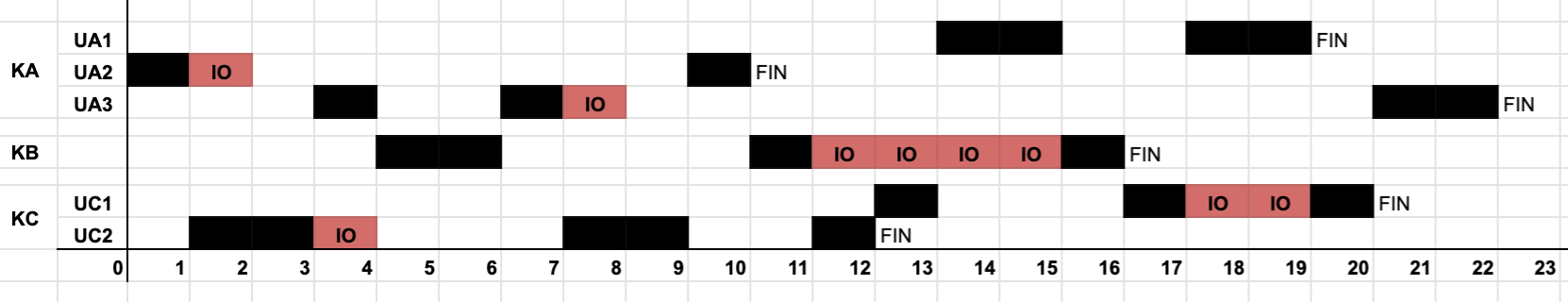
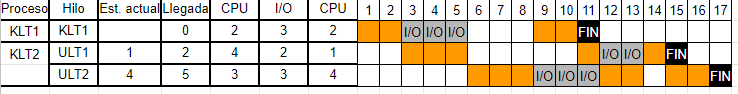
# Planificación

Sabiendo que los tiempos de arribo de los KLTs son : KA = 0, KC = 1 , KB = 3. Indique:

1. ¿Qué algoritmo podría estar usando el SO? (\*)
2. ¿Qué algoritmo podría estar usando la biblioteca de KA? (\*)
3. ¿Qué algoritmo podría estar usando la biblioteca de KC? (\*)
4. (Sin volver a resolver el ejercicio) Indique qué cambios hubiesen ocurrido si las bibliotecas hubiesen usado jacketing.
5. ¿En caso de utilizar otro algoritmo el SO, se podría haber terminado la ejecución total antes?

(\*) Justifique mostrando al menos 2 instantes

Un estudiante realizó el siguiente Gantt. Se presupone que el sistema operativo, para planificar, utiliza RR con Q=3 para los KLTs, y para ULTs utiliza SJF con desalojo.  


Responder:  
a) ¿Se puede determinar algún error en el Gantt? ¿A partir de qué instante? ¿Por qué razón podría haber sido?  
b) ¿A qué valor debería tender el coeficiente de estimación α para que el ULT1 desaloje al ULT2 en el instante 13? Realice las cuentas y justifique. (ESTn+1 = Rn\*α + ESTn(1-α))  
c) ¿Cambiaría el Gantt en algún instante si decimos que no hay jacketing entre los ULTs?

## 

# Sincronización

Como tienen mucho tiempo libre en su casa, dos hermanos, Fred y George, decidieron organizar su armario que cuenta con muchísima ropa desparramada.  
Fred se encarga de sacar de a una prenda del armario, probársela y si le queda bien, seleccionarla para volver a guardarla en un cajón. Si no, la separa para donar. Mientras, George va a ir tomando de a una la ropa que haya seleccionado su hermano, y en base al tipo de prenda la colocará en un cajón diferente.  
Existen 3 tipos de prendas, el armario cuenta con 3 cajones con una capacidad de 15 prendas cada uno.   
Su hermano menor, Ron, para molestarlos, aprovecha la ausencia de Fred cada vez que separa una prenda para donar, y saca otra previamente guardada en algún cajón y la vuelve a poner en el armario, por lo que Fred y George tendrán que realizar estas actividades infinitamente.   
La función tipoDePrenda() devuelve un número del 0 al 2 en base al tipo de prenda de la que se trate, y la función random() devuelve un número del 0 al 2 de manera aleatoria.   
Sincronizar utilizando únicamente semáforos para que la ropa de los hermanos esté ordenada en los cajones  
  
Variable compartida: cajones[3]

| Proceso Fred(Seleccionador) | Proceso George(Llenador) | Proceso Ron(Desordenador) |
| --- | --- | --- |
| while(true) {  agarrarPrenda()  if(quedaBien()) {  seleccionar()  } else {  separar()  }  } | while(true) {  id\_prenda = tipoPrenda()    ponerEnCajon(cajones[id\_prenda])  } | while(true) {  id\_prenda = random()  sacarPrendaDelCajonYDevolverAlArmario(cajones[id\_prenda])  } |

## 

# Resoluciones

a) A partir de los algoritmos brindados, se puede determinar un error a partir del instante 5 del Gantt. Es un error ya que debería de seguir la ejecución del ULT1 porque su estimación anterior es menor a la estimación anterior del ULT2, sin embargo, el ULT2 desaloja al ULT1. El error puede haber surgido por alguna confusión con las estimaciones.  
  
b) Si tenemos en cuenta que la estimación anterior es 1 para el ULT1 y 4 para el ULT2, y que la ráfaga anterior del ULT1 fue mayor a la ráfaga anterior del ULT2, lo que buscamos es darle una mayor importancia a la estimación anterior, por lo tanto buscamos que el estimador tienda a 0:  
  
Rafaga ULT1 en el instante 14 = 4\*0 + 1\*(1-0) = 1  
Rafaga ULT2 en el instante 14 = 3\*0 + 4\*(1-0) = 4 - 2 ejecutadas = 2  
Con estas rafagas obtenidas, el ULT1 desaloja al ULT2.  
  
c) A partir del instante 10 el Gantt cambiaría, ya que el ULT2 está bloqueado porque se está realizando una entrada/salida, y como no habría jacketing, el ULT1 no podría ejecutar.

sem\_espacioEnCajon = {15,15,15}

sem\_hayPrendaEnPila = 0

sem\_desordenar = 0

sem\_hayPrendaEnCajon = {0,0,0}

mutex\_cajon = {1,1,1}

| Proceso Fred(Seleccionador) | Proceso George(Llenador) | Proceso Ron(Desordenador) |
| --- | --- | --- |
| while(true) {  agarrarPrenda()  if(quedaBien()) {  seleccionar()  signal(sem\_hayPrendaEnPila)  } else {  separar()  signal(sem\_desordenar)  }  } | while(true) {  wait(sem\_hayPrendaEnPila)  id\_prenda = tipoPrenda()    wait(sem\_espacioEnCajon[id\_prenda])  wait(mutex\_cajon[id\_prenda])  ponerEnCajon(cajones[id\_prenda])  signal(mutex\_cajon[id\_prenda])  signal(sem\_hayPrendaEnCajon[id\_prenda])  } | while(true) {  wait(sem\_desordenar)  id\_prenda = random()  wait(sem\_hayPrendaEnCajon[id\_prenda])  wait(mutex\_cajon[id\_prenda])  sacarPrendaDelCajonYDevolverAlArmario(cajones[id\_prenda])  signal(mutex\_cajon[id\_prenda])  signal(sem\_espacioEnCajon[id\_prenda])  } |