**¿V o F? Sobre HILOS**

1. **En un sistema que cuentan con un solo procesador y dicho procesador no cuenta con múltiples hilos de ejecución no se obtienen mejoras al dividir los procesos pesados en threads**

FALSO. no es el procesador el que tiene que poder soportar el multithreading, sino el sistema operativo. “Multithreading refers to the ability of an OS to support multiple, concurrent paths

of execution within a single process” (stallings 8th p159)

1. **Los UTL tienen la ventaja de que implican un menor overhead de planificación en comparación con los KLT.**

VERDADERO

Thread switching does not require kernel mode privileges because all of the thread management data structures are within the user address space of a single process. Therefore, the process does not switch to the kernel mode to do thread management. This saves the overhead of two mode switches (user to kernel; kernel back to user). (stallings 8th p 167)

1. **Cuando en un proceso se usan Threads siempre se necesita la estructura PCB u otra similar,**

**Independientemente de que sean KLT o ULTs.**

VERDADERO

En un modelo de proceso monohilo, la representación de un proceso incluye su bloque de control de proceso y espacio de direcciones de usuario, pilas de usuario y núcleo para gestionar el comportamiento de las llamadas/retornos en la ejecución de los procesos

En un entorno mutihilo, sigue habiendo un único bloque de control del proceso y un espacio de direcciones de usuario asociado al proceso, pero ahora hay varias pilas separadas para cada hilo, asi como un bloque de control para cada hilo que contiene los valores de los registros, la prioridad y otra información relativa al estado del hilo

(stalling pagina 160)

1. **Multihreading se refiere a la habilidad de un sistema operativo para soportar múltiples threads en ejecución dentro de múltiples procesos**

FALSO

“Multithreading refers to the ability of an OS to support multiple, concurrent paths of execution within a single process” (stallings 8th p159)

1. **En la facilidad de KLT puro tiene el código de administración del thread en el area de aplicación (del proceso)**

FALSO  
In a pure KLT facility, all of the work of thread management is done by the kernel. There is no thread management code in the application level, simply an application programming interface (API) to the kernel thread facility. (stallings 8th 168)

1. **El tiempo de creación de un Thread es mucho menor que el de un proceso pesado, debido a que la estructura TCB es más pequeña que el PCB**

Aparentemente verdadero. because very little memory copying is required (just the thread stack), threads are faster to start than processes. To start a process, the whole process area must be duplicated for the new process copy to start.

(http://blogs.datalogics.com/2013/09/25/threads-vs-processes-for-program-parallelization/)

ni stallings (8th y 5th) ni silber (6h 9th) responden esto.

1. **Una de las ventajas del uso de kernel level threads (KLTs) es que el cambio de threads (Thread switch) de un mismo proceso es más rápido que el cambio entre procesos (process switch) ya que los primeros comparten el stack (pila) del proceso.**

FALSO

no es cierto que compartan la pilas, cada thread tiene su propia pila. (stalings 8th p160 fig4.2)

1. **Existe una relación directamente proporcional entre la cantidad de threads de una aplicación y la velocidad de ejecución de la misma**

FALSO. La velocidad no depende directamente de la cantidad de hilos;

1. **Una ventaja de los ULT es que el algoritmo de planificación puede ser propio de cada aplicación, pero como contrapartida debe generar una llamada al sistema por cada cambio de hilo en ejecución, lo que provoca overhead al cambiar de modo usuario a modo kernel.**

Falso, nunca cambia de modo ni genera una syscall.

When control is passed to the library, the context of the current thread is saved, and when control is passed from the library to a thread, the context of that thread is restored. The context essentially consists of the contents of user registers, the program counter, and stack pointers. All of the activity described in the preceding paragraph takes place in user space and within a single process. The kernel is unaware of this activity. The kernel continues to schedule the process as a unit and assigns a single execution state to that process. (stallings 8th p165)

1. **Una de las ventajas de KLT es que el cambio de threads de un mismo proceso es mas rápido que el cambio entre procesos ya que los primeros comparten el stack del proceso.**

Falso, los KLT no comparten el stack sino que tiene cada uno su propio stack. (stallings 8th p160 fig4.2)

1. **noteitch entre hilos del mismo proceso es más rápido que un process switch**

VERDADERO

Thread switching does not require kernel mode privileges because all of the thread management data structures are within the user address space of a single process. Therefore, the process does not switch to the kernel mode to do thread management. This saves the overhead of two mode switches (user to kernel; kernel back to user). (stallings 8th p167)

1. **Ejecutar threads en equipos con un solo procesador no tiene sentido**

FALSO

The key benefits of threads derive from the performance implications:

1.

It takes far less time to create a new thread in an existing process than to

create a brand-new process. Studies done by the Mach developers show that

thread creation is ten times faster than process creation in UNIX [TEVA87].

2.

It takes less time to terminate a thread than a process.

3.

It takes less time to switch between two threads within the same process than

to switch between processes.

4.

Threads enhance efficiency in communication between different executing

programs. (stallings 8th p161)

1. **Cuando se crea un nuevo thread, este tiene acceso a:**

**1-Una copia del segmento de datos del proceso**

**2-Acceso al segmento de código del proceso**

**3-El segmento de datos del proceso**

**4-Una copia de la pila del proceso, pero marcándola previamente como copy-on-write**

**5-Todas las anteriores**

**6-Ninguna de las anteriores**

1 y 2,

Within a process, there may be one or more threads, each with the following:

•A thread execution state (Running, Ready, etc.)

•A saved thread context when not running; one way to view a thread is as an

independent program counter operating within a process

•An execution stack

•Some per-thread static storage for local variables

•Access to the memory and resources of its process, shared with all other

threads in that process

(stallings 8th p160)

1. **Entre la categorías de implementación de hilos ULT y KLT, los KLT presentan respecto a los ULT la ventja de que pueden ejecutar en cualquier SO**

Falso

los ult pueden ejecutar en cualquier so. no se necesita ningun cambio en el nuevo nucleo para dar soporte a los ult. la biblioteca de los hilos es un conjunto de utilidades a nivel de aplicacion que comparten todas las aplicaciones. (stallings 5th Pag168)

1. **No tiene sentido asociar el estado suspendido a los threads**

VERDADERO

El estado de suspensión está asociado a los procesos, no a hilos individuales, puesto que como los hilos comparten el mismo espacio de direcciones de un proceso, se suspenderán todos cuando el proceso al cual pertenecen también lo haga

1. **Una ventaja de los hilos sobre los procesos es la de compartir cierto espacio de memoria con datos en común. Es por esto que el cambio de contexto entre ellos requiere menos tiempo que el cambio de contexto entre procesos, porque hay menos datos para conservar**

Falso.no es por que haya menos datos por conservar, hay que actualizar mas estructuras de control en el so si lo que cambiamos es un proceso y no un thread.

Process switching involves switching out all of the process abstractions and resources in favor of those belonging to a new process. Most notably and expensively, this means switching the memory address space. This includes memory addresses, mappings, page tables, and kernel resources—a relatively expensive operation. Thread switching is much, much cheaper, as it involves switching out only the abstraction unique to threads: The processor state. Switching processor state (such as the program counter and register contents) is generally very efficient. For the most part, the cost of thread-to-thread switching is about the same as the cost of entering and exiting the kernel.

fuente: <https://www.quora.com/How-does-thread-switching-differ-from-process-switching-What-is-the-performance-difference>

1. **Una de las ventajas de los hilos a nivel de usuario es que cuando uno es swapeado o suspendido, el resto de los hilos puede seguir ejecutando**

FALSO

los hilos no pueden pasar nunca a estado suspendido..

1. **La principal ventaja de los user threads es que permiten incrementar la utilización del procesador al utilizarlo aun cunado un user theads de un mismo proceso se encuentra en estado bloqueado**

VERDADERO

Siempre y cuando se utilice la técnica jacketing

1. **Una de las ventajas de los procesos sobre los hilos es que desde un proceso se puede crear otro proceso pero desde un hilo no se puede crear otro**

FALSO

Un hilo de un proceso puede crear otros hilos dentro del mismo proceso, proporcionando un puntero de instrucción y los argumentos del nuevo hilo. El nuevo hilo tendrá su propio contexto y su propio espacio de pila, y pasara a la cola de listos

1. **Los hilos de un mismo proceso comparten las variables globales, el heap, la pila y los manejadores de señales pero no comparten el contador de programa**

FALSO no comparten la pila, cada hilo tiene una pila propia.

Within a process, there may be one or more threads, each with the following:

•A thread execution state (Running, Ready, etc.)

•A saved thread context when not running; one way to view a thread is as an

independent program counter operating within a process

•An execution stack

•Some per-thread static storage for local variables

•Access to the memory and resources of its process, shared with all other

threads in that process

(stallings 8th p160)

1. **No tiene sentido la existencia de un único ULT dentro de la bibilioteca**

VERDADERO

No tiene sentido porque al tener un solo hilo de usuario dentro de un proceso ese hilo seria el procesos en si.

1. **Los hilos comparten siempre el estado del proceso en los estados suspendido y finalizado**

VERDADERO

En los hilos no tiene sentido aplicar estados de suspensión, ya que dicho estado es un concepto a nivel de proceso. Si se expulsa un proceso, todos sus hilos se deben expulsar porque comparten el espacio de direcciones del proceso. (Stalling pagina 162)

Cuando todos los procesos en MP se encuentran en estado bloqueado, el SO puede suspender un proceso poniéndolo en el estado suspendido y transfiriéndolo a disco. El espacio que se libera en MP puede usarse par traer otro proceso. (Stalling pagina 122)

Cuando finaliza un proceso finalizan todos los hilos del proceso

1. **Un ULT puede planificarse utilizando las políticas del Sistema Operativo para los procesos tradicionales.**

Verdadero, un ULT se puede planificar de cualquier manera dentro del mismo proceso, que el algoritmo sea el mismo que el que usa el SO no es una restricción.