Practica 2 (Procesos) - Resolucion

1.

Responda en forma sintética sobre los siguientes conceptos:

a. ¿Qué es un Sistema Operativo?

b. Enumere qué componentes/aspectos del Hardware son necesarios para cumplir los objetivos de un Sistema Operativo.

c. Enumere componentes de un Sistema Operativo

d. ¿Que es una llamada al sistema (*system call*)? ¿Cómo es posible implementarlas?

e. Defina y diferencie Programa y Proceso.

f. ¿Cuál es la información mínima que el Kernel debe tener sobre un proceso? ¿En qué estructura de datos asociada almacena dicha información?

g. ¿Qué objetivos persiguen los algoritmos de planificación (scheduling). h. ¿Qué significa que un algoritmo de scheduling sea apropiativo o no apropiativo (Preemptive o Non-Preemptive)?

i. ¿Qué tareas realizan los siguientes módulos de planificación?: i. Short Term Scheduler

ii. Long Term Scheduler

iii. Medium Term Scheduler

j. ¿Qué tareas realiza el Dispatcher? ¿Y el Loader?

k. ¿Qué significa que un proceso sea “CPU Bound” y “I/O Bound”?

l. ¿Cuáles son los estados posibles por los que puede atravesar un proceso? ¿Qué representa que un proceso se encuentre en los estados enumerados? Utilizando un diagrama explique las transiciones entre los estados.

m. ¿Cuáles de los schedulers mencionados anteriormentese encargan de las transiciones entre los estados enumerados?

n. Defina Tiempo de retorno (TR) y Tiempo de espera (TE) para un proceso.
o. Defina Tiempo Promedio de Retorno (TPR) y Tiempo promedio de espera (TPE) para un lote de procesos.

p. Defina tiempo de respuesta.

2.

Para los siguientes algoritmos de scheduling:

a. Explique su funcionamiento mediante un ejemplo.

b. ¿Alguno de ellos cuentan con parámetros para su funcionamiento?

c. Cual es el más adecuado según los tipos de procesos y/o SO.

d. Cite ventajas y desventajas de su uso.

➢ FCFS (First Come First Served)

-El primero en llegar es el primero en atenderse, o el mas viejo en la CPU en caso de haber muchos. Procesos cortos pueden sufrir largas esperas ya que no hay favoritismo para ningun tipo de prioridad/tardanza de CPU

➢ SJF (Shortest Job First)

-El trabajo mas corto es el que primero se atiende. Los procesos largos pueden sufrir starvation ya que pueden tardar mucho en ser atendidos o incluso nunca serlos si siempre llegan procesos con menor tiempo de CPU que ellos

➢ Round Robin

-Funciona con un Quantum que establece las unidades de tiempo que le corresponden equittativamente a cada proceso. Se maneja una queue para encolarlos una vez el quantum llega a 0 y en caso de empate se elige el proceso mas antiguo en el programa. Puede sufrir problemas de eficiencia con Quantums muy chicos debido a la alta cantidad de cambios de contexto y con Quantums muy grandes se comporta como un FCFS

➢ Prioridades

-Maneja N cola de prioridades para los N procesos que hay, cada proceso se encola en su cola de prioridades y se van ejecutando segun su jerarquia. Pueden sufrir inanicion procesos con prioridad muy baja ya que puede que nunca sean atendidos (se resuelve con aging y escalarlo a prioridades mas altas luego de un determinado tiempo)

3.

Dado el siguiente lote de procesos en el que todos arriban al sistema en el instante 0 (cero):

Job	Unidades de CPU
1	7
2	15
3	12
4	4
5	9

a. Realice los diagramas de Gantt según los siguientes algoritmos scheduling:

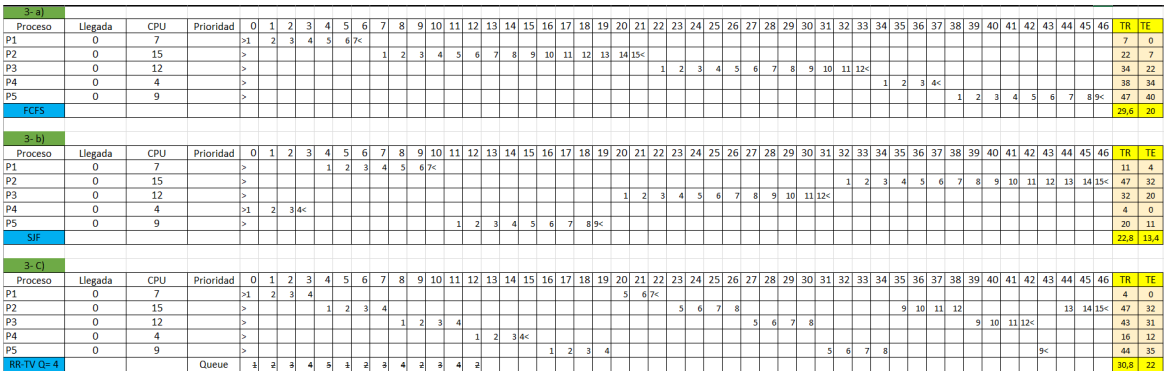
i. FCFS (First Come, First Served)

ii. SJF (Shortest Job First)

iii. Round Robin con quantum = 4

b. Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.

c. En base a los tiempos calculados compare los diferentes algoritmos.



3- a)

Proceso	Llegada	CPU	Prioridad	TR	TE
P1	0	7		7	0
P2	0	15		22	7
P3	0	12		34	22
P4	0	4		38	34
P5	0	9		47	40
FCFS				29,6	20

3- b)

Proceso	Llegada	CPU	Prioridad	TR	TE
P1	0	7		11	4
P2	0	15		47	32
P3	0	12		32	20
P4	0	4		4	0
P5	0	9		20	11
SJF				22,8	13,4

3- C)

Proceso	Llegada	CPU	Prioridad	TR	TE
P1	0	7		4	0
P2	0	15		47	32
P3	0	12		43	31
P4	0	4		16	12
P5	0	9		44	35
RR-TV Q= 4			Queue	30,8	22

Ejer 3 diagramas.xlsx

TIMER FIJO NO SE TOMA EN PRACTICA

4.

Dado el siguiente lote procesos:

Job	Llegada	Unidades de CPU
1	0	4
2	2	6
3	3	4
4	6	5
5	8	2

a. Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:

i. FCFS (First Come, First Served)

ii. SJF (Shortest Job First)

iii. Round Robin con quantum = 1

iv. Round Robin con quantum = 6 (igual pero con 6 xd)

b. Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.

c. En base a los tiempos calculados compare los diferentes algoritmos.

d. En el algoritmo Round Robin, qué conclusión se puede sacar con respecto al valor del quantum. e. ¿Para el algoritmo Round Robin, en qué casos utilizará un valor de quantum alto y qué ventajas y desventajas obtendría?

-Si el Quantum es muy bajo hay muchos cambios de contexto (ineficiencia) pero es mas interactivo
-Si el quantum es muy alto se comporta casi com FCFS, menos cambio de contexto y mayor tiempo de respuesta

4- a)																								
Proceso	Llegada	CPU	Prioridad	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		>1		2	3	4<																	4	0
P2	2	6				>			1	2	3	4	5	6<											8	2
P3	3	4					>							1	2	3	4<								11	7
P4	6	5								>								1	2	3	4	5<			13	8
P5	8	2										>											1	2<	13	11
FCFS																									9,8	5,5

4- b)																								
Proceso	Llegada	CPU	Prioridad	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		>1		2	3	4<																	4	0
P2	2	6				>													1	2	3	4	5	6<	19	13
P3	3	4					>		1	2	3	4<													4	0
P4	6	5								>			1	2	3	4	5<								7	2
P5	8	2										>						1	2<						7	5
SJF																									8,2	4

4- C)																								
Proceso	Llegada	CPU	Prioridad	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		>1		2	3	4<																	5	1
P2	2	6				>	1			2			3				4				5		6<		18	12
P3	3	4					>		1		2			3				4<							12	8
P4	6	5								>		1				2				3		4		5<	14	9
P5	8	2										>			1				2<						8	6
RR-TV Q= 1			Queue	1	1	2	1	3	2	3	4	2	3	5	4	2	3	5	4	2	4	2	4		11	7,2

EN EL ROUND ROBIN CUANDO HACE EL QUANTUM EL P3 Y LLEGA EL 4 A LA READY QUEUE SE DEBERIA PONER EL 3 ANTES QUE EL 4 PORQUE ESTA ANTES EN EL PROGRAMA

Ejer 4 diagramas.xlsx

5.

Una variante al algoritmo SJF es el algoritmo SJF apropiativo o SRTF (Shortest Remaining Time First):

a. Realice el diagrama de Gantt para este algoritmo según el lote de trabajos del ejercicio 4.

b. ¿Nota alguna ventaja frente a otros algoritmos?

5 a-b)

Proceso	Llegada	CPU	Prioridad	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		>1	2	3	4<																		4	0
P2	2	6				>													1	2	3	4	5	6<	19	13
P3	3	4					>	1	2	3	4<														5	1
P4	6	5								>				1	2	3	4	5<							9	4
P5	8	2										>1	2<												2	0
SRTF			Queue	1	2	3	4	5																	7,8	3,6

Ejer 5 diagramas.xlsx

6.

Suponga que se agregan las siguientes prioridades al lote de procesos del ejercicio 5, donde un menor número indica mayor prioridad:

Job	Prioridad
1	3
2	4
3	2
4	1
5	2

a. Realice el diagrama de Gantt correspondiente al algoritmo de planificación por prioridades según las variantes:

i. No Apropiativa

ii. Apropiativa

b. Calcule el TR y TE para cada job así como el TPR y el TPE.

c. ¿Nota alguna ventaja frente a otros algoritmos? ¿Bajo qué circunstancias lo utilizaría y ante qué situaciones considera que la implementación de prioridades podría no ser de mayor relevancia?

-Se usan de manera efectiva para tareas que el sistema operativo necesita ejecutar de manera inmediata y los usaria en casos de que se requieren dar prioridad a ciertos procesos que son mas importantes que otros ya que se adaptan mejor en nuestro sistema al llevar varias colas de prioridades

6)																									
Proceso	Llegada	CPU	Prioridad	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4	3	>1	2	3												4<							15	11
P2	2	6	4			>													1	2	3	4	5	6<	19	13
P3	3	4	2				>1	2	3						4<										9	5
P4	6	5	1							>1	2	3	4	5<											5	0
P5	8	2	2									>				1	2<								6	4
PRIORIDADES APROPIATIVO			Queue 1	4																					11	6,6
			Queue 2	~~3~~	~~3~~	~~5~~																				
			Queue 3	~~1~~																						
			Queue 4	~~2~~																						

6)																									
Proceso	Llegada	CPU	Prioridad	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4	3	>1	2	3	4<																		4	0
P2	2	6	4			>													1	2	3	4	5	6<	19	13
P3	3	4	2				>	1	2	3	4<														5	1
P4	6	5	1							>		1	2	3	4	5<									7	2
P5	8	2	2									>					1	2<							7	5
PRIORIDADES NO APROPIATIVO			Queue 1	4																					8,4	4,2
			Queue 2	~~3~~	~~5~~																					
			Queue 3	~~1~~																						
			Queue 4	~~2~~																						

Ejer 6 diagramas.xlsx

7. Inanición (*Starvation*)

a. ¿Qué significa?

b. ¿Cuál/es de los algoritmos vistos puede provocarla?

c. ¿Existe alguna técnica que evite la inanición para el/los algoritmos mencionados en el inciso b?

-Significa que esta en una situacion donde no puede ejecutarse debido a que la competencia por otros recursos del sistema entre diferentes

procesos es siempre tomada en cuenta por otro y no este mismo. Sucede por ejemplo cuando en un **algoritmo SJF** un proceso de tiempo largo de CPU puede nunca ser atendido si siempre llegan procesos que tienen menor tiempo que el, el **STRF** lo mismo, y el de **Prioridades** lo genera cuando tu proceso tiene baja prioridad y siempre llegan a la cola procesos con prioridad mas alta

8.

Los procesos, durante su ciclo de vida, pueden realizar operaciones de I/O como lecturas o escrituras a disco, cintas, uso de impresoras, etc. El Kernel mantiene para cada dispositivo, que se tiene en el equipo, una cola de procesos que espera por la utilización del mismo (al igual que ocurre con la Cola de Listos y la CPU, ya que la CPU es un dispositivo más).

Cuando un proceso en ejecución realiza una operación de I/O el mismo es expulsado de la CPU y colocado en la cola correspondiente al dispositivo involucrado en la operación.

El Kernel dispone también de un "I/O Scheduling" que administra cada cola de dispositivo a través de algún algoritmo (FCFS, Prioridades, etc.). Si al colocarse un proceso en la cola del dispositivo, la misma se encuentra vacía el mismo será atendido de manera inmediata, caso contrario, deberá esperar a que el Kernel lo seleccione según el algoritmo de scheduling establecido.

Los mecanismos de I/O utilizados hoy en día permiten que la CPU no sea utilizada durante la operación, por lo que el Kernel puede ejecutar otro proceso que se encuentre en espera una vez que el proceso bloqueado por la I/O se coloca en la cola correspondiente.

Cuando el proceso finaliza la operación de I/O el mismo retorna a la cola de listos para competir nuevamente por la utilización de la CPU.

Para los siguientes algoritmos de Scheduling:

➢ FCFS

➢ Round Robin con quantum = 2

Y suponiendo que la cola de listos de todos los dispositivos se administra mediante FCFS, realice los diagramas de Gantt según las siguientes situaciones:

a. Suponga que al lote de procesos del ejercicio 5 se agregan las siguientes operaciones de entrada salida:

Job	I/O (Recurso,Instante,Duración)
1	(R1, 2, 1)
2	(R2, 3, 1) (R2, 5, 2)
4	(R3, 1, 2) (R3, 3, 1)

10 a)																										
Proceso	Llegada	CPU	Prioridad	I/O (r,l,d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		(R1,2,1)	>1	2	R1	-		3	4<															7	3
P2	2	6		(R2,3,1) (R2,5,2)			>1	2	3	R2						4	5	R2	R2		6<					15	9
P3	3	4		-				>				1	2	3	4<											8	4
P4	6	5		(R3,1,2) (R3,3,1)							>							1	R3	R3		2	3	4	5<	15	10
P5	8	2		-									>						1	2<						8	6
FCFS				Queue	~~1~~	~~2~~	~~1~~	~~3~~	~~2~~	4	~~5~~	~~2~~	4													10,6	6,4
				Queue R1	~~1~~																						
				Queue R2	~~2~~	~~2~~																					
				Queue R3	4																						

10 a)																										
Proceso	Llegada	CPU	Prioridad	I/O (r,l,d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		(R1,2,1)	>1	2	R1		3	4<																6	2
P2	2	6		(R2,3,1) (R2,5,2)			>1	2					3	R2					4	5	R2	R2	6<			17	11
P3	3	4		-				>			1	2<			3	4<										9	5
P4	6	5		(R3,1,2) (R3,3,1)							>			1	R3	R3					2	3	R2	4	5<	15	10
P5	8	2		-									>				1	2<								6	4
RR-TV Q=2				Queue	~~1~~	~~2~~	~~1~~	~~3~~	~~2~~	4	~~3~~	~~5~~	~~2~~	4	~~2~~	4										10,6	6,4
				Queue R1	~~1~~																						
				Queue R2	~~2~~	~~2~~																					
				Queue R3	4	4																					

b. Suponga que al lote de procesos del ejercicio 5 se agregan las siguientes operaciones de entrada salida:

Job	I/O (Recurso,Instante,Duración)
1	(R1, 2, 3) (R1, 3, 2)
2	(R2, 3, 2)
3	(R2, 2, 3)
4	(R1,1,2)

10 b)																										
Proceso	Llegada	CPU	Prioridad	I/O (r,l,d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		(R1,2,3) (r1,3,2)	>1	2	R1	R1	R1			3	R1	R1					4<							15	11
P2	2	6		(R2,3,2)			>1	2	3	R2	R2			4	5	6<										10	5
P3	3	4		(R2,2,3)				>		1	2	R2	R2	R2						3	4<					14	10
P4	6	5		(R1,1,2)							>		1	ER1	R1	R1						2	3	4	5<	15	10
P5	8	2		-									>				1	2<								6	4
FCFS				Queue	~~1~~	~~2~~	~~3~~	~~1~~	4	~~2~~	~~5~~	~~1~~	~~3~~	4												12	8
				Queue R1	~~1~~	~~1~~	4																				
				Queue R2	~~2~~	~~3~~																					

10 b)																										
Proceso	Llegada	CPU	Prioridad	I/O (r,l,d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		(R1,2,3) (r1,3,2)	>1	2	R1	R1	R1			3	R1	R1	4<											11	7
P2	2	6		(R2,3,2)			>1	2	3	R2	R2		4	5								6<				16	10
P3	3	4		(R2,2,3)				>		1	2	R2	R2	R2		3	4<									12	8
P4	6	5		(R1,1,2)							>							1	2				3	4	5<	15	10
P5	8	2		-									>							1	2<					8	6
RR-TV Q=2				Queue	~~1~~	~~2~~	~~2~~	~~3~~	~~1~~	~~2~~	~~1~~	~~3~~	4	~~5~~	~~2~~	4										12,4	8,2
				Queue R1	~~1~~	~~1~~	4																				
				Queue R2	~~2~~	~~3~~																					

Ejer 8 diagramas.xlsx

9.

Algunos algoritmos pueden presentar ciertas desventajas cuando en el sistema se cuenta con procesos ligados a CPU y procesos ligados a entrada salida. Analice las mismas para los siguientes algoritmos:

a. Round Robin

-RR para procesos CPU BOUND genera desperdicio de tiempo de CPU si se usa Timer variable y un proceso termina antes que su tiempo de Quantum

-Para procesos I/O bound es el mejor para atender las tareas del sistema pero puede generar inanicion/ineficiencia de CPU

b. SRTF (Shortest Remaining Time First)

-SRTF para CPU BOUND genera inanicion de procesos que tengan larga duracion debido a lo explicado previamente, ademas de muchos context switch si en la cola hay procesos que tienen un tiempo de ejecucion bajo y similiar

-Para procesos I/O bound son beneficicios

10.

Para equiparar la desventaja planteada en el ejercicio 10, se plantea la siguiente modificación al algoritmo:

Algoritmo VRR (Virtual Round Robin): Este algoritmo funciona igual que el Round Robin, con la diferencia que cuando un proceso regresa de una I/O se coloca en una cola auxiliar. Cuando se tiene que tomar el próximo proceso a ejecutar, los procesos que se encuentran en la cola auxiliar tienen prioridad sobre los otros. Cuando se elije un proceso de la cola auxiliar se le otorga el procesador por tantas unidades de tiempo como le faltó ejecutar en su ráfaga de CPU anterior, esto es, se le otorga la CPU por un tiempo que surge entre la diferencia del quantum original y el tiempo usado en la última ráfaga de CPU.

a. Analice el funcionamiento de este algoritmo mediante un ejemplo. Marque en cada instante en que cola se encuentran los procesos.

b. Realice el ejercicio 9)a) nuevamente considerando este algoritmo, con un quantum de 2 unidades

10)																										
Proceso	Llegada	CPU	Prioridad	I/O (r,l,d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	TR	TE
P1	0	4		(R1,2,1)	>1	2	R1		3	4<																6	2
P2	2	6		(R2,3,1) (R2,5,2)			>1	2			3	R2		4	5	R2	6<									11	5
P3	3	4		-				>				1	2									3	4<			16	12
P4	6	5		(R3,1,2) (R3,3,1)							>					1	R3	R3		2	3	R3		4	5<	15	10
P5	8	2		-									>					1	2<							6	4
VRR Q=2				Queue	~~1~~	~~2~~	~~2~~	~~3~~	~~4~~	~~3~~	~~2~~	~~5~~	~~3~~													11	6,6
				Queue R1	~~1~~																						
				Queue R2	~~2~~	~~2~~																					
				Queue R3	~~4~~	~~4~~																					
				Queue AUX	~~1~~	~~2~~	~~2~~	~~4~~	~~4~~																		

(NADA SEGURO)

Ejer 10 diagramas.xlsx

11. **Suponga que un Kernel utiliza un algoritmo de VRR para planificar sus procesos. Para ello, el quantum es representado por un contador, que es decrementado en 1 unidad cada vez que ocurre una interrupción de reloj. ¿Bajo este esquema, puede suceder que el quantum de un proceso nunca llegue a 0 (cero)? Justifique su respuesta.**

-Teniendo en cuenta que solo decrementa cuando ocurre una interrupcion a reloj **si** se puede dar el caso en el que el Quantum de un proceso nunca llegue a 0 si este mismo termina su rafaga de CPU antes de que termine su tiempo de Quantum, en ese caso no habria interrupcion y no se decrementaria el contador

12. **El algoritmo SJF (y SRTF) tiene como problema su implementación, dada la dificultad de conocer la duración de la próxima ráfaga de CPU. Es posible realizar una estimación de la próxima, utilizando la media de las ráfagas de CPU para cada proceso. Así, por ejemplo, podemos tener la siguiente fórmula (1):**

$$S_{n+1} = \frac{1}{n} T_n + \frac{n-1}{n} S_n$$

Donde:

T*i* = duración de la ráfaga de CPU i-ésima del proceso.

S*i* = valor estimado para el i-ésimo caso

S*i* = valor estimado para la primera ráfaga de CPU. No es calculado.

a. Suponga un proceso cuyas ráfagas de CPU reales tienen como duración: 6, 4, 6, 4, 13, 13, 13. Calcule qué valores se obtendrían como estimación para las ráfagas de CPU del proceso si se utiliza la fórmula 1, con un valor inicial estimado de S1=10.

La fórmula 1 le da el mismo peso a todos los casos (siempre calcula la media). Es posible reescribirla permitiendo darle un peso mayor a los casos más recientes y menor a casos viejos (o viceversa). **Se plantea la siguiente fórmula 2:**

$$S_{n+1} = \alpha T_n + (1 - \alpha) S_n$$

Con 0 < *α* < 1.

b. Analice para qué valores de *α* se tienen en cuenta los casos más recientes.

c. Para la situación planteada en a) calcule qué valores se obtendrían si se utiliza la fórmula 2con *α* = 0, 2; *α* = 0, 5 y *α* = 0,8.

d. Para todas las estimaciones realizadas en ay c¿Cuál es la que más se asemeja a las ráfagas de CPU reales del proceso?

13. **Colas Multinivel**

Actualmente los algoritmos de planificación vistos se han ido combinando para formar algoritmos más eficientes. Así surge el algoritmo de Colas Multinivel, donde la cola de procesos listos es dividida en varias colas, teniendo cada una su propio algoritmo de planificación.

a. Suponga que se tiene dos tipos de procesos: *Interactivos* y *Batch*. Cada uno de estos procesos se coloca en una cola según su tipo. ¿Qué algoritmo de los vistos utilizará para administrar cada una de estas colas?. A su vez, se utiliza un algoritmo para administrar cada cola que se crea. Así, por ejemplo, el algoritmo podría determinar mediante prioridades sobre qué cola elegir un proceso.

-Procesos interactivos → RR ; Procesos Batch → SJF si los tiempos son conocidos sino FCFS

b. Para el caso de las dos colas vistas en a: ¿Qué algoritmo utilizaría para planificarlas?

-Algoritmo de Prioridades

14. Suponga que en un Kernel se utiliza un algoritmo de planificación de colas multinivel. El mismo cuenta con 3 colas de procesos listos, en las que los procesos se encolan en una u otra según su prioridad. Hay 3 prioridades (1 , 2 , 3), donde un menor número indica mayor prioridad. Se utiliza el algoritmo de prioridades para la administración entre las colas.

Se tiene el siguiente lote de procesos a ser procesados con sus respectivas operaciones de I/O:

Job	Llegada	CPU	I/O (rec,ins,dur)	Prioridad
1	0	9	(R1, 4, 2) (R2, 6, 3) (R1, 8, 3)	1
2	1	5	(R3, 3, 2) (R3, 4, 2)	2
3	2	5	(R1, 4, 1)	3
4	3	7	(R2, 1, 2) (R2, 5, 3)	2
5	5	5	(R1, 2, 3) (R3, 4, 3)	1

Suponiendo que las colas de cada recurso (dispositivo de entrada/salida) se administran a través de FCFS y que cada cola de procesos listos se administra por medio de un algoritmo RR con un quantum de 3 unidades, realice un diagrama de Gantt:

a. Asumiendo que NO hay apropiación entre los procesos.

b. Asumiendo que hay apropiación entre los procesos.

14) N-A

Proceso	Llegada	CPU	Prioridad	I/O (r,I,d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	TR	TE
P1	0	9	1	(R1,4,2) (R2,6,3) (R1,8,3)	>1	2	3	4	R1	R1				5	6	R2	R2	R2	7	8	R1	R1	R1			9<												22	13
P2	1	5	2	(R3,3,2) (R3,4,2)		>				1	2	3	R3	R3							4	R3	R3				5<												22	17
P3	2	5	3	(R1,4,1)			>																						1	2	3				4	R1	5<	30	25
P4	3	7	2	(R2,1,2) (R2,5,3)				>								1	ER2	ER2	R2	R2			2	3	4			5	R2	R2	R2	6	7<					26	19
P5	5	5	1	(R1,2,3) (R3,4,3)						>		1	2	R1	R1	R1	3	4	R3	R3		5<																13	8
COLA Disp	FCFS			Queue Prioridad 1	~~1~~	~~1~~	~~5~~	~~1~~	~~5~~	~~1~~	~~5~~	~~1~~																											22,6	16,4
COLA Prio	RR TV Q=3			Queue Prioridad 2	~~2~~	~~4~~	~~2~~	~~4~~	~~4~~																															
				Queue Prioridad 3	~~3~~	~~3~~	~~3~~																																	
				Queue R1	~~1~~	~~5~~	~~1~~	~~3~~																															
				Queue R2	~~1~~	~~4~~	~~4~~																																	
				Queue R3	~~2~~	~~5~~																																		

14) A

Proceso	Llegada	CPU	Prioridad	I/O (r,I,d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	TR	TE
P1	0	9	1	(R1,4,2) (R2,6,3) (R1,8,3)	>1	2	3	4	R1	R1		5	6	R2	R2	R2	7	8	R1	R1	R1	9<															18	9
P2	1	5	2	(R3,3,2) (R3,4,2)		>			1									2					3	R3	R3		4	R3	R3	5<								25	20
P3	2	5	3	(R1,4,1)			>																					1	2				3	4	R1		30	25
P4	3	7	2	(R2,1,2) (R2,5,3)				>						1	ER2	ER2	R2	R2			2			3	4	5	R2	R2	R2		6	7<					25	18
P5	5	5	1	(R1,2,3) (R3,4,3)						>1	2	R1	R1		3	4	R3	R3	R3	5<																		11	6
COLA Disp	FCFS			Queue Prioridad 1	~~1~~	~~1~~	~~5~~	~~1~~	~~5~~	~~1~~	~~5~~	~~1~~																											21,8	15,6
COLA Prio	RR TV Q=3			Queue Prioridad 2	~~2~~	~~4~~	~~2~~	~~4~~	~~2~~	~~4~~	~~2~~	~~2~~	~~4~~																											
				Queue Prioridad 3	~~3~~	~~3~~	~~3~~																																	
				Queue R1	~~1~~	~~5~~	~~1~~	~~3~~																															
				Queue R2	~~1~~	~~4~~	~~4~~																																	
				Queue R3	~~5~~	~~2~~	~~2~~																																	

Ejer 14 diagramas.xlsx

15.

En el esquema de Colas Multinivel, cuando se utiliza un algoritmo de prioridades para administrar las diferentes colas los procesos pueden sufrir starvation.

La técnica de envejecimiento se puede aplicar a este esquema, haciendo que un proceso cambie de una cola de menor prioridad a una de mayor prioridad, después de cierto periodo de tiempo que el mismo se encuentra esperando en su cola. Luego de llegar a una cola en la que el proceso llega a ser atendido, el mismo retorna a su cola original.

Por ejemplo: Un proceso con prioridad 3 está en cola su cola correspondiente. Luego de X unidades de tiempo, el proceso se mueve a la cola de prioridad 2. Si en esta cola es atendido, retorna a su cola original, en caso contrario luego de sucederse otras X unidades de tiempo el proceso se mueve a la cola de prioridad 1. Esta última acción se repite hasta que el proceso obtiene la CPU, situación que hace que el mismo vuelva a su cola original.

Para los casos a y b del ejercicio 15 realice el diagrama de Gantt considerando además que se tiene un envejecimiento de 4 unidades.

16.

La situación planteada en el ejercicio 16, donde un proceso puede cambiar de una cola a otra, se la conoce como Colas Multinivel con Realimentación.

Suponga que se quiere implementar un algoritmo de planificación que tenga en cuenta el tiempo de ejecución consumido por el proceso, penalizando a los que más tiempo de ejecución tienen. (Similar a la tarea del algoritmo SJF que tiene en cuenta el tiempo de ejecución que resta).

Utilizando los conceptos vistos de Colas Multinivel con Realimentación indique qué colas implementaría, qué algoritmo usaría para cada una de ellas así como para la administración de las colas entre sí.

Tenga en cuenta que los procesos no deben sufrir inanición.

Usaria 3 colas, una por cada prioridad siendo mayor la prioridad segun menor sea el numero. Los procesos mas cortos irian a la cola de prioridad mas alta y los procesos mas largos irian a las colas de baja prioridad ya que dice que hay que penalizarlos. Para manejar el tiempo de CPU usaria un RR chico para los procesos cortos y para los procesos largos podes manejarlos con aging o sino en vez de RR hacer un VRR y que los procesos largos vayan a colas de prioridad auxiliares cuando no terminen su quantum

17.

A cuáles de los siguientes tipos de trabajos:

- **cortos acotados por CPU**
- **cortos acotados por E/S**
- **largos acotados por CPU**
- **largos acotados por E/S**

benefician las siguientes estrategias de administración:

a. prioridad determinada estáticamente con el método del más corto primero (SJF).

-Beneficia a los cortos acotados por CPU y cortos acotados por E/S

b. prioridad dinámica inversamente proporcional al tiempo transcurrido desde la última operación de E/S.

-Beneficia a los Largos acotados por E/S y a los cortos acotados por E/S

18. **Analizar y explicar por qué si el quantum en Round-Robin se incrementa sin límite, el algoritmo de planificación se aproxima a FIFO.**

-Ocurre porque cuanto mas grande sea el quantum mas posibilidades tenes que el proceso use X tiempo de CPU determinado por el Quantum y en ese

tiempo llegue a terminar su uso completo de CPU sin ser expulsados por el algoritmo en si. No habria rotacion de algoritmos ya que terminarian su tiempo de CPU en su primera rafaga y se trataria como un FIFO

19. Dado los siguientes programas en un pseudo-código que simula la utilización de llamadas al sistema para crear procesos en Unix/Linux, indicar qué mensajes se imprimirán en pantalla (sin importar el orden en que saldrían):

```
a. Caso 1

printf(“hola”)

x = fork()

if x < 1 {

    execv(“ls”)

    printf(“mundo”)

    exit(0)

}

exit(0)

// IMPRIME 'HOLA' Y EL HIJO IMPRIME 'LS'

b. Caso 2

printf(“hola”)

x = fork()

if x < 1 {

    execv(“ps”)

    printf(“mundo”)
```

```
    exit(0)
}
execv(“ls”)
printf(“fin”)
exit(0)
// IMPRIME 'HOLA', HIJO IMPRIME 'PS', PADRE IMPRIME 'LS',
```

c. Caso 3

```
printf(“Anda a rendir el Primer Parcial de Promo!”)
newpid = fork()
if newpid == 0 {
    printf(“Estoy comenzando el Examen”)**
    execv(“ps”)**
    printf(“Termine el Examen”)
}
printf(“¿Como te fue?”)
exit(0)
printf(“Ahora anda a descansar”)
// IMPRIME ANDA A RENDIR,
// EL HIJO IMPRIME 'ESTOY COMENZANDO EL EXAMEN', HIJO IMPRIME
```

```
'PS'
// PADRE IMPRIME 'COMO TE FUE'
```

20.

Dado el siguiente programa en C, analice su objetivo sin necesidad de ejecutarlo. (Investigue el funcionamiento del iterador *for* en C para poder responder:)

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main ( void ) {

   int c ;

   pid_t pid ;

   printf( " Comienzo . : \ n " ) ;

   for ( c = 0 ; c < 3 ; c++ ) {

      pid = fork ( ) ;

   }

   printf ( " Proceso \n " ) ;

   return 0 ;

}
```

a. ¿Cuántas líneas con la palabra “Proceso” aparecen al final de la ejecución de este programa?

-Siendo que el for itera 3 veces (0,1,2) y cada vez que se entra se duplica el numero de proceesos deberian aparecer 8 lineas con la palabra ‘Proceso’

b. *¿El número de líneas es el número de procesos que han estado en ejecución?*.

-Si, corresponde a la cantidad de procesos que ‘sobrevivieron’ al bucle for y ejecutaron la linea del print

Ejecute el programa y compruebe si su respuesta es correcta. Modifique el valor del bucle for y compruebe los nuevos resultados.

21.

Modifiquemos el programa anterior. Ahora, además de un mensaje, vamos a añadir una variable y antes de finalizar el programa vamos a mostrar el valor de la misma:

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main ( void ) {

   int c ;

   pid_t pid ;

   int p = 0;

   printf( " Comienzo . : \ n " ) ;

   for ( c = 0 ; c < 3 ; c++ ) {

      pid = fork ( ) ;

   }
```

```
    p++;
    printf ( " Proceso %d\n ",p ) ;
    return 0 ;
}
```

a. ¿Qué valores se imprimirán en la consola?

-Se imprime Comienzo al prinicpio y luego cada proceso imprimira ‘Proceso 1’

b. ¿Todas las líneas tendrán el mismo valor o algunas líneas tendrán valores distintos?

-Todas las lineas van a tener el mismo valor porque la variable p es local a cada proceso y siempre empieza en 0 y se aumenta en 1

c. ¿Cuál es el valor (o valores) que aparece*?*. Compile y ejecute el programa y compruebe si su respuesta es correcta.

-Aparecera 8 veces ‘Proceso 1’ impreso, para cambiar esto lo cambiaria en el inciso d

d. Realice modificaciones al programa, por ejemplo: modifique el valor del bucle for, cambie el lugar dónde se incrementa la variable p, y compruebe los nuevos resultados.

-Moveria el p++ adentro del for para que los valores a imprimir sean diferentes segun el proceso que los toma y cuantas veces entra al for

22. ¿Qué es la MMU y qué funciones cumple?

-Es la unidad de manejo de memoria, un componente de HW que se encarga de traducir direcciones logcicas en memoria a direcciones fisicas donde se almacenaran

23. **¿Que representa el espacio de direcciones de un proceso?**

-Conjunto de direcciones de memoria virtual que un proceso puede utilizar y esta delimitado y aislado. Contiene el codigo del proceso, datos estaticos y

dinamicos, su stack, su heap

24. Explique y relacione los siguientes conceptos:

Dirección Lógica o Virtual

Dirección Física

-La direccion logica o virtual es la dierccion de memoria que un proceso utiliza en su codigo para acceder a datos o instrucciones que el conoce. La direccion fisica es la direccion real en memoria RAM donde se almacenan los datos o instrucciones. Se necesita hacer una traduccion a traves de la MMU y la tabla de paginas para, desde una direccion logica, poder determinar cual es su direccion fisica y asi accederla/utilizarla

25.

En la técnica de Particiones Múltiples, la memoria es dividida en varias particiones y los espacios de direcciones de los procesos son ubicados en estas, siempre que el tamaño del mismo sea menor o igual que el tamaño de la partición.

Al trabajar con particiones se pueden considerar 2 métodos (independientes entre sí):

Particiones Fijas

Particiones Dinámicas

a. Explique cómo trabajan estos 2 métodos. Cite diferencias, ventajas y desventajas.

-Particiones fijas: el sistema particiona de una forma establecida la memoria y queda fija durante todo el tiempo. Asigna un proceso en cada particion segun algun criterio. El problema es que genera fragmentacion interna

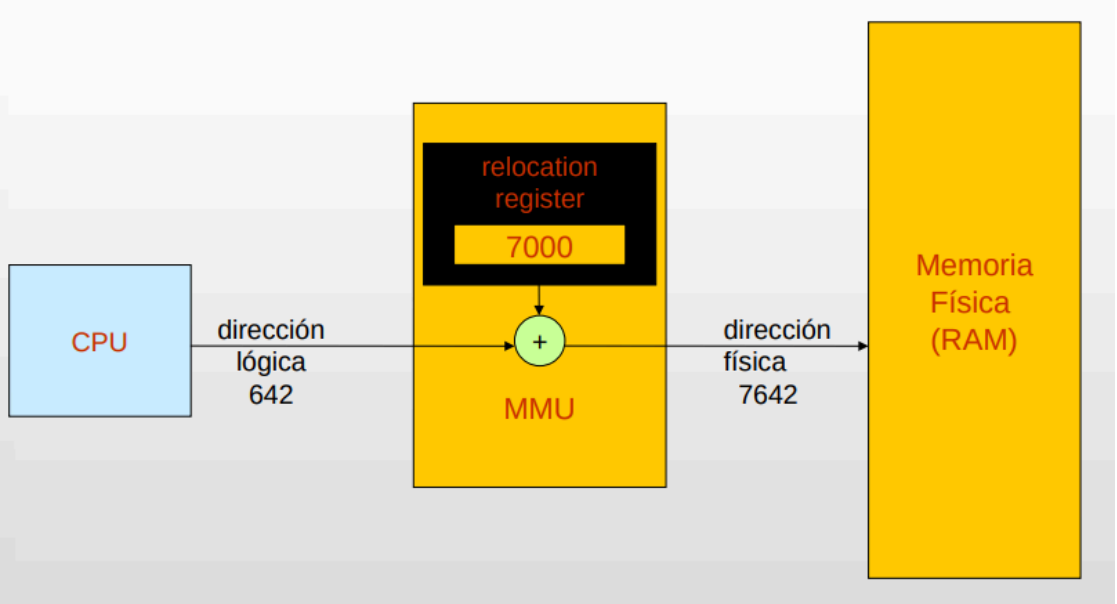
-Particiones dinamicas: el tamaño de las particiones varia y la cantidad de ellas tamien. Alojan procesos en cada particion esegun algun criterio pero esta particion se genera de forma dinamica al tamaño justo que necesita el proceso. El problema es que genera fragmentacion externa

b. ¿Qué información debe disponer el Kernel para poder administrar la memoria principal con estos métodos?

-El kernel debe contar con el tamaño de memoria total disponible, el registro base (direccion de comienzo del proceso) y registro limite

(direccion final del proceso). Luego la MMU traducira la direccion y el Kernel aplicara el desplazamiento a la direccion fisica

c. Realice un gráfico indicando cómo se realiza la transformación de direcciones lógicas a direcciones físicas.



26.

Al trabajar con particiones fijas, los tamaños de las mismas se pueden considerar:

➢ Particiones de igual tamaño.

➢ Particiones de diferente tamaño.

Cite ventajas y desventajas entre las alternativas.

-Particiones de igual tamaño: Distribuyen equitativamenete la memoria con facil implementacion y administracion. El problema es que no son felxibles al cambio y genera fragmentacion

-Particiones de diferente tamaños: Flexibles a ajustes de tamaño pero requiere de mayor control y administracion y una complejidad adicional. Genera una mayor posibilidad de fragmentacion externa

27.

Fragmentación. Ambos métodos de particiones presentan el problema de la fragmentación:

Fragmentación Interna (Para el caso de Particiones Fijas)

Fragmentación Externa (Para el caso de Particiones Dinámicas)

a. Explique a qué hacen referencia estos 2 problemas

-Fragmentacion interna se refiere a la parte de una particion que queda incompleta debido a que el proceso metido no completa la particion

-Fragmentacion externa referencia a los huecos que van quedando en la memotira a medida que los procesos terminan. Ocurre debido a que no se pueden encontrar una forma contigua de almacenar los procesos ya que esta fragmentacion puede estar exparsida por diferentes partes

b. El problema de la Fragmentación Externa es posible de subsanar. Explique una posible técnica que permita al Kernel mitigar este problema.

-Fragmentacion externa se solucionaria con compactacion

28. **Analice y describa cómo funcionan las siguientes técnicas de asignación de particiones: Best Fit, Worst Fit, First Fit y Next Fit. Tenga en cuenta que información debe mantener el Kernel. Indique, para los métodos de particiones dinámicas y fijas, con cuál técnica se obtienen mejores resultados y justifique.**

-First fit: busca desde el inicio de la memoria y selecciona el primer bloque que sea lo suficientemente grande para el proceso

-Best fit: busca desde el inicio de la memoria selecciona el bloque libre mas pequeño que pueda contener al proceso

-Worst fit: busca desde el inicio de la memoraiselecciona el bloque mas grande disponible y asigna el proceso ahi

-Next fit: selecciona el primer bloque que sea lo suficientemente grande para el proceso pero la busqueda empieza desde la ultima asignacion

29. Segmentación

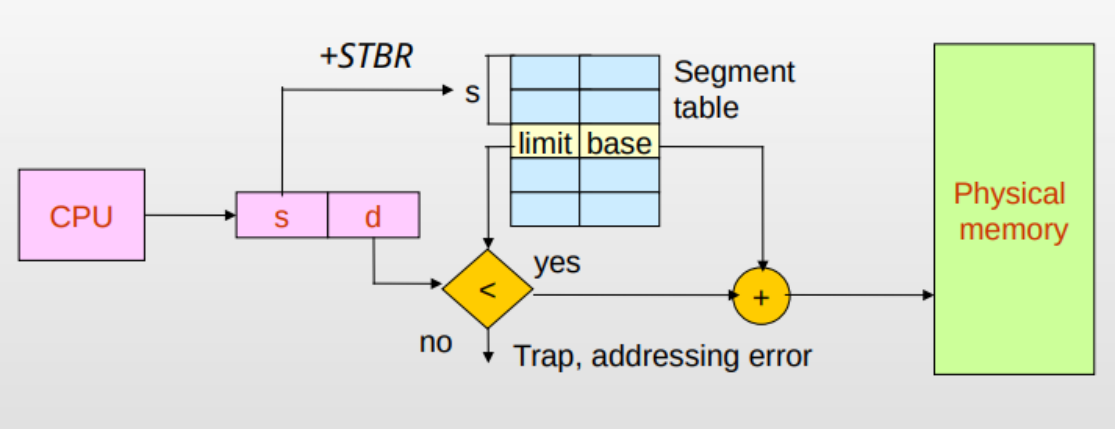
a. Explique cómo trabaja este método de asignación de memoria.

-Se basa en dividir al proceso en secciones separadas denominadas segmentos, siendo cada una de ellas una unidad logica como programa principal, procedimiento, funciones, etc. Logicamente el proceso esta organizado todo junto pero se sube a memoria principal dividido en segmentos (utilizando tabla de segmentos para que el kernel la asigne)

b. ¿Qué estructuras son necesarias en el Kernel para poder ejecutarse sobre un Hardware que utiliza segmentación?

-Se necesita de la tabla de segmentos, en la cual se encuentra la direccion fisica de comienzo y de fin de cada procesol (Base y Limit register)

c. Explique, utilizando gráficos, cómo son resueltas las direcciones lógicas a físicas.



d. En este esquema: ¿se puede producir fragmentación (interna y/o externa)?

-Externa

30.

Dado un esquema de segmentación donde cada dirección hace referencia a 1 byte y la siguiente tabla de segmentos de un proceso, traduzca, de corresponder, las direcciones lógicas indicadas a direcciones físicas. La Dirección lógica está representada por: **segmento:desplazamiento**

Segmento	Dir. Base	Tamaño
0	102	12500
1	28699	24300
2	68010	15855
3	80001	400

i.0000:9001 ii.0001:24301 iii.0002:5678 iv. 0001:18976 v. 0003:0

-i) Segmento: 0, Desplazamiento: 9001

12500 > 9001? Si

9001 + 102 = 9003

-ii) Segmento: 1, Desplazamiento: 24301

24300 > 24301? NO, dir invalida

-iii) Segmento: 2, Desplazamiento: 5678

15833 > 5678? Si

68010 + 5678 = 73688

-iv) Segmento: 1, Desplazamiento: 18976

24300 > 18976? Si

28699 + 18976 = 47675

-v) Segmento 3, Desplazamiento: 0

400 > 0? SI

80001 + 0 = 80001

31. Paginación

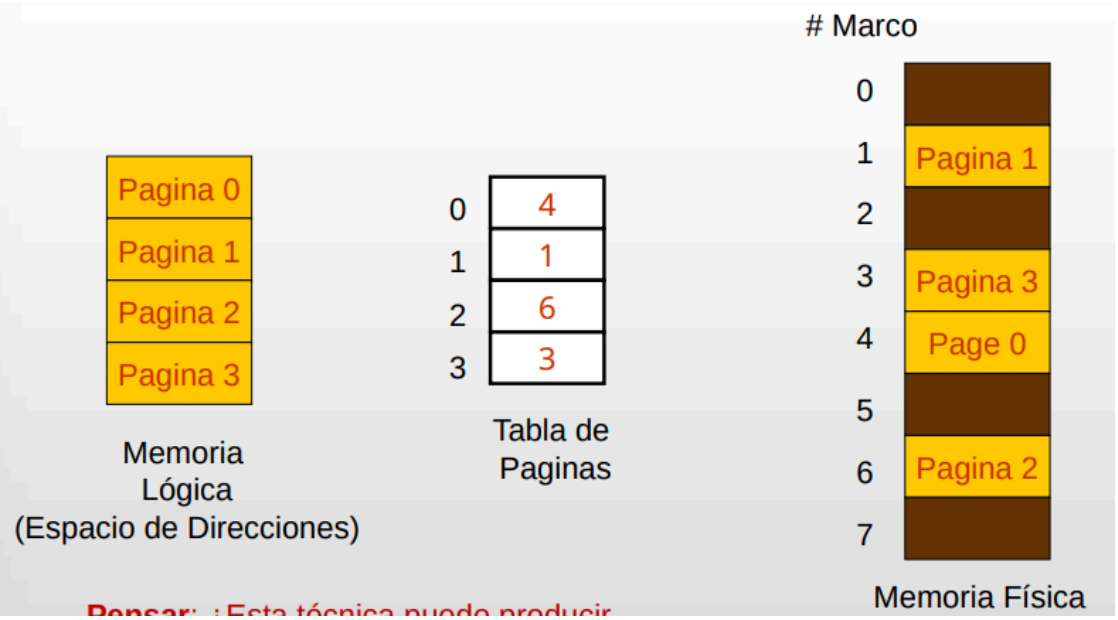
a. Explique cómo trabaja este método de asignación de memoria.

-Memoria fisica se divida en partes iguales denominadas frames/marcos. La memoria logica se divida en partes de igual tamaño que los marcos que se denominan paginas. Se interpreta como un numero de pagina y un desplazamiento dentro de la misma contenidos en una tabla de paginas

b. ¿Qué estructuras son necesarias en el Kernel para poder ejecutarse sobre un Hardware que utiliza paginación?

-El Kernel debe mantener una tabla de paginas para cada proceso, cada entrada contiene el marco en el que se coloca ese numero de pagina especifico. Se guarda la base del marco que fue cargado en la PCB

c. Explique, utilizando gráficos, cómo son resueltas las direcciones lógicas en físicas.



d. En este esquema: ¿se puede producir fragmentación (interna y/o externa)?

-En este esquema se puede producir fragmentacion interna en el caso que un proceso no sea exactamente del mismo tamaño que una pagina y requiera de 2 o mas paginas. Por ejemplo si cada pagina es de tamaño 20kb y llega un proceso de 30kb este mismo se dividiria en 2 paginas y en la 2da pagina te sobrarian 10kb que no pueden ser asignados a ninguna pagina.

32.

Suponga un sistema donde la memoria es administrada mediante la técnica de paginación, y donde:

El tamaño de la página es de 512 bytes

Cada dirección de memoria referencia 1 byte.

Se tiene una memoria principal de 10240 bytes (10 KiB) y los marcos se enumeran desde 0 y comienzan en la dirección física 0.

Suponga además un proceso P1 con un tamaño lógico de 2000 bytes y con la siguiente tabla de páginas:

Página	Marco
0	3
1	5
2	2
3	6

a. Realice los gráficos necesarios (de la memoria principal, del espacio de direcciones del proceso, de tabla de páginas y de la tabla de marcos) en el que refleje el estado descripto.

Página	Marco
0	3
1	5
2	2
3	6

Marco	Inicio-Fin
0	0-511
1	512-1023
2	1024-1535
3	1536-2047
4	2048-2559
5	2560-3071
6	3072-3583

b. Indicar si las siguientes direcciones lógicas corresponden al espacio lógico del proceso P1 y en caso afirmativo indicar la dirección física a la que corresponden:

i. 35 ii. 512 iii. 2051 iv. 0 v. 1325 vi. 602

Num pagina: Dir Logica DIV tamaño pagina

Desplazamiento: Dir Logica MOD tamaño pagina

Direccion fisica: Base marco + desplazamiento

i) 35, es valida

35 DIV 512 = Pagina 0

Desplazamiento: 35 MOD 512 = 35

Direccion fisica: 1536 + 35 = 1573

ii) 512, es valida

512 DIV 512 = Pagina 1

Desplazamiento: 512 MOD 512 = 0

Direccion fisica: 2560 + 0 = 2560

iii) 2051, no es valida

2051 DIV 512 = Pagina 4 → Marco ??

iv) 0, es valida

0 DIV 512 = Pagina 0

Desplazamiento: 0 MOD 512 = 0

Direccion fisica: 1536 + 0 = 1536

v) 1325, es valida

1325 DIV 512 = Pagina 2

Desplazamiento: 1325 MOD 512 = 301

Direccion fisica: 1025 + 301 = 1325

vi) 602, es valida

603 DIV 512 = Pagina 1

Desplazamiento: 601 MOD 512 = 90

Direccion fisica: 2560 + 90 = 2650

c. Indicar, en caso de ser posible, las direcciones lógicas del proceso P1 que se corresponden a las siguientes direcciones físicas:

i. 509 ii. 1500 iii. 0 vi. 2000 iv. 3215 v. 1024

Num marco: Dir Fisica DIV tamaño pagina

Desplazamiento: Dir Fisica MOD tamaño pagina

Direccion logica: Tamaño pagina + desplazamiento

i) 509, No es valida

509 DIV 512 = Marco 0

Desplazamiento: 509 MOD 512 = 509

ii) 1500, es valida

1500 DIV 512 = Marco 2

Desplazamiento: 1500 MOD 512 = 476

Direccion logica: 512 + 476 = 988

iii) 0, no es valida

0 DIV 512 = Marco 0

Desplazamiento: 0 MOD 512 = 0

iv) 3215, es valida

3215 DIV 512 = Marco 6

Desplazamiento: 3215 MOD 512 = 143

Direccion logica: 512 + 143 = 655

v) 1024, es valida

1024 DIV 512 = Marco 2

Desplazamiento: 1024 MOD 512 = 0

Direccion logica: 512 + 0 = 512

33.

Dado un esquema donde cada dirección hace referencia a 1 byte, con páginas de 2 KiB (KibiBytes), donde el frame 0 se encuentra en la dirección física 0. Con las siguientes siguientes primeras entradas de la tabla de páginas de un proceso, **traduzca las direcciones lógicas indicadas a direcciones físicas:**

i. 5120 ii. 3242 iii. 1578 iv. 2048 v. 8191

Página	Marco
0	16
1	13
2	9
3	2
4	0

Marco	Inicio-Fin
0	0-2047
2	4096-6143
9	18432-20479
13	26624-28671
16	32768-34815

2 kib = 2048

Num pagina: Dir Logica DIV tamaño pagina

Desplazamiento: Dir Logica MOD tamaño pagina

Direccion fisica: Base marco + desplazamiento

i) 5120

5120 DIV 2048 = Pagina 2

Desplazamiento: 5120 MOD 2048 =1024

Direccion fisica: 18432 + 1025 = 19456

ii) 3242

3242 DIV 2048 = Pagina 1

Desplazamiento: 3242 MOD 2048 = 1194

Direccion fisica: 26624 + 1194 = 27818

iii) 1578

1578 DIV 2048 = Pagina 0

Desplazamiento: 1578 MOD 2028 = 1578

Direccion fisica: 32768 + 1578 = 34346

iv) 2048

2048 DIV 2048 = Pagina 1

Desplazamiento: 2048 MOD 2028 = 0

Direccion fisica: 26624 + 0 = 26624

iv) 8191

8191 DIV 2048 = Pagina 3

Desplazamiento: 8191 MOD 2028 = 2047

Direccion fisica: 4096 + 2047 = 6143

34.

Tamaño de la Página. Compare las siguientes situaciones con respecto al tamaño de página y su impacto, indicando ventajas y desventajas:

a. Un tamaño de página pequeño.

b. Un tamaño de página grande.

-Pequeño → Menor fragmentacion interna, mas paginas por proceso, tablas de paginas mas grandes, mas paginas pueden estar en memoria

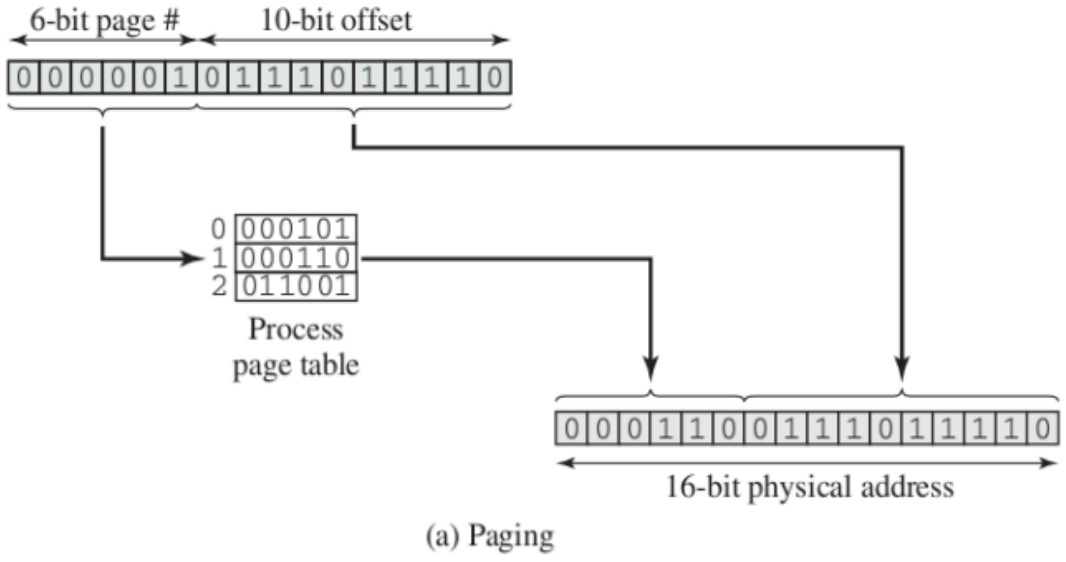
-Grande → Mayor fragmentacion interna, memoria secundaria transfiere bloques de datos mas grandes de manera mas eficiente, mas rapido mover paginas a memoria principal

35.

Existen varios enfoques para administrar las tablas de páginas, Explique brevemente cómo trabajan estos mecanismos. Tenga en cuenta analizar cómo se resuelven las direcciones y qué ventajas/desventajas presentan una sobre otra.:

-Tablas de páginas de 1 nivel.

-Solo hay una tabla de paginas que mapea las direcciones logicas a fisicas. La transformacion de direcciones se realiza medinte la tabla, agarrando la direccion logica y descomponiendola en un indice que se utiliza para acceder a la entrada correspondiente en un tabla y a eso le aplicas un desplazamiento



(a) Paging

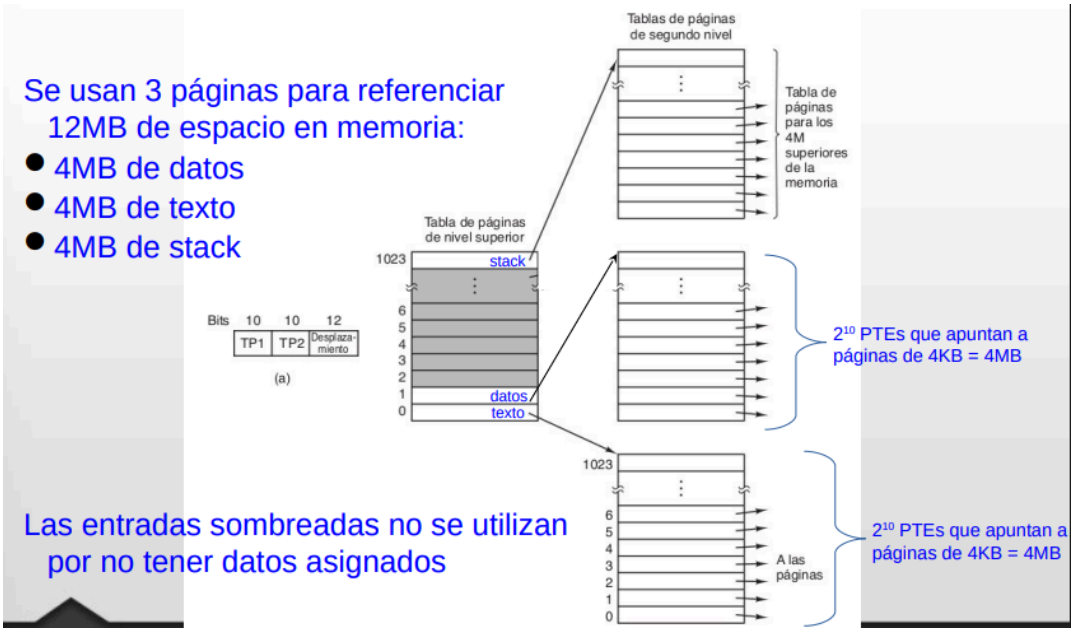
-De una direccion de 32 bits, usa 20 de para num de pagina, 10 desplazamiento. Tiene 2^20 cantidad maxima de paginas y el tamaño de cada una es de 2^12

-Cada numero de cada PTE (Page Table Entries) es de 4 bytes, osea que la cantidad de PTEs que entran en un marco es de 2^10 (4kb/4b, es el tamaño de marco dividido cuanto mide cada marco)

-Todo proceso necesita el tamaño total para su tabla de paginas, terminas teniendo mas espacio para la tabla de paginas que para el tamaño del codigo del proceso en si

-Tablas de páginas de 2 niveles o más

-Para solucionar este problema se hizo un mapeo en memoria de tablas de paginas de 2 niveles (tabla de paginas de nivel superior mapean a tablas de paginas de 2do nivel). Cada tabla de paginas tiene el mismo tamaño pero se busca un menor numero de paginas por tabla para que ocupen la menor cantidad de RAM posible. Se carga una parte parcial en la tabla de paginas (datos, texto, stack, como en la imagen) y es lo unico que se necesita resolver formando un direccionamiento indirecto, es decir, las tablas de segundo nivel se pueden llevar a memoria secundaria liberando RAM pero los accesos aumentan (explicado abajo)



-De tablas de paginas de 4 MB de tamaño pasarias a tablas de 16kib, este metodo soluciona el problema de grandes espacios desperdiciados para tablas de pagina. El problema de esto es que para andar resolviendo los calculos tablas de memoria de segundo nivel estas accediendo a memoria todo el tiempo (MMU + meterse a la de primer nivel y buscar las de 2do nivel, y asi con cuantos N niveles haya)

-Tablas de páginas invertidas.

-Solo unca tabla en todo el sistema que contiene las entradas de paginas para todos los procesos. El espacio de direcciones hace referencia al

espacio de la RAM

-Se le aplica una funcion de hash a la direccion virtual y obtiene una nueva direccion y con ese valor accede a una tabla invertida para encontrar el marco asociado. La tabla invertida mantiene informacion de qué hay cargado en la RAM en ese moment, es cecir, se sabe si la direccion esta cargada en la memoria fisica o no, es al reves a lo que venimos viendo

36.

Dado un esquema de paginación, donde cada dirección hace referencia a 1 byte, se tiene un tamaño de dirección de 32 bits y un tamaño de página de 2048 bytes. Suponga además un proceso P1 que necesita 51358 bytes para su código y 68131 bytes para sus datos.

a. De los 32 bits de las direcciones ¿Cuántos se utilizan para identificar página? ¿Cuantos para desplazamiento?

-21 bits para identificar pagina, 11 bits para el desplazamiento (2^11 = 2048)

b. ¿Cuál sería el tamaño lógico máximo de un proceso?

-2^32 bytes, 4.294.967.296 = 4GiB

c. ¿Cuántas páginas máximo puede tener el proceso P1?

-2^21

d. Si se contaran con 4GiB (Gibibytes) de RAM, ¿cuantos marcos habría disponibles para utilizar?

-2^21

e. ¿Cuántas páginas necesita P1 para almacenar su código?

-51358/2048 = 25,06 → Necesita 26 paginas

f. ¿Cuántas páginas necesita P1 para almacenar sus datos?

-68131/2048 = 33.27 → Necesita 34 paginas

g. ¿Cuántos bytes habría de fragmentación interna entre lo necesario para almacenar su código y sus datos? Ayuda: Recuerde que en una misma página no pueden convivir código y datos.

-Para el codigo: 2048 - (51378 MOD 2048) = 2048 - 148 = 1890 bytes desperdiciados

-Para los datos: 2048 - (68131 MOD 2048) = 2048 - 163 = 1885 bytes desperdiciados

h. Asumiendo que el hardware utiliza tabla de páginas de un (1) nivel, que la dirección se interpreta como 20 bits para identificar página y 12 bits para desplazamiento, y que cada Entrada de Tabla de Páginas (PTE) requiere de 1 Kib. ¿Cuál será el tamaño mínimo que la tabla de páginas del proceso P1 ocupará en RAM?

-2^12 = 4096

-P1 usa 30 paginas → 119489/4096 = 29,18 → 30 paginas

-30 × 1024 (PTE) = 30720 bytes es el tamaño minimo de la tabla de paginas para poder cubrir al proceso

i. Asumiendo que el hardware utiliza tabla de páginas de dos (2) niveles, que la dirección se interpreta como 10 bits para identificar primer nivel, 10 bits para identificar el segundo nivel, 12 bits para desplazamiento y que cada Entrada de Tabla de Páginas (PTE) requiere de 1 Kib. ¿Cuál será el tamaño mínimo que la tabla de páginas del proceso P1 ocupará en RAM?

-2^12 = 4096

-P1 usa 30 paginas → 119489/4096 = 29,18 → 30 paginas

-No entendi bien pero creo que usa 1 entrada en nivel 1 y 30 en nivel 2 → 31 x 1 kib(4096) = 31744 bytes

j. Dado los resultados obtenidos en los dos incisos anteriores (h, i) ¿Qué conclusiones se pueden obtener del uso de los 2 modelos de tablas de página?

37. Cite similitudes y diferencias entre las 4 técnicas vistas: Particiones Fijas, Particiones Dinámicas, Segmentación y Paginación.

38.

Dada la técnica de administración de memoria por medio de segmentación paginada:

a. Explique cómo funciona.

b. Cite ventajas y desventajas respecto a las técnicas antes vistas.

c. Teniéndose disponibles las siguientes tablas:

TabladeSegmentos	TabladePáginas
Núm. Seg.Dir. base15002150035000	Nro. SegmentoNro. PáginaDirec. Base114028036021202253031120215O

Indicar las direcciones físicas correspondientes a las siguientes direcciones lógicas (segmento,página,desplazamiento):

i. (2,1,1) ii. (1,3,15) iii. (3,1,10) iv. (2,3,5)

Página	Marco
0	3
1	5
2	2
3	6