75.08 Sistemas Operativos
Primer Parcial 4 de Octubre de 2022
Nombre y Apellido: [redacted]
Padron: [redacted]

1. Kernel y Procesos (10 ptos.)

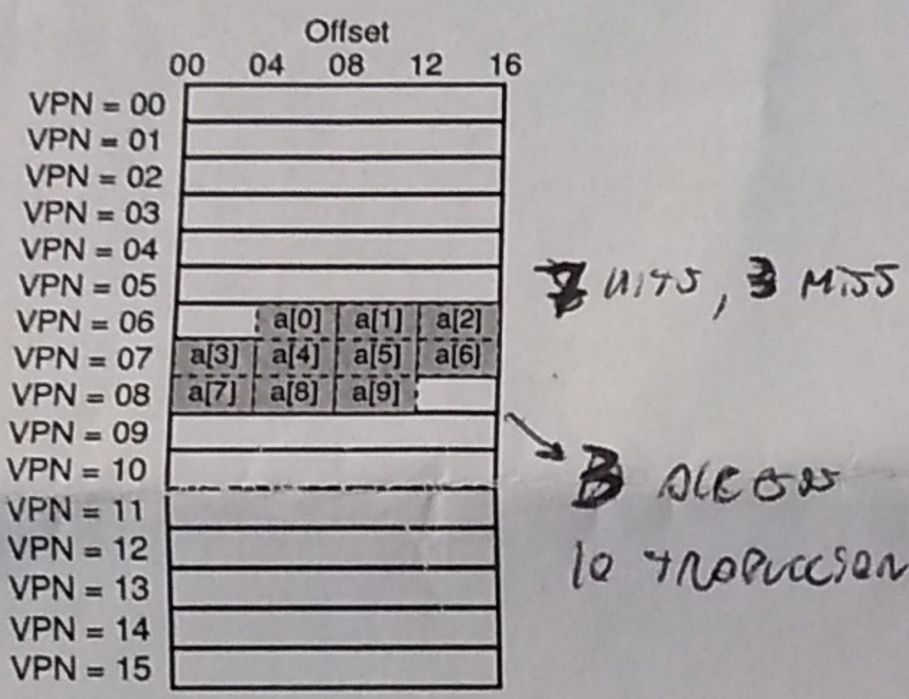
   a. Describa que es un proceso: qué abstrae, cómo lo hace, cuál es su estructura. Además explique el mecanismo por el cual el proceso cree tener la memoria completa de la máquina cuando en realidad solo tiene lo necesario para su funcionamiento.

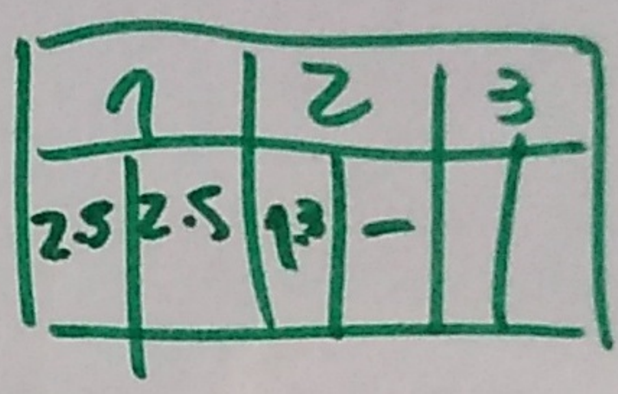
   b. Cuál / cuáles mecanismos utiliza el kernel para garantizar el aislamiento entre procesos. Estos mecanismos están relacionados con el hardware, porque deben existir y donde se ve su funcionamiento.

INS

2. Memoria (10 ptos.)

   a. Dado el siguiente esquema explique cómo se realizan las traducciones recorriendo el arreglo en un modelo de memoria virtual con tlb y paginación de dos niveles. En el mismo esquema decir cuantos miss, hit , accesos a memoria y traducciones hay.





Responda:

- [ ] En las traducciones hay 3 hits y 7 miss en la TLB.
- [ ] Hay 10 accesos a memoria.
- [x] En las traducciones hay 7 hits y 3 miss en la TLB.
- [ ] Hay 3 traducciones completas de VA a PA.
- [x] Hay 3 accesos a memoria en total.
- [x] Hay 10 traducciones completas de VA a PA.

B. Suponga que virtual address con las siguientes caracteristicas:

4 bit para el segment number
12 bits para el page number
16 bits para el offset

Segment table	Page Table A	Page Table B
0 Page B	0 CAFE	0 F000
1 Page A	1 DEAD	1 D8BF
X invalid	2 BEEF	2 3333
	3 BA11	x INVALID

Traducir las siguientes direcciones virtuales a físicas: 00000000, 20022002, 10022002, 00015555 .

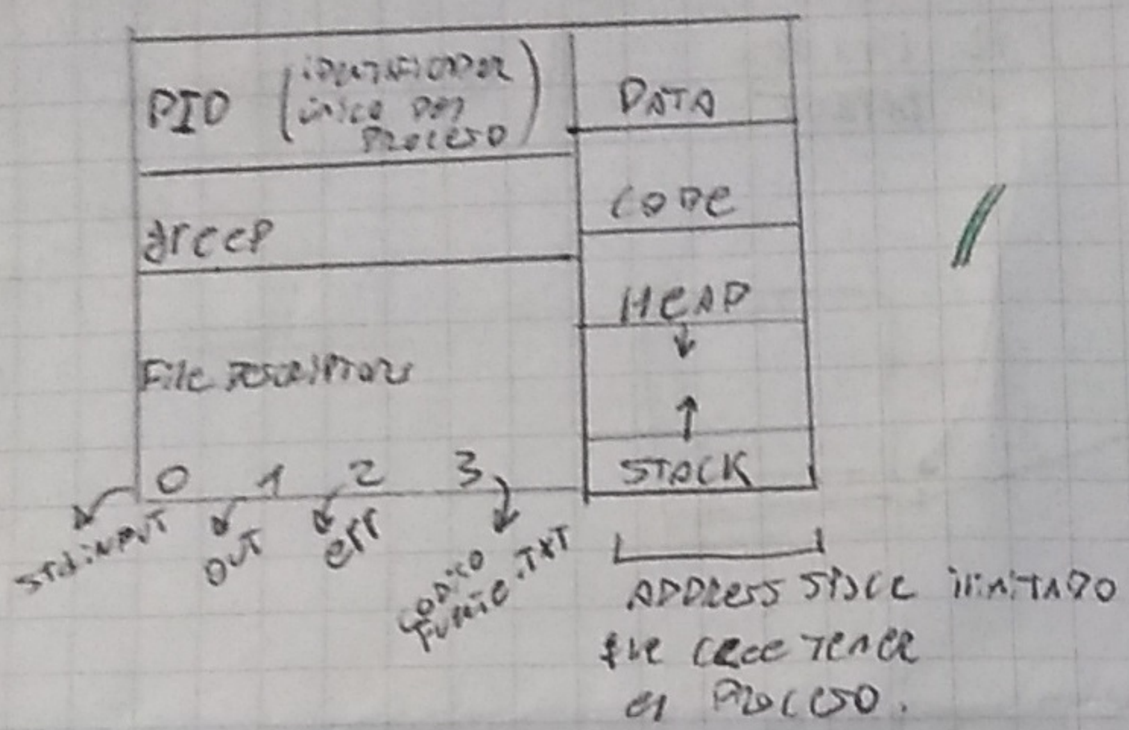
Concurrencia y Scheduling (10 ptos.)

a. Explique con un ejemplo MLFQ.

b. ¿Cuáles de los siguientes mecanismos son compartidos entre threads de un mismo programa?

- [ ] Stack Segment
- [x] File descriptors
- [x] Registros de CPU — Cada thread tiene su propio Register.
- [ ] Heap
- [ ] Code segment
- [ ] METADATA del Thread
- [ ] Data Segment
- [x] Signals

1) a) La definición de un proceso varía según cada autor pero en general se puede decir que es un programa en ejecución con permisos restringidos. a través de la memoria virtual un proceso tiene su propio. stack, heap y text. donde se crea la ilusión que tiene todo el espacio disponible para sí mismo. (es un proceso puede tener la dirección 0 del stack cuando la realidad es distinta) se puede explicar su estructura, más fácilmente con el sig. diagrama:



- Privilegios
- Estados
- Info. privilegios
- señales

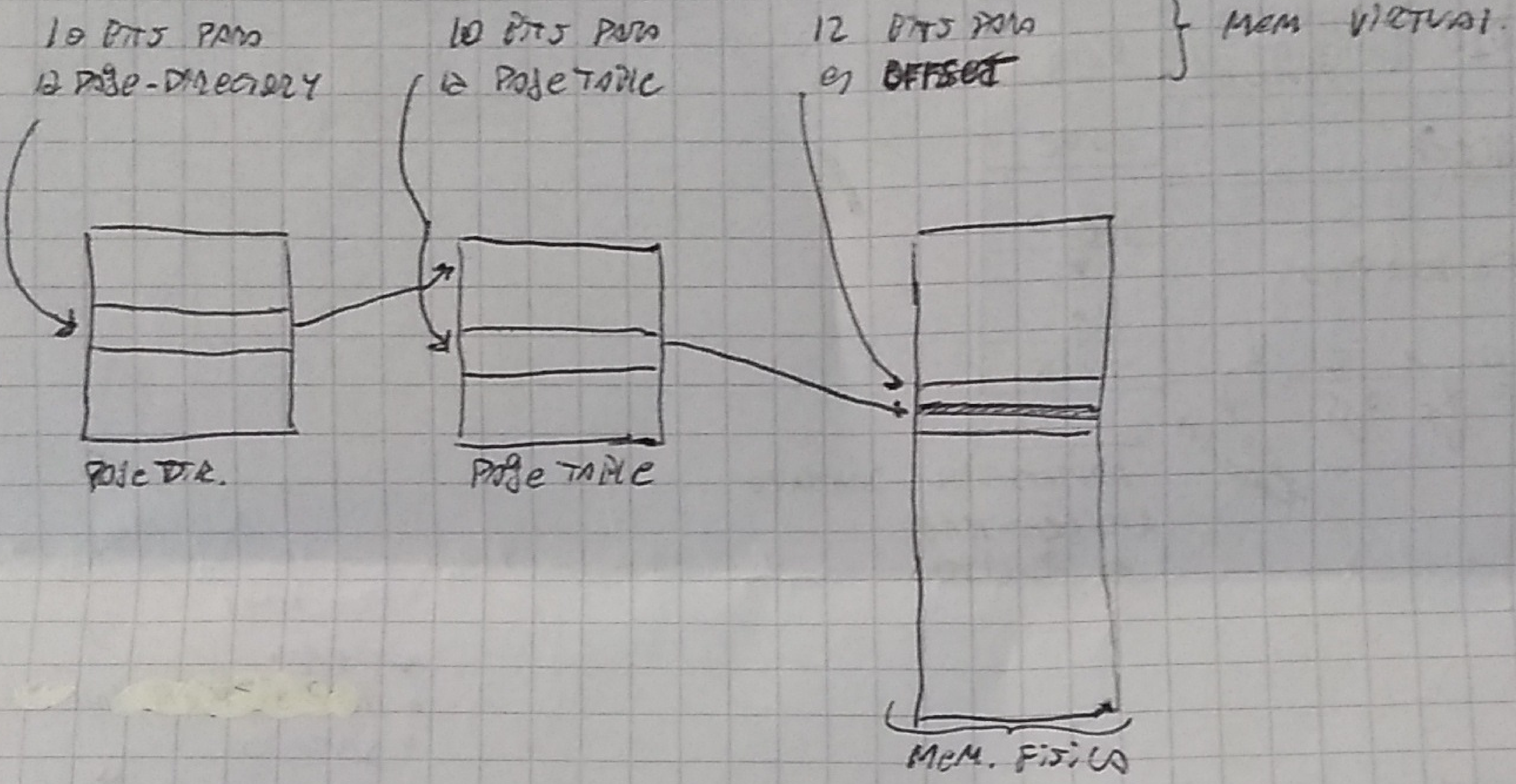
b) El kernel es la comunicación entre el hardware y el "software" todo aquel recurso que intente acceder al hardware está regulado por el kernel. En sistemas unix-like hay 4 capas, llamados rings. (en distintos niveles de privilegios, con ring 0 siendo el kernel y ring 3 el lado que usa el usuario. de esta forma no es posible para un proceso acceder sin antes el permiso del kernel. de esta forma gestionan ajenos. si no estuviese esta capa de protección, como ocurre por ejemplo en windows, donde existe la ejecución directa (un proceso se ejecuta directamente en el CPU que si bien es más rápido no hay nada que impida a un proceso de acceder a recursos no permitidos por el S.O.

se puede pasar de un nivel a otro con interrupciones del sistema (ring 3 → 0) y retorno a estas interrupciones (ring 0 → 3)

2) El Table Lookaside Buffer es un cache que permite solucionar problemas en la paginación siguiendo los principios de localidad espacial (si se accede a un recurso hay más chances que se acceda a un recurso cercano a él) y localidad temporal (si se accede a un recurso hay más probabilidades que se vuelva a acceder al mismo. ej: un loop)

y la paginación a dos niveles es una forma de abstracción de la mem. física utilizando la mem. virtual, se encarga del mapeo.

ej. para una dir de 32 bits.



b) Paso de hexa. a decimal para mayor facilidad

0 = 0
⋮
9 = 9
10 = A
11 = B
12 = C
13 = D
14 = E
15 = F

1) • 0000 0000 0000 0000 0000 0000 0000 0000 } Dir. virtual

seg. table | page table A | offset.

B   0 (F000 + 0

→ 0000F000 } Dir física (hexa) **No**

Al revés era → F000 0000

3) • 0001 0000 0000 0002 0002 0000 0000 0002

A | 2 (BEEF) |

$$\begin{array}{r} B\ E\ E\ F \\ +\ 2\ 0\ 0\ 2 \\ \hline D\ E\ F\ 7 \end{array}$$

→ 1002DEF7 } Dir física

2) • 0002 0000 0000 0002 0002 0000 0000 0002 → Dirección inválida **No**

invalid

• 0000 0000 0000 0001 0005 0005 0005 0005

B | 1 (D8BF) |

$$\begin{array}{r} D\ 8\ B\ F \\ +\ 5\ 5\ 5\ 5 \\ \hline 1\ 8\ E\ 7\ 9 \end{array}$$

overflow!

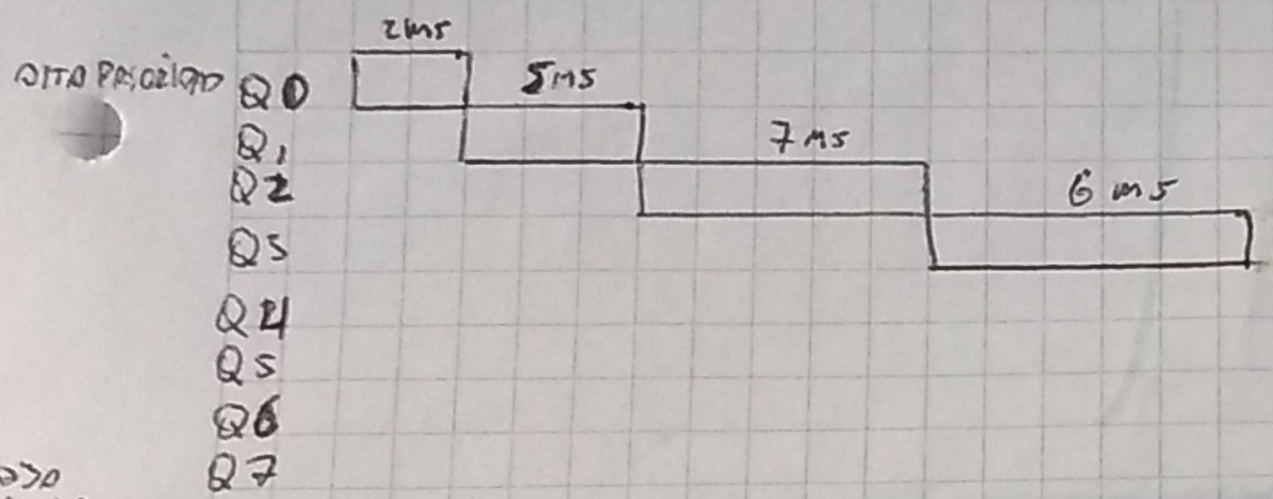
→ Dirección inválida **No**

NOTA

A GRANDES RASGOS

3) a) MULTI LEVEL FEEDBACK QUEUE SE TRATA DE UN ALGORITMO PARA LA ORGANIZACIÓN Y USO DE RECURSOS DE LA CPU QUE DISTRIBUYE LOS PROGRAMAS EN EJECUCIÓN SON 7 COLAS CON DISTINTOS NIVELES DE PRIORIDAD DONDE UN PROGRAMA SELO LLEVADO DE UNA A OTRA DEPENDIENDO DE CUANTO TARDE SU EJECUCIÓN.

POR EJEMPLO SI LA PRIMER COLA TIENE UN ~~TIME~~ TIME SLICE DE 2MS Y EN CADA COLA SE INCREMENTA EN 2 MS/C. SI EL TIEMPO DE EJECUCIÓN TOTAL DEL PROGRAMA SON 20 MS ENTONCES.



2+5+7+6 = 20

EL PROGRAMA SE TERMINARÍA DE EJECUTAR EN LA COLA 4

ESTE ALGORITMO OPTIMIZA EL TURNAROUND TIME Y MINIMIZA EL RESPONSE TIME

tiene 5 reglas

- EL PROCESO CON MÁS PRIORIDAD SE EJECUTA PRIMERO. ej) P(A) > P(B) ∴ EJEC (A) Y NO a B.
- SI DOS PROCESOS TIENEN LA MISMA PRIORIDAD SE EJECUTA ROUND ROBIN. ej)
- CUANDO ENTRA UN PROCESO NUEVO TIENE LA MÁS ALTA PRIORIDAD
- UNA VEZ QUE UN PROCESO USÓ SU TIEMPO DISPONIBLE EN UNA COLA PASA A LA SIGUIENTE DE MENOR PRIORIDAD
- CADA CIERTO TIEMPO LOS PROCESOS VUELVEN A TENER MÁX. PRIORIDAD (BOOST)