1)- A)- ¿Qué tipos de eventos provoca la evolución de un proceso del estado en ejecución a listo? B)- ¿Es posible que un proceso evolucione desde un estado bloqueado a un estado en ejecución? Justifique.
A)- ~~Cuando un proceso está en ejecución significa que está trabajando en la CPU en ese instante y al pasar a listo significa que su ejecución se detuvo para que pase a ejecutarse otro proceso.~~ Los eventos que pueden provocar esta evolución pueden ser que haya superado un tiempo de ejecución, es decir el planificador decide que el proceso que se estaba ejecutando ya lo hizo un tiempo suficiente y es hora de que otro proceso pase a usar la CPU, o simplemente el proceso que se estaba ejecutando finalizó correctamente, entonces cambia su estado. También esta la posibilidad que haya ocurrido una interrupción de E/S o del timer por lo qué sucedería el dicho pasaje de estado.
B)- No, no es posible. Un proceso en estado bloqueado significa que el mismo no puede ejecutarse hasta que ocurra un determinado evento, sin depender de la CPU. La correcta transición entonces, una vez que se produce ese evento determinado por el cual está esperando el proceso bloqueado, entonces pasará a estado listo necesariamente, y si el procesador está libre pasará al estado de ejecución sino tendrá que esperar, pero necesariamente pasa de bloqueado a listo, no de bloqueado a ejecución.

2)- Cuando se invoca una llamada al sistema (system call), el control evoluciona en algún momento de la capa de usuario a la capa kernel. Detalle en forma de pseudocódigo la manera en que se concretan tales cambios de privilegios, considerando las estructuras de datos involucradas.
Las llamadas al sistema se producen voluntariamente desde el modo usuario hacia el modo kernel, y cuando esta llamada se completa, se retorna al modo usuario. Son por ejemplo crear o borrar archivos, leer o escribirlos, crear nuevo proceso, etc.
En las llamadas al sistema en unix, el read tiene 3 parámetros, uno para especificar el archivo, uno para decir donde se van a colocar los datos y otro para indicar cuantos bytes se van a leer. Entonces la llamada al sistema realiza los siguientes pasos:
-Insertar estos parámetros del read en la pila.
-Se llaman a todos los procedimientos.
-Coloca en un registro el número de la llamada al sistema el cual está esperando el SO.
-Se ejecuta una interrupción para pasar de modo usuario a kernel.
-El código kernel examina el número de llamadas al sistema y la pasa al manejador (handler) a través de una tabla de apuntadores.
-Se ejecuta el manejador de llamadas al sistema.
-Cuando termina de trabajar el manejador pasa al procedimiento de biblioteca que está en el modo usuario.
-Este procedimiento regresa al programa de usuario.
-El programa usuario limpia la pila.

3)- ¿Qué diferencia existe entre una llamada al sistema y una invocación a un procedimiento convencional?
La diferencia que hay entre las mismas es que en una llamada al sistema cambia desde modo usuario a modo kernel para realizar las operaciones que tenga que realizar allí para luego volver a pasar a modo usuario. Mientras que una invocación a un procedimiento convencional directamente se realiza en modo usuario sin tener que pasar al modo kernel.

4)- ¿Por qué entiende las variables locales se almacenan en la pila en lugar de un segmento de memoria convencional?
Las variables locales son almacenadas en una pila que son un espacio de anotación temporal, ya que es mucho más rápido que almacenarlo en un segmento de memoria. Se debe a que estas variables están activas solamente durante la ejecución y cuando termina, esas variables ya no existen. Por lo tanto no tiene lógica reservar un espacio de memoria para estas variables. Entonces en la pila se crean espacios para estas variables y se destruyen al acabar la función.

5)- Detalle las estructuras de datos que provee el kernel en un sistemas de inodos para gestionar los archivos y el rol de cada una. Explique a través de pseudocódigo el uso de las mismas cuando se realice una operación de escritura de un archivo. Considere que el inodo tiene 2 enlaces directos y uno indirecto simple con 2 enlaces.
Un sistema de inodos (nodos índices), hace referencia a los archivos asociándolos con una lista de los atributos y direcciones de disco de los bloques del archivo. Dado un inodo es posible encontrar todos los bloques del archivo. Estos bloques de archivo pueden ser enlaces directos, es decir éstos apuntan

directamente a los datos que tienen almacenados, luego tenemos los enlaces indirectos simples, que cada uno de ellos apunta a una serie de enlaces directos que éstos a su vez apuntan a los datos, enlaces indirectos dobles, cada uno apunta a un enlace indirecto simple y este a su vez a los datos, y así sucesivamente. El pseudocódigo para realizar una escritura de un archivo sería el siguiente:
Debo tener en cuenta el tamaño del archivo que voy a escribir y la capacidad de almacenamiento de cada inodo.
Recorrer los inodos y encontrar espacio libre para realizar el almacenamiento, por ejemplo encuentro en los enlaces directos y voy guardando los datos.
Si no es suficiente la memoria para almacenar debo ir recorriendo los inodos disponibles.
Si están llenos los directos entro en los indirectos simples y recorro donde puedo ir guardando el archivo.

6)- ¿Qué objetivos intentan satisfacer las arquitecturas de SO monolítica y microkernel?
La estructura monolítica es la más común en los SO comerciales como Windows, Linux, etc., aquí se ejecuta todo el SO como un solo programa en modo kernel, donde cada procedimiento en el sistema puede llamar libremente a cualquier otro. No oculta información y todos los procedimientos son visibles para cualquier otro procedimiento.
La estructura de microkernel tiene la ventaja de elegir qué colocar en modo kernel y qué en modo usuario. Aquí suelen colocarse en kernel solamente los componentes esenciales. Esto es beneficioso porque un error en el kernel puede paralizar todo el sistema, entonces es conveniente colocar lo menos posible ahí. Divide el SO en módulos pequeños, solo el microkernel se ejecuta en kernel y el resto en modo usuario.

7)- Dónde se implementan los threads: en el espacio de usuario o kernel? Si especifica varias alternativas especifique ventajas y desventajas de cada una.
Los threads se implementan tanto a nivel de usuario como a nivel kernel. A nivel de usuario cada proceso posee una tabla para el control de los hilos. La ejecución de estos es más rápida porque no necesitan de llamadas al sistema para cambiar de contexto y tienen su propio scheduling. Pero en el kernel no puede interferir en este último. A nivel kernel, los threads pueden sincronizarse para lograr determinismo, planificarse y en vez de tener una tabla de hilos por procesos, posee una sola tabla en general.

8)- Detalle las ventajas y desventajas de cada arquitectura de sistemas operativos: microkernel, etc.
Las arquitecturas más usadas en los SO son el monolítico y el microkernel. El sistema monolítico tiene como objetivo ejecutar la mayoría del sistema operativo en el kernel, teniendo así una estructura de 3 capas. La primera está el procedimiento principal que llama a servicios. Estas que están en la segunda capa ejecutan las llamadas al sistema y necesitan de procedimientos utilitarios para por ejemplo obtener datos del programa. El problema (desventaja) está en que al ejecutarse todas las funcionalidades del SO como si fuera un programa, al querer agregar una nueva funcionalidad hay que compilar todo el programa de nuevo. Pero pueden lograr portabilidad del sistema. Tiene tres grandes inconvenientes: el tamaño del núcleo, la falta de extensibilidad y la mala capacidad de mantenimiento.
Por otro lado, el microkernel trata de ejecutar la mayoría del SO afuera del kernel, es decir en modo usuario. Al tener cada funcionalidad en módulos pequeños y separados, es más fácil depurarlos. Además, si se corrompe alguna funcionalidad, el SO puede seguir funcionando y no se bloquea.

9)- A- Si un proceso crea a su hijo vía fork(), es posible que un error del hijo afecte al proceso padre? B- Y viceversa?
~~Si un proceso hijo tiene un error, este no afectaría al padre, ya que el espacio de direcciones es distinto. Pero si al padre se le produce un error, el hijo si puede tener errores ya que es una copia del el mismo y hereda el contexto en el que inicializó el padre. (Analizar todas las circunstancias y lo que puede suceder con fork, wait, exec).~~
Solo podría afectarlo en el caso de que el padre espere al hijo mediante wait() o waitpid(). Un error en el proceso padre no afecta al hijo.

10)- Qué estado agregaría al diagrama de estados de los procesos si el sistema operativo utilizara swapping? (Considere cuáles son los procesos que pueden sacarse de memoria principal).
Agregaría un estado llamado inactivo o en desuso, el proceso que no esté cargado en memoria puede permanecer en el estado bloqueado hasta que se cargue la parte de este que necesite en memoria, mientras que los procesos que ya no utilicen alguna parte de este pero aún no han terminado pueden

pasar a este estado nuevo, de esta forma se puede saber cuáles partes de los procesos pueden sacarse de memoria principal para darle lugar a uno que necesite ejecutarse.
~~Un diagrama de estados de procesos tiene los estados bloqueado, listo, en ejecución siempre que estén activos, es decir en memoria principal. Se le puede agregar teniendo en cuenta el swapping, los procesos inactivo que se almacena en el disco, esperando tener lugar en memoria principal. (Buscar en libro y diapositivas).~~

11)- Cuántos procesos se crean cuando se ejecuta este programa?

```
main(int argc, char ** argv){
doFork(3)
}
doFork(n){
if (n>0)
{ fork(); doFork(n-1);}
}
```

n=1 fork() – 2 procesos
n=2 fork() – 2 procesos + doFork(1)=2*2=4 procesos
n=3 fork() – 2 procesos + doFork(2)=2*4=8 procesos
En total se crean 8 procesos.

12)-Cuáles son los distintos criterios que utilizan los mecanismos de scheduling?
Los distintos criterios son:

- Utilización de la CPU: se desea mantener a la cpu tan ocupada como sea posible.
- Tasa de procesamiento: número de procesos que se completan por unidad de tiempo
- Tiempo de ejecución: es el intervalo que va desde el instante que comienza a ejecutarse un proceso hasta el instante en que este finaliza.
- Tiempo de espera: es la suma de los períodos invertidos en esperar en la cola de procesos preparados.
- Tiempo de repuesta: es el tiempo transcurrido desde que se envía una solicitud hasta que se produce la primer repuesta

~~a) Planificador de la CPU o a corto plazo: es el responsable de decidir quién, cuándo, cómo y por cuánto tiempo recibe el procesador un proceso que está preparado para ejecutar.~~
~~b) Planificador a mediano plazo: es el encargado de regir las transiciones de procesos entre memoria principal y secundaria, actúa intentando maximizar la utilización de los recursos. Este planificador puede ser invocado cuando quede espacio libre de memoria por efecto de la terminación de un proceso o cuando el suministro de procesos caiga por debajo de un límite especificado.~~
~~c) Planificador a largo plazo: es un administrador que se encarga de organizar la ejecución con un adecuado planeamiento de recursos para que el trabajo se ejecute ordenadamente y eficientemente según la modalidad de procesamiento. El Scheduler se ejecuta con poca frecuencia, sólo cuando se necesita crear un nuevo proceso en el sistema, cuando termina un proceso, o ingresa un usuario en el sistema, por lo que tiene prioridad máxima para ejecutar. Es el responsable de controlar el nivel de multiprogramación del sistema y el balance de carga del sistema.~~

13) Porque:
a- Variables locales en pila
b- Variables estáticas en .data
c- Variables estáticas junto a globales
a-Es la variable a la que se le otorga un ámbito local. Tales variables sólo pueden accederse desde la función o bloque de instrucciones en donde se declaran. En la mayoría de lenguajes de programación las variables locales son variables automáticas almacenadas directamente en la pila de llamadas. Esto significa que cuando una función recursiva se llama a sí misma, las variables locales reciben, en cada instancia de la función, espacio para el direccionamiento de memoria separada.
b-Son un tipo especial de variable local y que permite conservar el valor de la variable hasta la próxima llamada de la función. En este caso, llamadas recursivas a la función también tienen acceso a la variable.
c-Las variables estáticas en funciones globales pueden considerarse variables globales, dado que su valor permanece en la memoria durante todo el tiempo de vida del programa. La única diferencia es que sólo pueden accederse desde una única función

14) Que eventos interrumpen el flujo de ejecución:
a- Eventos sincrónicos o asincrónicos
b- Que estado debe almacenarse y dónde?
~~a y b) Cuando el tiempo de respuesta es muy pequeño o mucho menor que el tiempo de duración total de la transacción, tiene sentido esperar por la respuesta (aquí se interrumpe la ejecución) y entonces podemos invocar al proceso de forma sincrónica. Este evento invoca a un módulo, rutina o función devolviendo un resultado en un área de intercambio de datos prácticamente de inmediato.~~
a-Pueden ser tanto eventos sincrónicos (excepción y llamada al sistema) como asincrónicos (interrupciones).
b-Debe almacenarse en la pila del kernel del proceso el estado en si de este y su program counter.

15) Mecanismos del SO para proteger código y datos de aplicaciones de usuario
Instrucciones privilegiadas y protección de memoria
~~• Establecimiento de una política de control de accesos. Incluyendo un sistema de identificación y autentificación de usuarios autorizados y un sistema de control de acceso a la información.~~
~~• Definición de una política de instalación y copia de software.~~
~~• Uso de la criptografía para proteger los datos y las comunicaciones.~~
~~• Uso de cortafuegos (FireWall)~~
~~• Definición de una política de copias de seguridad.~~

16)- Una dirección virtual es traducida por el SO en una o más direcciones físicas.
No, una dirección virtual debe siempre traducirse en una única dirección física, ya que de lo contrario se violaría la protección de un proceso. Dos direcciones virtuales pueden apuntar a una dirección física que es el caso de la memoria compartida.

17)- Indique de qué manera el SO asegura la protección de las aplicaciones.
El SO asegura la protección de las aplicaciones utilizando un registro llamado límite que evita que cualquier otra aplicación pueda salir de su límite de su memoria. Otro método que utiliza es marcando tales segmentos como ocupados, cuando son utilizados por una aplicación, utilizando un bit. Además ofrece otra protección con otro bit que indica si la sección de memoria es de lectura / escritura.

18)- La fragmentación es solo un problema que surge por el uso de segmentos.
No, además de la segmentación que produce *fragmentación externa* (son los espacios de memoria que no están asignados pero que tampoco pueden usarse), existe la paginación, que produce *fragmentación interna* (son los espacios libres de memoria asignados a procesos, pero que éstos no ocupan).

19)- Especifique ventajas y desventajas del uso de segmentación + paginación.
Ventajas:
-Se garantiza la facilidad de implantar la compartición y enlaces.
-Se simplifican las estrategias de almacenamiento.
-Se elimina el problema de la fragmentación externa y la necesidad de compactación.
Desventajas:
-Las tres componentes de la dirección y el proceso de formación de direcciones hace que se incremente el costo de su implementación. El costo es mayor que en el caso de de segmentación pura o paginación pura.
-Se hace necesario mantener un número mayor de tablas en memoria, lo que implica un mayor costo de almacenamiento.
-Sigue existiendo el problema de fragmentación interna de todas o casi todas las páginas finales de cada uno de los segmentos. Bajo paginación pura se desperdician solo la última página asignada, mientras que bajo segmentación paginada el desperdicio puede ocurrir en todos los segmentos asignados.

20)- La segmentación posibilita que los hilos de ejecución compartan todos los segmentos de un proceso.
La segmentación permite dividir la memoria en segmentos lógicos para el programa código, datos, heap, stack de esta manera cada tabla de segmentos está bien delimitada y permite que los hilos compartan los segmentos que necesiten del proceso.

21)- Los hilos de ejecución terminan cuando el thread main termina con exit().

Los hilos pueden terminar antes que el thread main finalice.

22)- Los segmentos a diferencia de las páginas pueden crecer.
A diferencia de la paginación en donde las páginas y marcos de páginas tienen exactamente el mismo tamaño, la segmentación permite que los segmentos crezcan dinámicamente según las necesidades del programa en ejecución.

23)- El mecanismo de tabla de páginas invertidas utiliza una tabla hash por cada proceso.
Con el mecanismo de tabla de paginas invertidas, cada vez que llega un proceso se hace un hash con el proceso ID y el número de página: hash(PID,n°pag) y de esa manera se obtiene el frame de referencia. El tamaño de la tabla hash es proporcional al tamaño de la memoria física.

24)- En un sistema de paginación donde las direcciones virtuales son de 32 bits y las páginas de 4 MB, cuantas entradas tendrá la tabla de páginas, si la misma tiene un nivel.
4MB = 2^22bytes ⟶ 22 bits para offset ⟶ 10 bits para acceso a la tabla.
La tabla de paginación tendrá 2^10 = 1024 entradas

25)- ¿Por qué necesita mantener el sistema operativo una pila a nivel de usuario y otra a nivel de kernel?
Necesita ambas pilas, ya que la pila a nivel usuario es utilizada por la aplicación, y la pila a nivel kernel es utilizada cuando la aplicación produce una interrupción o llamada al sistema, donde se almacena la información de a donde se debe volver el proceso, luego de finalizado el proceso en el nivel kernel. Es necesaria la pila a nivel de kernel por confianza y seguridad, ya que en un multiprocesador puede haber otros hilos corriendo del mismo proceso que modifiquen la memoria durante la llamada al sistema, por lo que al usar modo kernel se evita que el programa en modo usuario pueda modificar la dirección de retorno.

26)- A- La tabla de excepciones define un conjunto de rutinas a ejecutar según el evento que ocurra. Alguno de los números que contiene la tabla son señalados por los diseñadores de los procesos y otros, por los diseñadores de los sistemas operativos. Ejemplifique algunos eventos de cada grupo. B- Cuando ocurre un evento (excepción, llamada al sistema o interrupción), se ejecuta una rutina específica, luego de la cual, el control puede retomarlo la instrucción que generó el evento, la instrucción siguiente o puede abortar. Ejemplifique las tres situaciones.
A- Si los eventos son definidos por los diseñadores del procesador, tenemos por ejemplo una interrupción por timer o división por cero. En tanto que si los eventos son definidos por los programadores del SO tenemos por ejemplo una solicitud de un archivo (llamada al sistema).
B- Una llamada al sistema de por ejemplo una solicitud de un archivo, devuelve el control a la siguiente instrucción. Una interrupción por timer corta un proceso. Cuando este proceso vuelve a adquirir el procesador continua en la instrucción en la que fue interrumpida. Al producirse una división por cero el programa aborta (finaliza).

27)- Especifique en que caso(s) round robin resulta un método que tiene un tiempo de finalización promedio pésimo.
Round Robin tiene un tiempo de finalización promedio pésimo cuando su utiliza un quantum muy pequeño (se gastá demasiado tiempo en cambios de contexto) y cuando todos los procesos tienen el mismo tiempo de ráfaga de CPU (mismo tiempo que FIFO). (Si el quantum es muy grande se parece al FIFO y éste por lo general tiene un tiempo de respuesta promedio malo cuando tiene varias tareas de distintos tamaños).

28)- Especifique si el usuario es capaz de manipular, en alguna situación, direcciones físicas o si solo puede manipular direcciones virtuales a través del código.
Las aplicaciones de usuario siempre manejan direcciones virtuales. El hardware (MMU) luego se encarga de mapearlas a direcciones físicas.
~~Por lo general el usuario solo se maneja con direcciones virtuales, por el motivo de utilizar lenguajes de alto nivel. Pero en lenguaje ensamblador por ejemplo (Assembler) el usuario puede acceder a espacios de memoria en forma absoluta.~~

29)- Luego de que un trap es procesado por el SO, pueden ocurrir tres eventos:

a-El control retorna a la instrucción que produjo el trap

b-El control retorna a la instrucción siguiente a la que produjo el trap

c-El programa aborta

Ejemplifique y justifique las 3 situaciones

a- Interrupción de hardware. El procesador interrumpe el proceso activo por una interrupción del mouse por lo que después de completar la tarea de I/O retorna a la instrucción donde se produjo el trap.

b- Llamada al sistema. El proceso necesita saber la hora del sistema, el so toma el control pasando a modo kernel y retorna al proceso en la siguiente instrucción.

c- Excepción. El proceso divide por cero por lo que produce un error fatal y el proceso no puede continuar.

30)- El mecanismo de round robin tiene un tiempo de respuesta promedio mayor que SJF. Determine si es V o F y justifique

Falso. Puede ocurrir en el mecanismo de SJF que haya un proceso que nunca se ejecute por ser demasiado largo y que continuamente entren procesos con menor tiempo de ejecución por lo que el proceso más largo nunca se ejecutaría y el promedio de repuesta podría decirse que tendería a infinito frente a round robin.

31)- El context switch entre threads es menos costoso, usualmente, que el context switch entre procesos. Justifique

Si los threads pertenecen al mismo proceso tardará menos que un cambio de contexto entre procesos, ya que los hilos de un mismo proceso comparten el código y los datos de este.

32)- Determine si es V o F la siguiente afirmación: varios procesos pueden ejecutar el mismo programa concurrentemente mientras que el mismo proceso puede ejecutar varios programas secuencialmente.

Se puede tomar como falso si se tiene en cuenta la definición que un proceso es una instancia de un programa solamente, por lo cuál no puede ejecutar varios programas secuencialmente. Si no se toma una definición tan estricta sino que se toma como que un proceso es una instancia de cualquier programa entonces es verdadero.

33)- Determine si es V o F la siguiente afirmación: una biblioteca a nivel de usuario implementa cada llamada al sistema ejecutando primeramente una instrucción "evolucionar a modo kernel". La rutina a nivel de usuario luego invoca la función adecuada a nivel de kernel.

Falso. La biblioteca a nivel de usuario realiza una llamada al sistema y el sistema operativo toma el control, con esto se cambió de modo usuario a kernel donde el manejador de traps llama a la función correspondiente y solo devuelve el resultado al programa para retorna a modo usuario.

34)- Que propiedades tiene el mecanismo de páginas invertidas que lo hace más eficiente que a un esquema de paginación convencional? Justifique y ejemplifique

En el sistema solo habrá una tabla de páginas invertida y esta solo tendrá una entrada por cada frame en la memoria física, por lo que la principal ventaja de este método es que reduce la memoria física ocupada por la tabla de páginas ya que sino se tiene una tabla de páginas por proceso. Pero por otra parte aumenta el tiempo de búsqueda de páginas ya que se debe explorar la tabla cada vez que hay una referencia a una página, debido a que esta se encuentra ordenada según la memoria física y las búsquedas se realizan según la memoria virtual.

35)- Indique si las direcciones impresas son virtuales o físicas:

```
Int main (….){
        Printf("dirección código", (void*)main);
        Printf("dirección heap", (void*)malloc(1));
        Int x=3;
        Printf("dirección stack", (void*)&x);
        Return x;
}
```

Dirección de código: 0x1095afe50
Dirección de heap: 0x1096008c0
Dirección de stack: 0x7fff691aea64

Todas las direcciones son virtuales ya que siempre en lenguajes de alto nivel se manejan este tipo de direcciones.

36)- Indique para qué puede utilizarse el mecanismo de semáforos.
Los semáforos es un mecanismo utilizado para la sincronización entre hilos. Pueden ser usados tanto para exclusión mutua (como los locks), o como variables de condición, para esperar a que otro hilo haga algo.

37)- En un esquema de monoprogramación una aplicación puede acceder a toda la memoria física. Determine si es V o F. Justifique.
Verdadero. En un esquema de monoprogramación al ejecutarse una aplicación esta tiene toda la memoria disponible para su acceso, inclusive la ocupada por el SO.

38)- Es posible que dos procesos compartan memoria cuándo se usan tablas de múltiples niveles?
Si, se hace de la misma manera que con paginación de un solo nivel, haciendo que múltiples páginas apunten al mismo marco físico.

39)- Justifique si la siguiente afirmación es V o F: el uso de páginas de menor tamaño provoca una tabla de páginas más grande.
Verdadero. Ya que se deben utilizar más bits para codificar un mayor número de páginas.

40)-Asocie los conceptos de la columna A con las definiciones de la columna B

	A	B
1	**Sincronización**	**Porción de código que sólo un thread puede ejecutar en un momento dado**
2	**Exclusión mutua**	**Asegurarse de que sólo un thread realice una funcionalidad en un momento dado**
3	**Sección crítica**	**Usar operaciones atómicas para asegurar la cooperación entre threads**

- Sincronización: usar operaciones atómicas para asegurar la cooperación entre threads.
- Exclusión mutua: Asegurarse de que sólo un thread realice una funcionalidad en un momento dado
- Sección crítica: porción de código que sólo un thread puede ejecutar en un momento dado.

41)- ¿Asumiendo un tamaño de página de 4KB y cada entrada de la tabla de páginas de 4 B, cuántos niveles de tablas de páginas son necesarios para representar un espacio de direcciones de 32 bits, considerando cada tabla de páginas ocupa una página? Justifique
Se necesitarían 3 niveles de páginas, aunque no quedarían todas las tablas de página del mismo tamaño, habría 2 tablas de página con un tamaño de 4KB cada una y la restante de 2KB.

42)- Responda con 1 oración:

a. Qué estructura en Unix se utiliza para gestionar los sectores asociados a un archivo dado?
b. ¿Cómo se denomina la porción de dato mínima en una unidad de almacenamiento externa?
c. ¿Cómo se llama la componente de hardware que traduce una dirección virtual en física?
d. Qué llamada al sistema (que usted conoce) se invoca cuando el usuario tipea .”/prog”

a- Inodo

b- Cluster? Buscar
c- El memory management unit (MMU) o unidad de gestión de memoria.
d- Buscar

43)- El concepto de memoria virtual es útil aún cuando el espacio de memoria utilizado para ejecutar un conjunto de programas es menor que el espacio de memoria física disponible. Justifique
Si ya que además de la ilusión de memoria infinita la traducción de memoria brinda estrategias para la seguridad, como puede ser el método de base y límite

44)- Explique de qué manera el mecanismo de scheduling de colas múltiples (multi-level feedback scheduler) puede aproximar al mecanismo SRTF
Existe el mecanismo llamado colas múltiples retroalimentadas que consiste en que los distintos procesos puedan pasar de una cola a otra según su duración, por lo que tendrán más prioridad las colas que traten a procesos de menor duración y a medida que lleguen procesos con mayor duración se van asignando a colas con menor prioridad.

45)- Explique de qué manera la traducción de direcciones es una estrategia para proteger a los procesos
Según el esquema de gestión de memoria se brindan estrategias distintas. En el caso más simple se tiene el esquema de base y límite donde cada proceso sólo puede acceder a la memoria comprendida entre estos dos registros (entre base y base+límite), en el caso de segmentación se tiene el mismo principio, pero en vez de aplicada a todo un proceso se aplica a cada segmento de este, logrando de esta manera poder compartir cada segmento por separado. Luego en paginación cada proceso tiene su propia tabla de páginas por lo que se sólo podrá ingresar a las páginas que se encuentren dentro de esta.

46)- Indique si una jerarquía de tablas de páginas siempre ocupa menos espacio que una única tabla de páginas, considerando el mismo espacio de direcciones virtuales. Justifique
Si siempre tendrá menor tamaño la jerarquía de tablas de páginas, a lo sumo si se ocupan todas las entradas en la jerarquía recién ocupará el mismo tamaño que se si se tendría una única tabla de páginas.

47)-Qué características tienen que tener los procesos para que un SJF tienda a FCFS

Todos los procesos tienen que tener la misma ráfaga de CPU, por lo que en este caso en SJF prioriza al que llega primero y tendería a FCFS.

48)-Qué características tienen que tener los procesos y los quantum para que un SJF tienda a Round-Robin

Todos los procesos tienen que tener la misma ráfaga de CPU y los quantum deben ser del mismo tamaño de las ráfagas de CPU de los procesos.

49)-

L=3	R=0	
Hilo A	Hilo B	Hilo C

{
Sem_wait(L);
Putc('C');
Sem_post(R);
}
	{
	Sem_wait(R);
	Putc('A');
	Putc('B');
	Sem_post(R);
		{
		Sem_wait(R);
		Putc('D');
		}

	}	

a)-Está protegida la sección crítica?

Si se consideran los putc como sección crítica no (lo dudo porque no es algo compartido), en todo caso sería correcto decir que no hay sección crítica. Buscar bien

b)-Cuántas veces se mostrará en pantalla 'D'?

No hay un número fijo de veces ya que se produce una condición de carrera.

c)-Cuántas veces como mínimo se mostrará 'A'?

Minímamente se mostrará en pantalla dos veces.

d)-Es válida la secuencia 'CABAB...'?

No. Siguiendo los valores de los semáforos y como va quedando la secuencia:

L=2	R=0	C
L=2	R=1	C
L=2	R=0	CAB
L=2	R=1	CAB

En este punto no se puede ejecutarse de vuelta el hilo B por la propiedad de los semáforos de la espera acotada, por lo que debe ejecutarse el hilo C, como en la secuencia sigue una 'A' perteneciente al hilo B puede que se ejecute el hilo C y antes del Putc('D') cambie de contexto, pero así tampoco sería posible que se ejecute el hilo B ya que R sería igual a 0.

e)-Es válida la secuencia 'CABACDBCABDD'?

No. Misma explicación que el anterior ya la secuencia 'CABA' no es válida

50) Considerando la siguiente dirección de 64 bits

Seg	TP	TP	TP	Pag	Offset
6 bit	11 bit	11 bit	11 bit	11 bit	14 bit

a)- Cuál es el tamaño de cada página?

Cada página tiene 2^{14}

b)-Cuántos segmentos hay?

Hay en total 2^6 segmentos

c)-Cuál debería ser el tamaño de cada entrada si se hace que cada tabla de páginas ocupe una página?

No estoy seguro pero si cada tp ocuparía 14 bits para cada entrada habría disponible 2 bits solamente por lo que cada entrada sería de 2^2 bits

d)-Cuál es el máximo de memoria que se puede ocupar?

Calcular no se si tener en cuenta lo que ocupan las tablas de segmentos y de páginas.