1. Describa las diferencias entre un cambio de modo y un cambio de contexto en un sistema operativo. ¿Por qué el sistema operativo necesita realizar esos cambios?

2. Se tienen 3 procesos: P1, P2 y P3, con tiempos de ejecución: 85, 45 y 118 ms, respectivamente. Si actúa el planificador a largo plazo según el algoritmo SJF (Short Job First) se obtiene que: (Justifique su respuesta)

1. Los procesos se encuentran en la lista de preparados en el orden de llegada: P1, P2 y P3.
2. Los procesos se encuentran en la lista de preparados en el orden: P2, P1 y P3.
3. Los procesos se ejecutan en el orden de llegada: P2, P1 y P3.
4. Los procesos se ejecutan según la prioridad que posean los procesos.

1. Si tiene un sistema operativo que está diseñado para una sola CPU y desea adaptarlo a un sistema de múltiples CPU, describa los tipos de cambios / adiciones al software o hardware que necesitaría realizar en relación con la administración de procesos y por qué.

2. En relación con los threads es cierto que: (Justifique su respuesta)

    1. Cuando se crea un thread, su código hay que cargarlo desde el fichero ejecutable a memoria.
    2. Cada thread tiene su propia pila.
    3. Todos los threads que existen en el sistema en un momento dado, comparten el mismo código.
    4. Descomponer un proceso en dos threads sólo puede ejecutarse más rápido si el sistema cuenta, al menos, con dos procesadores.

1.- Explique cómo se puede implementar un semáforo. Brinde un escenario dónde sea más eficiente utilizar spinlock que locks bloqueantes.

2. ¿Cuál de las siguientes políticas de planificación es más adecuada para un sistema de multiproceso de tiempo compartido? Justifique su respuesta

1. Primero el trabajo más corto.
2. Round-Robin.
3. Prioridades.
4. FIFO.

A	0	10
B	1	15
C	6	2
D	9	9
E	12	7

Realizar la planificación de estos procesos en un sistema con dos procesadores usando el algoritmo de Colas de Múltiples Niveles con Realimentación usando 3 colas con los siguientes quantums:

- Cola 0: q = 1
- Cola 1: q = 2
- Cola 2: q = 4

Considere que inicialmente ingresan en la Cola 0, si exceden el quantum se los pasa a la cola del próximo nivel y así hasta que llega a la cola del último nivel, cuando un proceso retorna de una E/S se lo ubica en la Cola 0. Considere despreciables las rutinas de ejecución del sistema operativo.

1. Mostrar en un diagrama de Gantt la ejecución de los procesos e indicar cuál es el estado de las colas.
2. Calcular el instante de finalización, el tiempo de espera y el tiempo de retorno para cada proceso.

1.- Considere un proceso compuesto por dos hilos, *p* y *q*, que están compuestos por un conjunto de sentencias atómicas (A, B, C, D, y E). El proceso crea los dos hilos y comienza la ejecución concurrente. Muestre todos los posibles entrelazados de ejecución, considerando que los hilos contienen las siguientes sentencias.

void p()	void q()
{ A; B; C; }	{ D; E; }

2.- ¿Puede un sistema detectar inanición? Justifique su respuesta, si es Sí explique cómo lo puede realizar, si es No explique cómo puede manejar este problema.

Considere el siguiente conjunto de procesos:

Proceso	TLlegada	TEjecución
A	0	14
B	2	9
C	5	5
D	6	10
E	10	12

Realizar la planificación de estos procesos en un sistema con dos procesadores aplicando el algoritmo de Colas de Múltiples Niveles con Realimentación usando 3 colas con los siguientes quantums:

- Cola 0: q = 2
- Cola 1: q = 3
- Cola 2: q = 4

Considere que inicialmente ingresan en la Cola 0, si exceden el quantum se los pasa a la cola

1.- Suponga que la operación Signal () en una variable de condición no garantiza despertar solo un hilo, y no garantiza que los despierte en orden FIFO. ¿Podría esto afectar a los programas que utilizan variables de condición? Explica tu respuesta, brinde un ejemplo de un programa que fallaría usando esta semántica más débil.

2.- La siguiente es una propuesta de solución basada en software para el problema de exclusión mutua entre dos procesos.

boolean flag[2]={true, false};

P0:	P1:
repeat	repeat
while (flag[1]);	while (flag[0]);
flag[0]= true;	flag[1]= true;

1.- Se tienen tres procesos cuyo comportamiento se establece en la tabla que hay a continuación. "Llegada" representa el momento en el que el proceso llega al sistema, los valores que hay en las columnas etiquetadas por CPU son las unidades de tiempo de la ráfaga de CPU del proceso. Con E/S representa el tiempo que requiere su entrada salida.

Proceso	T. Arribo	CPU	E/S	CPU	E/S	CPU	E/S	CPU
A	0	10	15	5	20	10	-------	------
B	5	5	20	15	10	10	10	10
C	10	10	5	10	5	25	10	5

Mostrar en un diagrama de Gantt la ejecución de estos procesos y obtener el tiempo retorno y de espera para cada uno de los procesos considerando las siguientes planificaciones:

1. FIFO
2. Trabajo más corto primero (SJF)

Considere este conjunto de procesos:

Proceso	TLlegada	TEjecución (E/S)
A	0	5, (6), 3
B	2	3, (3), 3, (4), 1
C	4	1, (7), 2
D	7	5, (4), 2
E	12	1, (2), 2, (2), 1

Realizar la planificación utilizando el algoritmo de Round Robin con quantum igual a 3 en un sistema con dos procesadores. Considere despreciables las rutinas de ejecución del sistema operativo y que tiene un único canal para realizar las entradas salidas.

1. Mostrar en un diagrama de Gantt la ejecución de los procesos e indicar cuál es el estado de cada proceso y los cambios de la cola de listos.
2. Calcular el instante de finalización, el tiempo de espera y el tiempo de retorno para cada proceso.

1.- Suponga que tiene un sistema con cuatro hilos (threads), una variable compartida y un lock compartido. El código se escribió de tal manera que cualquier acceso de un hilo a la variable compartida siempre tendría el lock en ese hilo para su acceso. ¿Puede este sistema tener una condición de carrera que involucre la variable compartida? Justifica tu respuesta

2.- Una pareja se va a divorciar. Para dividir sus propiedades, han acordado el siguiente algoritmo. Cada mañana, cada uno puede enviar una carta al abogado del otro solicitando un bien. Dado que se tarda un día en entregar las cartas, han acordado que si ambos descubren que han solicitado el mismo artículo el mismo día, al día siguiente enviarán una carta anulando la solicitud. Entre sus propiedades se encuentran su perro Woofer, la caseta del perro, su canario Tweeter y la jaula del Tweeter. Los animales aman sus casas, por lo que se ha acordado que cualquier división de propiedad que separe a un animal de su casa es inválida, requiriendo que toda la división comience desde cero. Ambos quieren desesperadamente a Woofer. Para que puedan irse de vacaciones (por separado), cada cónyuge ha programado una computadora personal para manejar la negociación. Cuando regresan de vacaciones, las computadoras aún están negociando. ¿Por qué? ¿Es posible un interbloqueo? ¿Es posible inanición? Explique su respuesta

1.- Suponga que la operación Signal () en una variable de condición no garantiza despertar solo un hilo, y no garantiza que los despierte en orden FIFO. ¿Podría esto afectar a los programas que utilizan variables de condición? Explica tu respuesta, brinde un ejemplo de un programa que fallaría usando esta semántica más débil.

2.- La siguiente es una propuesta de solución basada en software para el problema de exclusión mutua entre dos procesos.

boolean flag[2]={true, false};

P0:	P1:
repeat	repeat
while (flag[1]);	while (flag[0]);
flag[0]= true;	flag[1]= true;
.....	.....
Sección Crítica	Sección Crítica
.....	.....

1.- Considere un proceso compuesto por dos hilos, *p* y *q*, que están compuestos por un conjunto de sentencias atómicas (A, B, C, D, y E). El proceso crea los dos hilos y comienza la ejecución concurrente. Muestre todos los posibles entrelazados de ejecución, considerando que los hilos contienen las siguientes sentencias.

void p()	**void q()**
{ A;	**{ D;**
B;	**E;**
C;	**}**
}	

2.- ¿Puede un sistema detectar inanición? Justifique su respuesta, si es Sí explique cómo lo puede realizar, si es No explique cómo puede manejar este problema.

1.- Suponga que tiene un sistema en el que hay un ciclo en un grafo de espera del sistema. ¿Es esta una señal definitiva de un interbloqueo? Justifica tu respuesta.

2.- Considere un sistema con cuatro tipo de recursos, R= (6, 9, 6, 4) (este es el número total de recursos en el sistema, y no lo que está disponible actualmente) y cuatro procesos con la siguiente matriz de máximo

Proceso	R1	R2	R3	R4
p0	2	5	0	4
p1	4	3	4	0
p2	3	1	6	3
p3	2	4	2	0

El alocador de recursos está considerando asignar los recursos según la siguiente tabla:

Proceso	R1	R2	R3	R4
p0	2	2	0	4
p1	3	2	3	0
p2	1	1	3	0
p3	0	4	0	0

Ejecute el algoritmo de seguridad en este sistema para determinar si el estado es seguro. Si es seguro, muestra una orden de ejecución de los procesos. Si no es seguro, especifique los procesos que pueden participar en un interbloqueo.

1.- ¿Puede un sistema detectar inanición? Justifique su respuesta, si es Sí explique cómo lo puede realizar, si es No explique cómo puede manejar este problema.

2.-Suponga dos hilos con una variable compartida x que ejecutan las siguientes instrucciones

P1	P2
x = 0;	x = 0;
x = x + 1;	x = x + 1;
x = x + 1;	x = x + 1;
x = x - 1;	x = x – 1;
x = x - 1;	x = x – 1;
if (x==1)	if (x==1)
Printf("Hola ...")	Printf("Hola ...")

¿Este código puede mostrar el mensaje "Hola ..."? Si la respuesta es afirmativa, Ilustre su secuencia numerando las instrucciones con el orden en que se ejecutan. Si es negativa, explique su respuesta.

2.- Considere el siguiente código para testear locks y variables de condición. Comienza cuando el hilo "principal" llama a ThreadTest (). Realice una traza de ejecución de este programa hasta que imprima el mensaje "Pare aqui". Considere que la planificación es FCFS y las colas se administran utilizando FIFO.

Lock *l;

Condition *cv;

```
void A(int arg) {
  l->Acquire();
  cv->Signal();
  Thread->Yield();
  cv->Wait();
  l->Release();
}
void B(int arg) {
  l->Acquire();
  cv->Wait();
  Thread->Yield();
```

```
void ThreadTest() {
  Thread *t;
  l = new Lock("lock");
  cv = new Condition("cv");
  createThread("A");
  createThread("B");
  Thread->Yield();
  Thread->Yield();
  printf("Pare aqui\n");
}
```

```
cv->Signal();
l->Release();
}
```

Thread->yield obliga al hilo que realiza la llamada a renunciar al uso del procesador y a esperar en la cola de listos.

1. Escribir la secuencia de cambios de contexto que ocurrieron hasta este punto,
2. Enumerar las colas en las que se encuentran los hilos en este punto, y su orden relativo si hay más de un hilo en una cola
3. Indicar el estado de cada uno de los hilos.