguiendo la misma conclusión que para el ejercicio 1, encontramos la siguiente planificación viable, con período principal 60 y secundario 5. **Nota: hay apropiación.**