8.- Para la siguiente cadena de referencia: 8,1,2,3,1,4,1,5,3,4,1,4. Suponiendo que se disponen de 3 marcos de página, el algoritmo de sustitución óptimo presenta:
a) 10 fallos de página b) 0 fallos de página **c) 7 fallos de página** d) 5 fallos de página

1.- **(3 puntos)** En un parque infantil hay un tobogán y un tren móvil. Los niños comparten las atracciones pero éstas tienen una capacidad limitada. Todos los niños quieren inicialmente montar en el tobogán y, después, montar en el tren. Pero se encuentran con el inconveniente de que sólo uno de ellos puede montar en el tobogán al mismo tiempo y sólo tres pueden montar en el tren. Define un proceso que ejecuten los niños concurrentemente de forma que se sincronicen estas actividades usando semáforos.

```
module Niños_felices
var
   semaphore: tren {general}
              tobogan{binario}

Process NiñosX;
   begin
      loop
         begin
            wait(tobogan);
            montar_tobogan();
            signal(tobogan);
            wait(tren);
            montar_en_tren();
            signal(tren);
         end;
  end;
```

```
Process Padre;
   begin
inicializa (tren=3);
inicializa (tobogan=1);

   cobegin
      Niños;
   coend;
   end;
```

2.- **(3 puntos)** En un sistema con gestión de memoria virtual por demanda de páginas, el tamaño de la página es de 1 Kb y el sistema posee 64 Kb de memoria física disponible para programas de usuario. En un determinado momento un programa de usuario que ocupa 9 páginas se carga para su ejecución. Considerando que en ese momento es el único proceso en ejecución, y que inicialmente se cargan las páginas 0, 4, 5 y 8 en los marcos 9, 3, 8 y 5 respectivamente.
a) Dibujar la tabla de páginas para esta situación.
b) Calcular la dirección física para las direcciones virtuales (2,50) y (5,20). Explicar el proceso de traducción de direcciones.
c) Con una política de reemplazo de páginas global, y partiendo de la situación inicial indicada, calcular los fallos de página que se producen con el algoritmo LRU para la siguiente cadena de referencia:
7 5 6 1 0 8 3 4 3 3 1 2 8 6 2 3 5 3 4
d) Calcular los fallos de página para la misma cadena de referencia, pero considerando que sólo se dispone de 6 marcos de página para este proceso (considerar que el orden de carga de páginas inicial fue 0, 4, 5 y 8)

a)

Nº Página	Nº Marco físico	Bit de presente/ausente
0	9	1
1	-	0
2	-	0
3	-	0
4	3	1
5	8	1
6	-	0
7	-	0
8	5	1

b) Dirección virtual (2,50): Al acceder a la tabla indica su bit de presente/ausente indica uqe no está, luego se da un fallo de página y se debe de iniciar el proceso de carga de página.
Dirección virtual (5,20), su bit de presente/ausente indica que está y según la tabla se encuentra en el marco físico 8, la dirección física inicial del marco será 8*1024 = 8192, y el desplazamiento es 20, luego la dirección será 8192+20=8212.

c) Inicialmente se tienen ya ocupados 4 marcos, pero el sistema tiene aún libres 60 marcos, (64 Kb de memoria con páginas de 1 Kb dan 64 marcos, de los que inicialmente sólo están 4 ocupados). Luego sólo se producirán 5 fallos de página que corresponden a las referencias a una nueva página.

d) En esta situación, se ha supuesto que todos los marcos están ocupados excepto 5, que son los que puede ocupar este proceso

0	4	5	8	7	5	6	1	0	8	3	4	3	3	1	2	8	6	2	3	5	3	4
	0	4	5	8	7	5	6	1	0	8	3	4	4	3	1	2	8	6	2	3	5	3
		0	4	5	8	7	5	6	1	0	8	8	8	4	3	1	2	8	6	2	2	5
			0	4	4	8	7	5	6	1	0	0	0	8	4	3	1	1	8	6	6	2
				0	0	4	8	7	5	6	1	1	1	0	8	4	3	3	1	8	8	6
						0	4	8	7	5	6	6	6	6	0	0	4	4	4	1	1	8
f	f	f	f	F		F	F	F		F	F				F		F			F		F

En las cuatro primeras páginas se habían producido 4 fallos, y a partir de entonces 10 fallos.

Para las tres preguntas siguientes, considérese los procesos Productor y Consumidor, que se describen a continuación, y que se ejecutan concurrentemente.
El semáforo buffer_vacio indica el número de registros que están vacíos. El semáforo buffer_lleno indica el número de registros que están llenos. N es el número de registros del buffer
semaphore: buffer_lleno, buffer_vacio, exclusión;
buffer_lleno=N-1; buffer_vacio=1;

```
Process Productor;
  begin
    wait(buffer_vacio);
    wait(exclusion);
     Produce;
     signal(exclusion);
     signal(buffer_lleno);
  end;
```

```
Process Consumidor;
  begin
    wait(buffer_lleno);
    wait(exclusion);
     Consume;
     signal(exclusion);
     signal(buffer_vacio);
  end;
```

1.- En la codificación realizada, el semáforo exclusión:
a) Se debe inicializar a 0. **b) Se debe inicializar a 1.** c) Se debe inicializar a N. d) No se debe inicializar.

2.- En la codificación realizada, ¿Qué proceso entra primero en la sección crítica?.
a) El productor. b) Los dos se quedan bloqueados. **c) No se puede determinar.** d) El consumidor.

3.- Si se ejecutan tres procesos productores seguidos de un consumidor, la situación de las variables es:
a) exclusión a 1, buffer_lleno=N, buffer_vacio=0. b) exclusión a 0, buffer_lleno=N, buffer_vacio=0.
c) exclusión a 1, buffer_lleno=N, buffer_vacio=0 y un productor en la cola del semáforo buffer_vacio.
d) exclusión a 0, buffer_lleno=N, buffer_vacio=0 y un productor en la cola del semáforo buffer_vacio.

4.- Se tienen 3 procesos: P_1, P_2 y P_3, con tiempos de ejecución: 65, 45 y 120 ms, respectivamente. Si actúa el planificador a corto plazo según el algoritmo SJF (*Short Job First*) se obtiene que:
a) Los procesos se encuentran en la lista de preparados en el orden: P_2, P_1 y P_3.
b) Los procesos se ejecutan en el orden: P_2, P_1 y P_3.
c) Los procesos se ejecutan en el orden: P_1, P_2 y P_3.
d) Los procesos se ejecutan según la prioridad que posean los procesos.

7.- El tiempo necesario para leer N bloques consecutivos de un archivo en un sistema con asignación mediante listas enlazadas es (t_b tiempo de búsqueda, t_r tiempo rotacional, t_t tiempo de transferencia):

a) $t_{E/S} = t_b + t_r + N * t_t$ **b)** $t_{E/S} = N * (t_b + t_r + t_t)$

c) $t_{E/S} = (N+1) * (t_b + t_r + t_t)$ d) $t_{E/S} = t_b + t_r + t_t$

8.- En un sistema de memoria virtual, el tiempo promedio de acceso (t_{pa}) es (a_m tiempo de acceso a memoria, p probabilidad de que ocurra un fallo de página, f_p tiempo que lleva resolver un fallo de página):

a) $t_{pa} = a_m + p * f_p$ **b)** $t_{pa} = (1-p) * a_m + p * f_p$

c) $t_{pa} = (1-p) * a_m + f_p$ d) $t_{pa} = p + a_m + f_p$

1.- **(3 puntos)** Los directores de un complejo turístico nos piden que regulemos con semáforos el acceso de los turistas a los templos a visitar desde una sala de exposiciones. Para acceder a los templos es necesario esperar en la sala a que venga a buscarnls un guía. El número total de guías es G. Si un visitante quiere acceder a los templos y hay más gente esperando a que venga un guía, deberá hacer cola. La visita es personalizada, es decir, cada guía se lleva sólo a un visitante. Si un guía esta disponible y no hay visitantes esperando a la visita de los templos, el guía descansará.

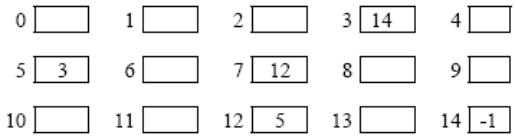
```
module Visita_Templos
var
  semaphore: guia {general}
             visitante{binario}
```

```
Process VisitanteX;
  begin
    loop
      begin
        signal(visitante);
        wait(guia);
        seguir al guía;
        ver los templos;
      end;
end;
```

```
Process guiaX;
  begin
    loop
    begin
      wait(visitante);
      signal(guia);
      explicar el templo;
    end;
  end;
```

```
Process Padre;
begin
inicializa (visitante,0)
inicializa (guia, G);
cobegin
visitantes;
guias;
coend;
end;
```

2.- (3 puntos) En la figura se representan los 15 primeros bloques de un dispositivo de un disco que en total dispone de 30 Mb de capacidad. La asignación de espacio en disco se realiza mediante listas enlazadas. Los bloques tienen un tamaño de 512 bytes.
a) Calcular para cada bloque, cuántos bytes se podrán asignar a datos y cuántos a punteros a otros bloques.
b) Calcular el tamaño máximo, en bytes, de los datos almacenados en el archivo *Notas*, que también se representa en la figura.
c) Indicar el problema que presenta el acceso directo a un bloque en los archivos de este sistema.
d) Si se utilizara un método mediante indexación, ¿se resolvería el problema indicado anteriormente para las listas enlazadas?
e) Para el método mediante indexación, calcular el tamaño máximo de los datos que se pueden ahora almacenar en el archivo notas. ¿Hay alguna variación con respecto al caso anterior?
f) ¿Existe pérdida de espacio?¿Cuánto?



Directorio
Notas
Bloque de comienzo: 7
Bloque Final: 14

a) El disco contiene un total de 30*1024*1024 bytes= 31.457.280 bytes
Como los bloques son de 512 bytes, el número total de bloques en el disco es de: 31.457.280 /512 = 61440, para poder direccional todos estos bloques se necesitan por lo menos 16 bits (2^{16} = 65.536), es decir 2 bytes. Luego habrá en cada bloque 2 bytes para punteros, y 510 para datos.

b) El archivo *Notas* ocupa los bloques 7, 12, 5, 3 y 14, cinco bloques. Como cada bloque usa 510 bytes para datos, en total se tendrá 5*510 = 2550 bytes.

c) El problema que presenta es que hay que leer todos los bloques para llegar a uno determinado. Así, para leer el bloque k deben leerse previamente los k-1 para ir accediendo a los punteros que apuntan al siguiente bloque.

d) Sí. En este método, el directorio contiene la dirección del bloque donde están los índices a los bloques de datos del archivo.

e) En este tipo de acceso, todos los bytes de un bloque se utilizan para datos, con lo cuál para el archivo notas se tendrán 5*512 = 2560 bytes. Con respecto al anterior método se utilizan dos bytes mas por cada bloque para datos.

f) La asignación mediante indexación presenta sin embargo pérdida de espacio. Si en la tabla de índices se le asigna un bloque entero, como los índices son de 2 bytes, el bloque está ocupado por 5 índices x 2 bytes = 10 bytes. Por lo que en el bloque está desaprovechado : 512 – 10 = 502 bytes para este fichero en concreto.

4.- Sea la siguiente carga de procesos. Para la política de planificación Tiempo que queda mas corto (SRT):

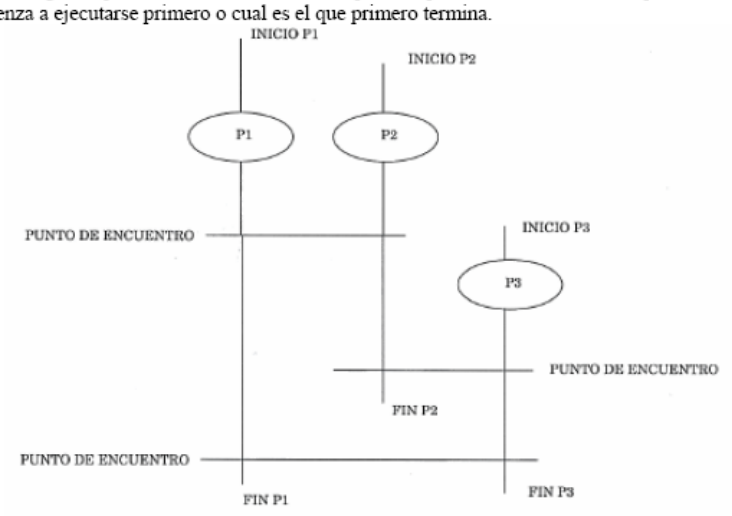
PROCESO	TIEMPO DE LLEGADA	TIEMPO DE EJECUCIÓN
A	0	3
B	1	5
C	3	2

a) El tiempo de espera de B = 5. **b) El tiempo de espera de B = 4.**

2.- ¿Cuál es el orden de ejecución de los procesos P1, P2 y P3 según el algoritmo **SJF** (primera tarea más corta) si sus tiempos de ejecución son 15 ms, 5 ms y 15 ms y el orden de llegada al sistema es a 0 ms, 5 ms y 10 ms, respectivamente?
a) P1, P2, P1 y P3. **b) P1, P2 y P3.** c) P2, P1 y P3.

6.- Supongamos un sistema de gestión de memoria virtual con paginación, en el que se utiliza como algoritmo de reemplazo el LRU. Existe un proceso al que se le asignan 4 marcos durante toda su ejecución y que hace referencia a la siguiente lista de páginas: 4 8 9 7 3 8 4 8 4 6 8
a) Una vez cargadas las cuatro primeras páginas en memoria, tras la referencia al resto de las páginas de la lista se producirán 3 fallos de página.
b) Si después de la lista anterior se necesita la página 3 se producirá un fallo de página.
c) Si después de la lista anterior se necesita la página 7 se expulsará a la página 4.
d) Ninguna de las afirmaciones anteriores es cierta.

1.- (3 puntos) Las aplicaciones de gran tamaño suelen estar compuestas de varios procesos. Hay veces que dos o más procesos se pueden ejecutar en paralelo hasta un determinado instante. Luego, para que puedan continuar cada uno de los procesos que se han estado ejecutando en paralelo, es necesario que se encuentren en un punto. Vamos a suponer que nuestro sistema es de tres procesos y los puntos de encuentro están dados en la figura adjunta. Se pide modelar con semáforos esta situación de forma que hasta que dos procesos no hayan llegado al punto donde se debe encontrar, ninguno de ellos puede continuar. El proceso que antes llegue al punto de encuentro deberá esperar a que el otro alcance dicho punto. No se sabe cuál de los dos procesos comienza a ejecutarse primero o cual es el que primero termina.




```
module Puntos_Encuentro
var
   semaphore: {binario} semP1P2, semP1P3, semP2P1, semP2P3, semP3P1, semP3P2;

   Process P1;                Process P2;                Process P3;
      begin                      begin                      begin
        instrucciones de P1;       instrucciones de P2;       instrucciones de P3;
        signal(semP2P1);           signal(semP1P2);           signal(semP2P3);
        wait(semP1P2);             wait(semP2P1);             wait(semP3P2);
        instrucciones de P1;       instrucciones de P2;       instrucciones de P3;
        signal(semP3P1);           signal(semP3P2);           signal(semP1P3);
        wait(semP1P3);             wait(semP2P3);             wait(semP3P1);
        instrucciones de P1;       instrucciones de P2;       instrucciones de P3;
        end;                       end;                       end;

process Padre;
begin
semP1P2=0, semP1P3=0, semP2P1=0, semP2P3=0, semP3P1=0, semP3P2=0;
cobegin
P1, P2, P3;
coend;
end;
```

2.- (3 puntos) Supongamos que tenemos una máquina con 16 MB de memoria principal y un esquema de gestión de memoria virtual paginado con páginas de 4 KB. Un proceso produce la siguiente secuencia de accesos a direcciones de memoria (mostradas aquí en hexadecimal):

02D4B8, 02D4B9, 02D4EB, 02D4EB, 02D86F, F0B621, F0B815, F0D963, F0B832, F0BA23, D9D6C3, D9B1A7, D9B1A1, F0BA25, 02D4C7, 628A31, F0B328, D9B325, D73425.

El sistema operativo asigna al proceso 4 marcos de memoria principal. Se pide:

a) Indicar cuál es el formato de una dirección física de memoria.
b) Dar la cadena de referencia de las páginas accedidas por el proceso.
c) Si el sistema operativo utiliza 4 marcos de memoria principal, describir el comportamiento del gestor de memoria utilizando cada uno de los siguientes algoritmos de reemplazo de páginas, indicando cuántos fallos de página se producen con los algoritmos FIFO y de la segunda oportunidad.

Solución:
a) A partir del tamaño y organización de la memoria, se calcula el formato de una dirección física:

16 MB = 2^{24} bytes, es decir, se requieren 24 bits para direccionar la memoria física. Como además se nos indica que el tamaño de página es de 4 KB, tendremos que
4 KB = 2^{12} bytes, por lo que se necesitan 12 bits para direccionar un byte de una página.

Por tanto, el formato de una dirección física de memoria será:

página	desplazamiento
12 bits	12 bits

b) Dado que la cadena de referencias viene dada en hexadecimal, y para expresar un número hexadecimal se necesitan 4 bits en binario, para direccionar la página se necesitan 3 dígitos hexadecimales y otros 3 para el desplazamiento. Por tanto la dirección física queda expresada según el siguiente formato:

página			desplazamiento		
12 bits			12 bits		
Hex	Hex	Hex	Hex	Hex	Hex
0	2	D	4	B	8

Así, la secuencia de páginas que formará la cadena de referencias es:
02D, F0B, F0D, F0B, D9D, D9B, F0B, 02D, 628, F0B, D9B, D73

c) Algoritmo FIFO

02D	F0B	F0D	F0B	D9D	D9B	F0B	02D	628	F0B	D9B	D73
02D	02D	02D		02D	F0B		F0D	D9D	D9B		02D
	F0B	F0B		F0B	F0D		D9D	D9B	02D		628
		F0D		F0D	D9D		D9B	02D	628		F0B
				D9D	D9B		02D	628	F0B		D73
F	F	F		F	F		F	F	F		F

En total se producen 9 fallos de página.

Algoritmo Segunda Oportunidad

02D	F0B	F0D	F0B	D9D	D9B	F0B	02D	628	F0B	D9B	D73
$02D_1$	**$02D_1$**	**$02D_1$**		**$02D_1$**	$D9B_1$	$D9B_1$	$D9B_1$	**$D9B_1$**	**$D9B_1$**		$D73_1$
	$F0B_1$	$F0B_1$		$F0B_1$	**$F0B_0$**	**$F0B_1$**	$F0B_0$	$F0B_0$	$F0B_1$		**$F0B_0$**
		$F0D_1$		$F0D_1$	$F0D_0$	$F0D_0$	$02D_1$	$02D_1$	$02D_1$		$02D_0$
				$D9D_1$	$D9D_0$	$D9D_0$	**$D9D_0$**	628_1	628_1		628_0
F	F	F		F	F		F	F			F

En total se producen 8 fallos de página.

1.- (3 puntos) Considere la siguiente carga de procesos:

PROCESO	TIEMPO DE LLEGADA	TIEMPO DE EJECUCIÓN
A	0	3
B	1	5
C	3	2
D	9	5
E	12	2

Para las siguiente políticas de planificación:

a) Round Robin o prioridad circular con cuanto igual a 4 y
b) Tiempo que queda mas corto (SRT)

Obtenga el diagrama de Gantt y el tiempo de espera medio por cada política.

Solución:

RR:

A	A	A	B	B	B	B	C	C	B	D	D	D	D	E	E	D	-	-	-
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Tiempo de retorno:
A = 3s
B = 10-1 = 9s
C = 9-3 = 6s
D = 17-9 = 8s
E = 16-12 = 4s

Tiempo de espera:
A = 3-3 = 0s
B = 9-5 = 4s
C = 6-2 = 4s
D = 8-5 = 3s
E = 4-2 = 2s

SRT:

A	A	A	C	C	B	B	B	B	B	D	D	E	E	D	D	D	-	-	-
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Tiempo de retorno:
A = 3s
B = 10-1 = 9s
C = 5-3 = 2s
D = 17-9 = 8s
E = 14-12 = 2s

Tiempo de espera:
A = 3-3 = 0s
B = 9-5 = 4s
C = 2-2 = 0s
D = 8-5 = 3s
E = 2-2 = 0s

6.- Suponiendo que se tienen 4 marcos de página inicialmente vacíos, y que se tiene la siguiente cadena de referencia 7,5,6,1,0,8,3,4,3,3,1,2,8,6,2,3,5,3,4. ¿Cuál es el número de fallos de página si se utiliza el algoritmo de sustitución de la segunda oportunidad?

a) 10 **b) 15** c) 20 d) Ninguna de estas soluciones son válidas

2 .- (3 puntos)

Tenemos un sistema de gestión de memoria paginada. Cada entrada a la tabla de páginas ocupa 4 bytes. Cada dirección de memoria tiene 32 bits, 20 para indicar el número de página y 12 para el desplazamiento. El tamaño de las páginas es de 4 Kbytes.

a) ¿Cuántas páginas puede llegar a tener un proceso?
b) ¿Qué tamaño en bytes puede ocupar la tabla como máximo?
c) Un fichero de tamaño 100 Mbytes. ¿Cuánto espacio consumiría en la tabla de páginas?
d) En un instante determinado de tiempo la tabla de páginas es la indicada en la figura. Indicar a que direcciones físicas de memoria principal corresponden las direcciones lógicas: (0,3000), (1,5050), (2,1058), (3, 515), (4,2015). Suponer que el marco 0 empieza en la dirección 0.

N° de página	Marco de página	Bit presente/ausente
0	5	1
1	1	1
2	1	0
3	2	0
4	9	1

Solución:

a)
Si hay 20 bits para indicar el número de página, entonces se podrá tener 2^{20} = 1.048.576 páginas

b)
El tamaño será 2^{20} * 4 bytes = 4.194.304

c)
100 Mb = 100 Mb/ 4 Kb = 25.600 páginas

Luego necesitará 25.600 entradas * 4 = 102.400 bytes. Luego consumirá en tablas 102.400 bytes.

d)

La dirección física viene dada por: nº marco * tamaño + desplazamiento
Si supone que el marco 0 empieza en la dirección 0 tendremos:

(0,3000) como está en el marco 5, la dirección física será 5*4096+3000
(1,5050) se produce un error ya que el desplazamiento es mayor que una página
(2,1058) como está en el marco 1 será 1*4096+1058, pero como el bit de ausente es 0 no tiene en realidad ninguna dirección física asociada
(3,515) como está en el marco 2 será 2*4096+515, pero como el bit de ausente es 0 no tiene en realidad ninguna dirección física asociada
(4,2015) como está en el marco 9 será 9*4096+2015

1.- (2 puntos) Una tribu de caníbales cenan en comunidad una gran olla que contiene M exploradores cocinados. Cuando un caníbal quiere comer, él mismo se sirve de la olla un explorador, a menos que esté vacía. Si la olla está vacía, el caníbal despierta al cocinero y espera a que éste llene la olla. Desarrollar el código de las acciones de los caníbales y el cocinero usando semáforos.

Solución:

```
program cena;
var
olla, i:integer;
mutex, comer,cocina: semáforo;

process canibalX;
begin
  while true
  begin
    wait(mutex)
    if olla=0 then
    begin
      signal(cocina);
      wait(comer);
    end;
    olla:=olla-1;
    signal(mutex);
  end;
end,

process cocinero;
begin
  while true
  begin
    wait(cocina);
    olla:=m;
    signal(comer);
  end;
end;


begin
  olla:=m,
  inicializa(mutex,1);
  inicializa(cocina,0);
  inicializa(comer,0);
  cobegin
    canibales;
    cocinero;
  coend;
end;
```

1.- Para la siguiente tabla, determinar el tiempo de retorno y de espera para P2 al aplicar el algoritmo de planificación a corto plazo Round Robin (Prioridad Circular) con cuanto de 3s.

Proceso	Tiempo de llegada (s)	Tiempo de ejecución (s)
P1	0	9
P2	1	5
P3	2	2

a) 13 y 8 s b) 14 y 9 s *c) 12 y 7 s* d) Ninguno de los anteriores

2.- Indique si las siguientes afirmaciones son verdaderas:

I. Con el algoritmo de planificación SRT la llegada de un nuevo proceso al sistema provoca siempre el desalojo del proceso actualmente en ejecución.

II. El algoritmo de planificación SJF concede la CPU al proceso que ocupa menos espacio en memoria.

a) I: sí, II: sí. b) I: sí, II: no. c) I: no, II: sí. *d) I: no, II: no.*

3.- Considere un semáforo cuyo valor actual es 3. Una operación de `señal` sobre el mismo...

2.- (3 puntos) Un planificador de disco que tiene 200 registros (del 0 al 199). Se acaba de atender una petición en el registro 137 y ahora se está accediendo al registro 118. La cola de peticiones pendientes es: 13, 149, 88, 191, 93, 150, 101, 183, 134. Indicar en que consiste los algoritmos de planificación de disco FCFS, SSTF, SCAN y LOOK. Calcular el número de pistas que atraviesa la cabeza para satisfacer cada una de las peticiones, utilizando los algoritmos indicados anteriormente. Comentar cuál es la diferencia entre SCAN y LOOK.
Solución:

Sección 6-9 y problema 6-14

FCFS:

Pista a la que se accede	**13**	**149**	**88**	**191**	**93**	**150**	**101**	**183**	**134**
N° pistas que se atraviesan	**105**	**136**	**61**	**103**	**98**	**57**	**49**	**82**	**49**

SSTF:

Pista a la que se accede	**134**	**149**	**150**	**183**	**191**	**101**	**93**	**88**	**13**
N° pistas que se atraviesan	**16**	**15**	**1**	**33**	**8**	**90**	**8**	**5**	**75**

SCAN:

Pista a la que se accede	**101**	**93**	**88**	**13**	**134**	**149**	**150**	**183**	**191**
N° pistas que se atraviesan	**17**	**8**	**5**	**75**	**147**	**15**	**1**	**33**	**8**

LOOK:

Pista a la que se accede	**101**	**93**	**88**	**13**	**134**	**149**	**150**	**183**	**191**
N° pistas que se atraviesan	**17**	**8**	**5**	**75**	**121**	**15**	**1**	**33**	**8**

La diferencia entre SCAN y LOOK está en que SCAN tiene que llegar hasta el final, por eso cuando pasa de la 13 a la 134 pasa por 147 pistas, y LOOK no tiene que llegar al final, al pasar de la 13 a la 134 pasa solo por 121 pistas.

8.- Un computador funciona a una velocidad de 10^7 ciclos/segundo, una instrucción emplea 4 ciclos máquina, y una operación de lectura/escritura de memoria tarda 1 ciclo máquina. Si se emplean 3 instrucciones en transferir cada palabra, la máxima velocidad de transferencia de datos utilizando un sistema de DMA:

a) 525.000 palabras/segundo
b) 10.000.000 palabras/segundo
c) 833.333 palabras/segundo
d) Ninguna de las anteriores

1.- (3 puntos) Un sistema operativo dispone de un algoritmo de planificación del procesador por colas multinivel realimentadas con las siguientes características:

- Tiene tres colas: la primera sigue un algoritmo de planificación Round-Robin o prioridad circular con cuanto de 2 ms; La segunda sigue un algoritmo de planificación Round-Robin con quantum de 4 ms y la tercera sigue una planificación FIFO.
- Los procesos entran en el sistema por la primera cola.
- Los procesos son degradados de cola si el sistema los expulsa del sistema por vencimiento del cuanto.
- La planificación entre colas es por prioridad siendo la más prioritaria la primera, luego la segunda y después la tercera.

Se tiene cuatro procesos con el siguiente comportamiento:

- P1: llega en el instante 0, ejecuta 1 ms de CPU, 6 ms de E/S, 1 ms de CPU y termina.
- P2: llega en el instante 1, ejecuta 3 ms de CPU y termina.
- P3: llega en el instante 2, ejecuta 7 ms de CPU, 3 ms de E/S, 1 ms de CPU y termina.
- P4: llega en el instante 3, ejecuta 7 ms de CPU y termina.

Se pide:

a) Rellenar el diagrama de estados de ejecución de los procesos, señalando *e* para indicar ejecución, *b* para bloqueado por E/S, *p* preparado para ejecutar y *t* terminado.
b) Indicar después de que instantes los procesos se degradan de cola. Especificar la situación después del instante 7.
c) Indicar el tiempo de retorno de los cuatro procesos.

Solución:

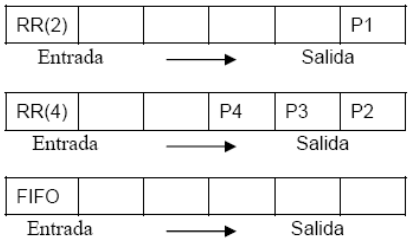
a)

ms: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
P1	e	b	b	b	b	b	b	e	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t
P2		e	e	p	p	p	p	p	e	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t
P3			p	e	e	p	p	p	p	e	e	e	e	p	p	p	p	e	b	b	b	e	t	t	t
P4				p	p	e	e	p	p	p	p	p	p	e	e	e	e	p	e	t	t	t	t	t	t

b)

P1 pasa de la cola 1 a la cola 2 después del instante **-X----** y a la cola 3 después del instante **-X----**
P2 pasa de la cola 1 a la cola 2 después del instante **---3--** y a la cola 3 después del instante **---X--**
P3 pasa de la cola 1 a la cola 2 después del instante **---5--** y a la cola 3 después del instante **-13----**
P4 pasa de la cola 1 a la cola 2 después del instante **---7--** y a la cola 3 después del instante **--17---**

Después del instante 7:



c) Los tiempos de retorno son:

P1 =8-0=8 ms
P2=9-1=8 ms
P3=22-2=20 ms
P4=19-3=16 ms

5.- El algoritmo del banquero...

a) requiere la declaración previa de cuántos procesos se van a ejecutar.
b) es un algoritmo de detección de interbloqueo
c) puede retener la ejecución de un proceso, aun cuando existan recursos disponibles para él.
d) sólo es implementable si el número de recursos es ilimitado

2.- (2 puntos). Considere un sistema de gestión de memoria virtual mediante paginación bajo demanda en el que se han medido estos tiempos:

· Tiempo medio de acceso a memoria principal: 50 nanosegundos.
· Tiempo medio de resolución de un fallo de página: 20 milisegundos.
· El resto de los tiempos se consideran despreciables

Calcule cuál es la máxima tasa de fallos de página aceptable si queremos mantener el tiempo medio de acceso a memoria (contando con los fallos de página) por debajo de los 200 nanosegundos.

Solución:

El tiempo medio de acceso viene dado por:

$$t_{pa}=(1-p)\times a_m+p\times f_p$$

donde a_m es el tiempo de acceso a memoria, que será: 50 nanoseg.
f_p es el tiempo que lleva resolver un fallo: 20 miliseg.
p es la probabilidad que ocurra un fallo.

Luego nos quedará:

$$(1\text{-}p)\times 50 \text{ nanoseg} + p\times 20 \text{ miliseg} < 200 \text{ nanoseg}$$

Hay que resolver esta desigualdad

$$20\times10^{-3}\,p+50\times10^{-9}(1-p)<200\times10^{-9}$$
$$20\times10^{6}\,p+50(1-p)<200$$
$$(20\times10^{6}-50)\,p<150$$
$$p<\frac{150}{20\times10^{6}-50}\approx 7.5\times10^{-6}$$

3.- (2 puntos) Suponer un sistema de archivos, parecido al de Unix, cuyos bloques son de tamaño de 1 Kb y los punteros a bloques son de 4 bytes. Se tienen 10 punteros a bloques directos de datos, un puntero a un bloque indirecto simple y uno a bloque indirecto doble. Se quiere incrementar el tamaño máximo del fichero. Cuál de las siguientes acciones permitiría un mayor aumento: Añadir un bloque de triple indirección o incrementar el tamaño del bloque a 4 Kb.

Solución:

El tamaño máximo de un fichero viene determinado por el número de bloques de datos posibles que permita el sistema. Entonces, tal y como es el sistema de archivos actual, el número de bloques máximo que puede tener un archivo es:
Directo----------10
Indirecto simple----------2^8
Indirecto doble----------2^{16}
Total en bytes $(2^{10}\times(10+2^8+2^{16}))$

Se ha tenido en cuenta para el computo de los bloques que en el uso del direccionamiento indirecto el número de punteros que caben en un bloque es $2^{10}/4=2^8$ punteros.

a) Si se añade un bloque de triple indirección, el máximo tamaño es:

Directo----------10
Indirecto simple----------2^8
Indirecto doble----------2^{16}
Indirecto triple----------2^{24}
Total en bytes $(2^{10}\times(10+2^8+2^{16}+2^{24}))=2^{34}+2^{26}+2^{18}+...$

b) Si se incrementa el tamaño del bloque a 4 Kb, se debe tener en cuenta que varía en número de punteros en los bloques de indirección, siendo ahora, $2^{12}/4=2^{10}$ punteros y, por lo tanto, el tamaño máximo quedará:

Directo----------10
Indirecto simple----------2^{10}
Indirecto doble----------2^{20}
Total en bytes $(2^{12}\times(10+2^{10}+2^{20}))=2^{32}+2^{22}+...$
Es mayor en el primer caso.

1.- (3 puntos) Se pide escribir un programa que conste de dos procesos concurrentes A y B sincronizados mediante dos semáforos de tal manera que el resultado final de la ejecución sea que los procesos escriban en la salida estándar en secuencia lo siguiente:

- Primero, el proceso A debe escribir: "Hola soy A".
- Segundo, el proceso B debe escribir: "Hola soy B".
- Tercero, el proceso A debe escribir: "Se despide A".
- Cuarto, el proceso B debe ecribir: "Se despide B".

Solución:

```
Program/Module Presentación;
var
    A_continua, B_continua: semaforo;

Process A;
begin
    put("Hola soy A");
    signal(B_continua);
    wait(A_continua);
    put("Se despide A");
    signal(B_continua);
end;

Process B;
begin
    wait(B_continua);
    put("Hola soy B");
    signal(A_continua);
    wait(B_continua);
    put("Se despide B");
end;

begin
    inicializa(A_continua, 0);
    inicializa(B_continua, 0);
    cobegin
        A, B;
    coend;
end;
```

2.- (3puntos).
En un sistema que implementa memoria virtual mediante demanda de páginas, un proceso genera la siguiente secuencia de referencias a páginas de memoria:

2 5 1 3 5 0 4 1 0 8 7 9 5 7 1 0 2 5 9 8 2

Con 5 marcos de página inicialmente vacíos estudiar:

a) cuantos fallos de página se producen si se utiliza el algoritmo LRU para la sustitución de páginas
b) cuantos fallos de página se produce si el algoritmo utilizado es el FIFO
c) razonar si aumentando indefinidamente el número de marcos de página se mejoraría la tasa de fallos de página.

Solución:

a) Este algoritmo sustituye la página en memoria que no ha sido referenciada en memoria desde hace mas tiempo.

2	5	1	3	5	0	4	1	0	8	7	9	5	7	1	0	2	5	9	8	2
X	2	5	1	3	5	0	4	1	0	8	7	9	5	7	1	0	2	5	9	8
X	X	2	5	1	3	5	0	4	1	0	8	7	9	5	7	1	0	2	5	9
X	X	X	2	2	1	3	5	5	4	1	0	8	8	9	5	7	1	0	2	5
X	X	X	X	X	2	1	3	3	5	4	1	0	0	8	9	5	7	1	0	0
F	F	F	F		F	F			F	F	F	F		F	F	F		F	F	

Hay 15 fallos de página

b) En este algoritmo se sustituye la página que mas tiempo lleva en memoria

2	5	1	3	3	0	4	4	4	8	7	9	5	5	1	0	2	2	2	8	8
X	2	5	1	1	3	0	0	0	4	8	7	9	9	5	1	0	0	0	2	2
X	X	2	5	5	1	3	3	3	0	4	8	7	7	9	5	1	1	1	0	0
X	X	X	2	2	5	1	1	1	3	0	4	8	8	7	9	5	5	5	1	1
X	X	X	X	X	2	5	5	5	1	3	0	4	4	8	7	9	9	9	5	5
F	F	F	F		F	F			F	F	F	F		F	F	F			F	

Hay 14 fallos de página

c) Hay que tener en cuenta que tenemos 9 páginas diferentes (10, si se cuenta la 6, aunque no se ha referenciado en el enunciado), luego como mínimo tenemos que tener 9 fallos de página (10 si no contamos la 6). En cuanto tengamos un número de marcos de página que ya nos permita tener 10 fallos de página, aunque aumentemos el número de marcos, los fallos no los reduciremos.

1.- (3 puntos) En un sistema concurrente existen dos tipos de procesos, A y B, que compiten por utilizar cierto recurso. En cualquier momento puede haber un máximo de N procesos de cualquier tipo usando el recurso (N constante). Por otro lado, para que un proceso de tipo A pueda entrar a emplear el recurso, debe haber al menos el doble de procesos de tipo B que procesos de tipo A dentro del recurso. Diseñe una solución usando semáforos.
Solución:

```
Program/module Compartir_recurso;

var mutex, avisa, hueco: semaforo;
    procA, procB, espera: integer;
```

```
Process AX
  begin
     espera(hueco);
     espera(mutex);
     if 2*procA>procB then
        begin
          espera:=espera+1;
          señal(mutex);
          señal(hueco);
          espera(avisa);
          espera(hueco);
          espera(mutex);
          espera:=espera-1;
        end;
     procA:=ProcA+1;
     señal(mutex);

     {utiliza el recurso}

     espera(mutex);
      procA:=ProcA-1;
      if (espera >0) and (2*procA<=procB) then
          señal(avisa);
      señal(mutex);
      señal(hueco);
  end;
```

```
Process BX
begin
      espera(hueco);
      espera(mutex);
      procB:=procB+1;
      if (espera >0) and (2*procA<=procB) then
         señal(avisa);
      señal(mutex);

     {utiliza el recurso}

      espera(mutex);
      procB:=procB-1;
      señal(mutex);
      señal(hueco);
   end;
```

```
Process Padre
   begin
      inicializa(mutex, 1);
      inicializa(hueco, N);
      inicializa(avisa, 0);
      ProcA=ProcB=espera=0;
      cobegin
        As, Bs;
      coend
   end;
```

2.- (3 puntos) En un sistema operativo se utiliza una estructura de nodos-i similar a la de Unix. Los bloques son de 2048 bytes. Las entradas en los nodos-i dedican 64 bits al tamaño del archivo y 32 bits a los punteros de los bloques.
El nodo-i tiene 8 entradas de direccionamiento directo, una de direccionamiento indirecto simple, y otra de direccionamiento indirecto doble. La entrada de archivos abiertos tiene una entrada para cada archivo con un campo de 64 bits que indica el desplazamiento.
Calcular el tamaño máximo de un archivo que utiliza todo el disco.

Solución:

Hay que considerar todos los parámetros que pueden limitar dicho tamaño y buscar el más restrictivo.

1) Considerar la estructura del sistema de archivos, el número máximo de bloques asignado a un archivo en su nodo-i (bloques).

Como el tamaño de un bloque es de 2.048 bytes, y los punteros son de 32 bits=4 bytes, entonces, el número de punteros a bloques que caben en un bloque es de 2048/4 = 512 punteros. Por lo tanto tendremos:

Direccionamiento directo: 8 bloques
Indirecto simple: 512 bloques
Indirecto doble: 512*512 bloques = 262144

Total= 262664 bloques

2) Considerando el tamaño de un puntero. Como el tamaño de un puntero es de 32 bits, el máximo número de bloques que se puede referenciar con un puntero es de 2^{32}

3) Teniendo en cuenta el tamaño del archivo en el nodo-i, que es de 64 bits. El tamaño máximo considerando sólo esta limitación sería de 2^{64} bytes, en bloques sería $2^{64}/2048 = 2^{64}/2^{11} = 2^{53}$

4) Teniendo en cuenta el campo desplazamiento en la tabla de archivos abiertos, que es de 64 bits.
Luego sería también de 2^{64} bytes

La solución será por tanto la mas restrictiva que corresponde a la opción 1, por la estructura de archivos.

1.- (3 puntos) En el parque de atracciones se ha establecido límite de usuarios por razones de seguridad (para ello se han instalado unos torniquetes a la entrada y a la salida que controlan en todo momento el número de personas en el parque) de forma que cuando está completo se debe esperar para entrar que alguien salga. En los días de vacaciones se forman largas colas, por lo que el parque ha decidido crear una entrada VIP, más cara que la NORMAL, pero queda preferencia a la hora de acceder a las instalaciones cuando el parque está lleno. Realizar el código de entrada y salida del parque, de forma que cuando alguien abandona el parque, su lugar será ocupado por una persona con entrada VIP, si no hubiera personas con entrada VIP esperando, entonces su lugar será ocupado por una persona con entrada NORMAL. Definir los procesos Entradas y Salidas que se ejecutan cuando una persona entra o sale del parque. Utilizar semáforos para gestionar la capacidad del parque y la cola de espera.

Solución:

```
Program/module Parque_Atracciones;

var sem_vip, sem_normal, mutex: semaforo;
    usuarios, usuarios_normales, usuarios_vip: integer;
    tiket: {vip, normal};
Const capacidad N
```

```
Process EntradaX (tiket: tiket)
  begin
     espera(mutex);
     if usuarios >= capacidad then
        if tiket = vip then
           begin
             usuarios_vip= usuarios_vip +1;
             señal(mutex);
             espera(sem_vip);
           end
        else
           begin
             usuarios_normales= usuarios_normales +1;
             señal(mutex);
             espera(sem_normal);
           end;
    else
      begin
        usuarios= usuarios+1;
        señal(mutex);
      end;
 end;
```

```
Process SalidaX(tiket: tiket)
begin
        espera(mutex);
        if usuarios_vip >= 0 then
           begin
             señal(sem_vip);
             usuarios_vip= usuarios_vip - 1;
           end;
        else
           if usuarios_normales >= 0 then
              begin
                señal(sem_normal);
                usuarios_normales= usuarios_normales - 1;
              end;
           else
              usuarios = usuarios - 1;
        señal(mutex);
   end;
```

```
Process Padre
   begin
      inicializa(mutex, 1);
      inicializa(sem_vip, 0);
      inicializa(sem_normal, 0);
      usuarios=usuarios_normales=usuarios_vip=0;
      cobegin
        EntradaS, SalidaS;
      coend
   end;
```

2.- (3puntos).
Una memoria virtual dispone de 9 marcos para programas. En ese momento se está ejecutando el proceso P1, que tiene concedidos 3 marcos, P2 y P3 solicitan memoria para ajustarse (el tamaño de P2 es el doble que el de P3, 1024 y 512 Kb respectivamente). Se realiza un reparto de todos los marcos libres de forma proporcional al tamaño de los procesos y ambos entran en ejecución. **Cuando el proceso P3 demanda por primera vez la página 3 el proceso P1 termina** y sus marcos se reparten entre los procesos P2 y P3 de forma proporcional a sus tamaños.

a) Determine cuántos marcos de página corresponden a cada proceso en cada instante.
b) Determine cuántos fallos de página se producen en el proceso P3 si la secuencia de petición de páginas es la siguiente:

1 2 1 5 2 3 7 6 5 4 3 7 5 6 7 3 7 6 4 3 7

Compare los resultados de este apartado para las políticas FIFO y LRU.

Solución:

a) Cuando los procesos P2 y P3 piden memoria el número de marcos libres es 6 pues 3 marcos están asignados al proceso P1. Como el reparto es proporcional al tamaño de los procesos, se tendrá:

Proceso	Tamaño	Factor asignación marcos	Marcos asignados
P2	1024	1024/(1024+512) = 0,6666	0,6666 * 6 = 4
P3	512	512/(1024+512) = 0,33333	0,3333 * 6 = 2

Es decir, 2/3 de los marcos se le asignan a P2 y 1/3 a P3.
Cuando el proceso P1 termina se reparten sus 3 marcos entre los procesos P2 y P3.

Proceso	Factor asignación marcos	Nuevos Marcos	Total Marcos
P2	0,67	0,6666 * 3 = 2	6
P3	0,33	0,3333 * 3 = 1	3

b) La secuencia de peticiones del proceso P3 se puede dividir en dos partes. La primera parte está formada por la secuencia de peticiones anteriores a la primera petición de la página 3, ya que durante este periodo P3 tiene asignados 2 marcos de página. La segunda parte está formada por el resto de la secuencia y durante este tiempo el proceso P3 tiene asignados 3 marcos de página.

Para FIFO

	1	2	1	5	2	3	7	6	5	4	3	7	5	6	7	3	7	6	4	3	7
X	1	2	2	5	5	3	7	6	5	4	3	7	5	6	6	3	7	7	4	4	4
X		1	1	2	2	5	3	7	6	5	4	3	7	5	5	6	3	3	7	7	7
						2	5	3	7	6	5	4	3	7	7	5	6	6	3	3	3
	F	F		F		F	F	F	F	F	F	F	F	F		F	F		F		

En total 15 fallos de páginas.

Para LRU

	1	2	1	5	2	3	7	6	5	4	3	7	5	6	7	3	7	6	4	3	7
X	1	2	1	5	2	3	7	6	5	4	3	7	5	6	7	3	7	6	4	3	7
X		1	2	1	5	2	3	7	6	5	4	3	7	5	6	7	3	7	6	4	3
						5	2	3	7	6	5	4	3	7	5	6	6	3	7	6	4
	F	F		F	F	F	F	F	F	F	F	F	F	F		F			F	F	F

En total 17 fallos de páginas.