# Planificación

Un sistema utiliza el algoritmo feedback con la siguiente configuración:

* Hay 3 colas de planificación de distinta prioridad
* Todos los procesos ingresan a una cola de RR Q = 2 (máxima prioridad)
* A los que no les alcanza ese Q son desalojados por interrupción de timer a otra cola menos prioritaria con Q = 3
* A los que tampoco les alcanza ese nuevo Q son desalojados a la cola menos prioritaria que es de tipo FIFO
* Las colas de RR sólo son desalojadas respecto al Q
* La cola de FIFO puede ser desalojada si llega un proceso a una cola más prioritaria (colocándose al final de la cola FIFO)
* Luego de volver a ready por fin de IO todos los KLTs vuelven a la cola más prioritaria (RR Q = 2)

|  |  | **T Arribo** | **CPU** | **IO** | **CPU** | **IO** | **CPU** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| KA | UA1 | 0 | 1 | 3 | 2 | 2 | 1 |
| UA2 | 1 | 3 | 1 | 1 |  |  |
| UA3 | 2 | 3 | 1 | 1 |  |  |
| KB | | 1 | 3 | 2 | 1 | 3 | 1 |
| KC | UC1 | 4 | 1 | 2 | 2 | 2 | 1 |
| UC2 | 5 | 3 |  |  |  |  |

A su vez KA utiliza una biblioteca de hilos con jacketing y algoritmo FIFO. Por su parte, KC utiliza una biblioteca de ULTs sin jacketing y algoritmo SJF (igualmente los ULTs realizan sus IOs a través de las funciones de la biblioteca)

1. Realice el diagrama Gantt con la traza anterior.
2. Responda:
   1. Seleccione un instante en que los 3 hilos hayan estado en distintas colas (en caso de no encontrarlo que sean en 2 colas distintas). Explique cómo cada uno llegó a dicha cola.
   2. ¿Qué tan provechoso fue para KA haber utilizado una biblioteca con jacketing con este algoritmo de planificación?

# 

# Sincronización

A continuación se muestra una simplificación de la vida normal de los alumnos de un curso de sistemas operativos, de su docente y sus ayudantes de TP.

alumno\_conectado = 0

clase\_terminada = 0

ayudantes\_libres = 4

pregunta = {0, 0, 0, 0}

respuesta = {0, 0, 0 ,0}

mutex\_getId = 1

| Docente curso (1 instancia) | Alumno (63 instancias) | Ayudante TP (4 instancias) |
| --- | --- | --- |
| while(true) {  wait(alumno\_conectado, 63)  darClase();  signal(clase\_terminada, 63)  } | while(true) {  signal(alumno\_conectado)  cursar();  wait(clase\_terminada)  wait(ayudantes\_libres)  id\_ayudante = getIdAyudante()  hacerPreguntas(id\_ayudante);  signal(pregunta[id\_ayudante])  wait(respuesta[id ayudante])  codearTP(); //se bloquea hasta su próxima clase de SO  } | while(true) {  wait(pregunta[getId()])  responder();  signal(respuesta[getId()])  signal(ayudantes\_libres)  } |

Al comienzo del día los alumnos se conectan a cursar la materia, la cual no comienza hasta que todos los alumnos estén conectados. Luego de terminada la clase buscan algún ayudante libre (que no esté dándole soporte a nadie) para que los ayude con el TP. Para que sea una sesión productiva el alumno primero le hace sus preguntas y luego el ayudante le responde. Una vez que sus dudas le fueron respondidas por su ayudante, se pone a codear el TP hasta el próximo sábado.  
Aclaraciones:  
 - La función getIdAyudante() del proceso Alumno devuelve un número del 0 al 3 correspondiente a un ayudante libre y falla si no hay ayudantes libres. La misma se considera atómica.  
- El proceso Ayudante TP cuenta con una función getId() que devuelve un número del 0 al 3 correspondiente a sí mismo y la misma se considera atómica.

Sincronice los procesos solamente utilizando semáforos para que se pueda tener un buen cuatrimestre sin que se produzca deadlock ni starvation.

# Resoluciones

[Resolución Ej plani](https://docs.google.com/spreadsheets/d/1qcIPRQuBtblLsyU7xxccGKN5_tIyStYJInVx7oKUBlU/edit#gid=0)

| Docente curso (1 instancia) | Alumno (63 instancias) | Ayudante TP (4 instancias) |
| --- | --- | --- |
| while(true) {  wait(alumnosConectados, 63)  darClase();  signal(claseTerminada, 63)  } | while(true) {  signal(alumnosConectados)  cursar();  wait(claseTerminada)  wait(ayudantesLibres)  id\_ayudante = getIdAyudante()  hacerPreguntas(id\_ayudante);  **signal(consulta[id\_ayudante])**  **wait(respondido[id\_ayudante])**  signal(ayudantesLibres)(una opción)  codearTP(); //se bloquea hasta su próxima clase de SO    } | while(true) {  **wait(consulta[getId()])**  responder();  **signal(respondido[getId()])**  signal(ayudantesLibres) (otra opción)  } |

**Inicializaciones:**

alumnosConectados = 0  
claseTerminada = 0  
ayudantesLibres = 4  
consulta = {0, 0, 0, 0}

respondido = {0, 0, 0, 0}