Conteste verdadero o falso y justifique su respuesta:

1. Los procesos que se encuentran en la cola de listos no están ejecutándose, por tal motivo es incorrecto decir que son procesos.   
     
   Falso. Los procesos que están en la cola de listos son procesos que ya se encuentran cargados en memoria. La definición de proceso es “porción de un programa que se encuentra cargado en memoria”. No necesariamente deben estar en ejecución para ser considerados un proceso, pueden estar bloqueados, suspendidos, listos…
2. Si tengo un sistema de tipo I/O bound resulta conveniente programarlo con threads ULT.   
     
   Falso. Al usar hilos ULT, si uno de los hilos se bloquea, todos los demás hilos se bloquearán también. Se podría usar jacketing para evitar este problema, pero el jacketing involucra tener que reescribir la biblioteca de llamadas al sistema, y esto es ineficiente. Si el sistema operativo admite hilos KLT, lo mejor sería utilizar aquellos hilos, para así no disminuir la eficiencia.
3. Si un proceso está realizando cálculos (CPU Bound), este puede ser interrumpido temporalmente por una interrupción de software proveniente de otro proceso de mayor prioridad.   
     
   Falso. Las interrupciones de software son generadas por el programa que está en ejecución. Si el proceso que se está ejecutando actualmente es el programa que realiza cálculos, ningún otro proceso salvo él mismo podrá generar una interrupción de software.
4. Los procesos livianos poseen su propio espacio de direcciones, por lo tanto, para compartir información entre ellos deben recurrir a mecanismos especiales provistos por el SO.  
     
   Falso. Los hilos comparten el espacio de direcciones, los archivos y recursos del proceso que los genera. Es decir, un hilo tiene acceso a la misma área de memoria, al mismo conjunto de datos, variables y demás. Es por esto que no es necesario utilizar ningún mecanismo para comunicarse entre hilos de un mismo proceso, simplemente pueden usar los recursos compartidos que tienen para hacerlo (e.g: escribiendo en la misma área de memoria).
5. Los compiladores permiten convertir un programa en proceso.  
     
   Falso. El compilador toma el programa/código fuente y lo traduce a un programa objeto. Este programa objeto luego debe ser linkeditado, es decir, enlazar las referencias del programa objeto con sus bibliotecas, lo que genera un programa ejecutable.   
     
   La ejecución de dicho programa ejecutable es lo que generaría un proceso. La compilación no es la generación de un proceso, sino la traducción de un programa fuente.
6. ¿Qué son las interrupciones? Explique por qué algunos autores utilizan el término “interrupción de software” y qué significado tiene. Mencione además que es el vector de interrupciones.   
   Una interrupción es la detención del proceso en ejecución debido a una condición externa al procesador. Es decir, el procesador es forzado a reconocer la ocurrencia de un evento en el sistema mediante una señal.   
     
   Las interrupciones por software son interrupciones generadas por el mismo proceso en ejecución. Generalmente, este tipo de interrupciones son llamadas al sistema (syscalls), las cuales son ejecutadas cuando el proceso necesita realizar una petición que solo puede ser resuelta con ayuda del kernel de por medio. Por ejemplo, una petición de E/S será una syscall, ya que es el sistema operativo el encargado de administrar y gestionar los periféricos, el proceso no tiene control sobre los mismos.   
     
   Son llamadas interrupciones de software ya que son producidas por un software, un programa en ejecución.   
     
   El vector de interrupciones es un área en memoria que contiene punteros a cada una de las rutinas de interrupción. Cada entrada del vector de interrupciones es la dirección de una rutina de atención a la interrupción. Las rutinas nos dicen cómo debemos manejar una interrupción, es, qué es lo que debemos hacer.   
     
   7. Explique las diferencias entre un algoritmo de planificación de procesos con prioridades fijas y uso de prioridades variables. ¿Podría un algoritmo utilizar ambos tipos de prioridad?   
     
   Un algoritmo de planificación con prioridades fijas se caracteriza por mantener la misma prioridad a un proceso con la cual fue asignada desde el comienzo. En cambio, un algoritmo de prioridades variables cambia la prioridad de dicho proceso en base a algunos parámetros (como por ejemplo, el tiempo de ejecución, el tiempo de uso de E/S, si es un proceso corto, etc.).   
     
   Los algoritmos de planificación son ideados e implementados por aquellos que estén a cargo del desarrollo del S.O. Tranquilamente alguien podría crear un algoritmo que utilice ambos tipos de prioridad.   
     
   Uno podría crear un algoritmo de planificación que posea colas de distintas prioridades para los procesos, y que los procesos posean prioridades fijas. El algoritmo podría ejecutar los procesos de ciertas colas con más prioridad, pero aun así guiarse por la prioridad fija del proceso dentro de la cola para elegir cuál ejecutar. Así mismo, un proceso podría ser enviado a una cola de prioridad menor por cierto motivo (¿mucho uso de E/S?).