

Sistemas Operativos

Threads - I



Sistemas Operativos

- ✓ Versión: Marzo 2018
- ✓ Palabras Claves: Threads, Hilos, ULT, KLT, Procesos, Concurrencia, Paralelismo, Multithreading

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts)



Concurrencia y Paralelismo

- ✓ Es común dividir un programa en diferentes “tareas” que, independientemente o colaborativamente, solucionan el problema
- ✓ Es común contar con un pool de procesadores para ejecutar nuestros programas



Analicemos estas situaciones

- ✓ Procesador de texto: ingreso de caracteres, auto-guardado, análisis ortográfico/gramatical
- ✓ Aplicaciones que muestran una animación, o un gráfico a medida que se ingresan datos
- ✓ Acceso simultáneo a diferentes fuentes de E/S
- ✓ Tendencia de los procesadores actuales a contar con varios núcleos (multiprocesadores)



Los lenguajes de programación

- ☑ Brindan herramientas que nos permiten separar las diferentes “tareas” de los programas en unidades de ejecución diferentes:
 - ✓ Java – heredar de “Thread”, implementar la interface “Runnable”
 - ✓ Delphi – Heredar de “TThread”
 - ✓ C#, C, etc
 - ✓ Ruby – Thread.new{CODIGO}
 - ✓ PHP – Heredad de Thread
 - ✓ Javascript – HTML5 Web Workers



Primeros SO – Procesos

- ✓ Programa en Ejecución
- ✓ Unidad de asignación de los recursos
- ✓ Conceptos relacionados con proceso:
 - ✓ Espacio de direcciones
 - ✓ Punteros a los recursos asignados (stacks, archivos, etc.)
 - ✓ Estructuras asociadas: PCB, tablas
- ✓ Único hilo de ejecución por proceso



SO Actuales - Threads

- ☑ Unidad básica de utilización de CPU
- ☑ Proceso:
 - ✓ Espacio de direcciones
 - ✓ Unidad de propiedad de recursos
 - ✓ Conjunto de threads (eventualmente uno)
- ☑ Thread:
 - ✓ Unidad de trabajo (hilo de ejecución)
 - ✓ Contexto del procesador
 - ✓ Stacks de Usuario y Kernel
 - ✓ Variables propias
 - ✓ Acceso a la memoria y recursos del PROCESO



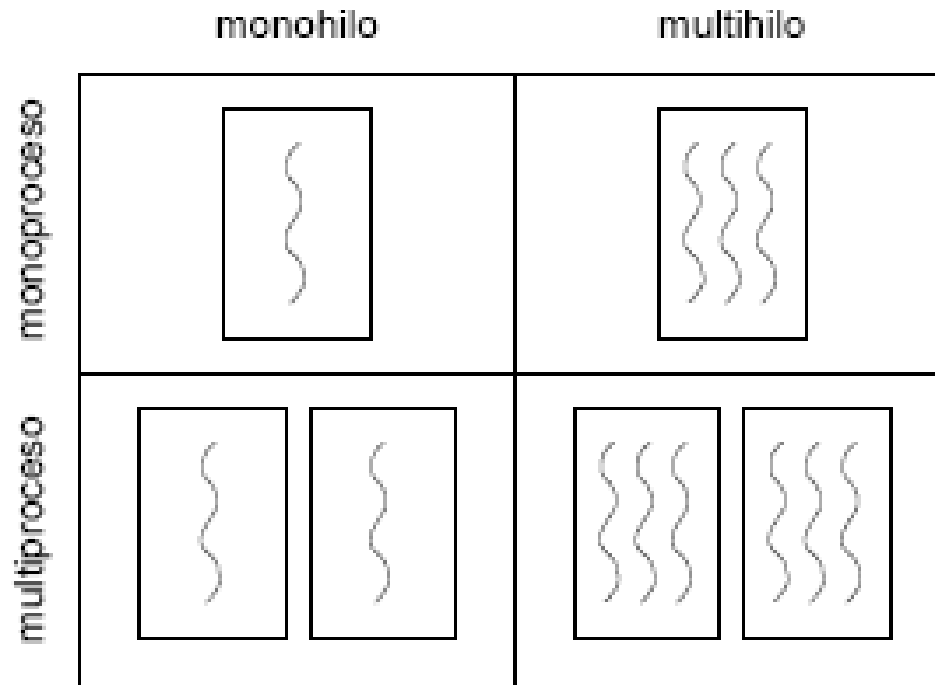
Procesos e Hilos

- ☑ Porqué dividir una aplicación en threads?
 - ✓ Respuestas percibidas por los usuarios, paralelismo/ejecución en background
 - ♦ Ejemplo: El servicio de impresión de Word ejecuta en background y nos permite seguir editando
 - ✓ Aprovechar las ventajas de múltiples procesadores
 - ♦ Con n CPUs pueden ejecutarse n threads al mismo tiempo
 - ♦ Pregunta: Dada una aplicación con un único thread, agregar un nuevo procesador hará que esta se ejecute mas rápido?
 - ✓ Características complejas
 - ♦ Sincronización
 - ♦ Escalabilidad: cantidad de threads por proceso? (# de CPU, excesivos cambios de contexto de hilos del mismo proceso...)



Threads

☑ SO Monothreading vs. Multithreading

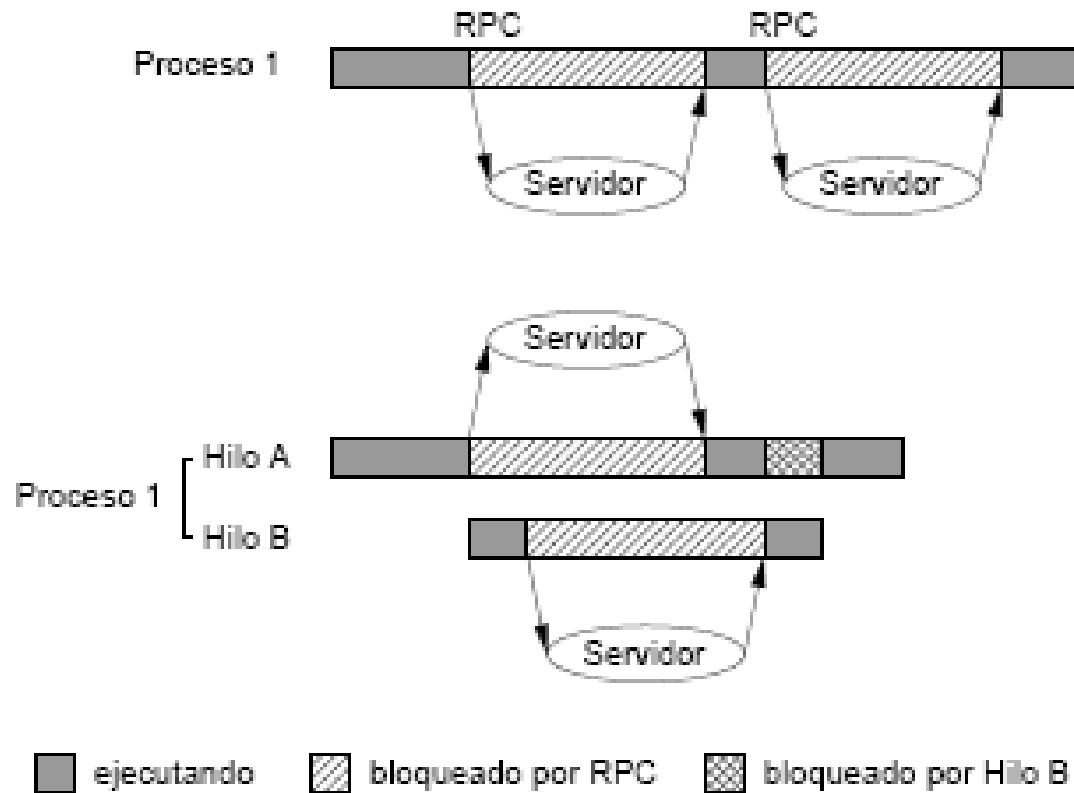


Threads - Ventajas

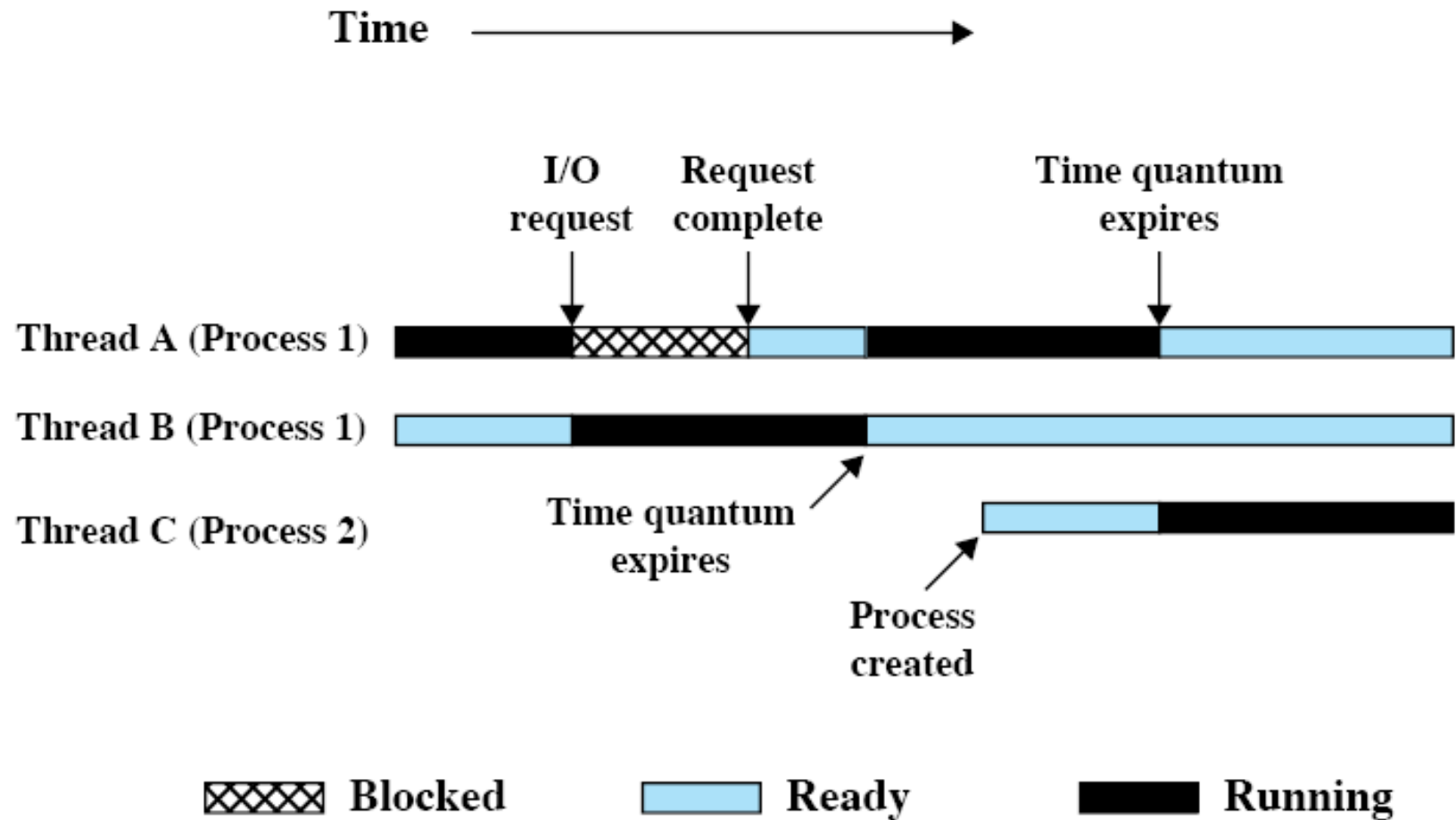
- ✓ Sincronización de Procesos
- ✓ Mejorar tiempos de Respuesta
- ✓ Compartir Recursos
- ✓ Economía
- ✓ Analicemos uso de RPC, o servidor de archivos



Threads- Ejemplo 1



Threads - Ejemplo 2



Algunos conceptos

☑ Hyper Threading

- ✓ Permite al software programado para ejecutar múltiples hilos (multi-threaded) procesar los hilos en paralelo dentro de un único procesador .
- ✓ Simular dos procesadores lógicos dentro de un único procesador físico
 - ♦ Duplica solo algunas “secciones” de un procesador
 - ♦ Registros de Control (MMU, Interrupciones, Estado, etc)
 - ♦ Registros de Proposito General (AX, BX, PC, Stack, etc.)
- ✓ Resultado: mejoría en el uso del procesador (entre 20 y 30%)

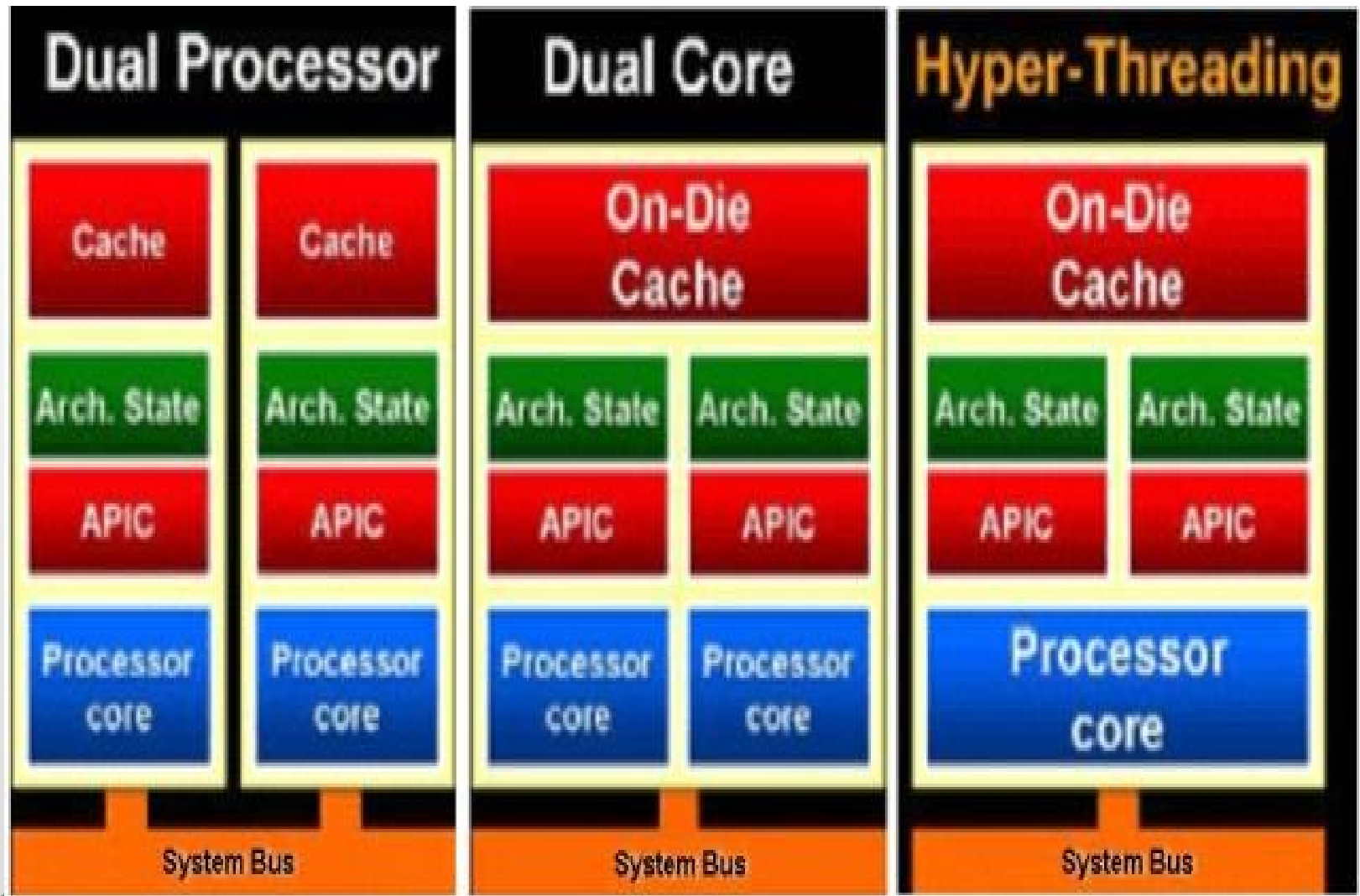


Algunos conceptos

- ✓ Sistemas Dual-core: una CPU con dos cores por procesador físico. Un circuito integrado tiene 2 procesadores completos. Los 2 procesadores combinan cache y controlador.
- ✓ Sistemas Dual-processor (DP): tiene 2 procesadores físicos en el mismo chasis. Pueden estar en la misma motherboard o no. Cache y controlador independientes.
- ✓ En ambos casos, las APIC (Advanced Programmable Interrupt Controllers) están separadas por procesador. De esta manera proveen administración de interrupciones x procesador.



Algunos conceptos



Estructura de un hilo

- ☑ Cada hilo contará con:
 - un estado de ejecución
 - un contexto de procesador
 - Stacks(s)
 - Acceso a memoria y recursos del proceso
 - archivos abiertos,
 - Señales,
 - código
 - datos) que compartirá con el resto de los hilos.
- ☑ TCB – Thread Control Block



Análisis:

☑ Analizar los siguientes puntos respecto a procesos

- Context switch
- Creación

Dstrucción

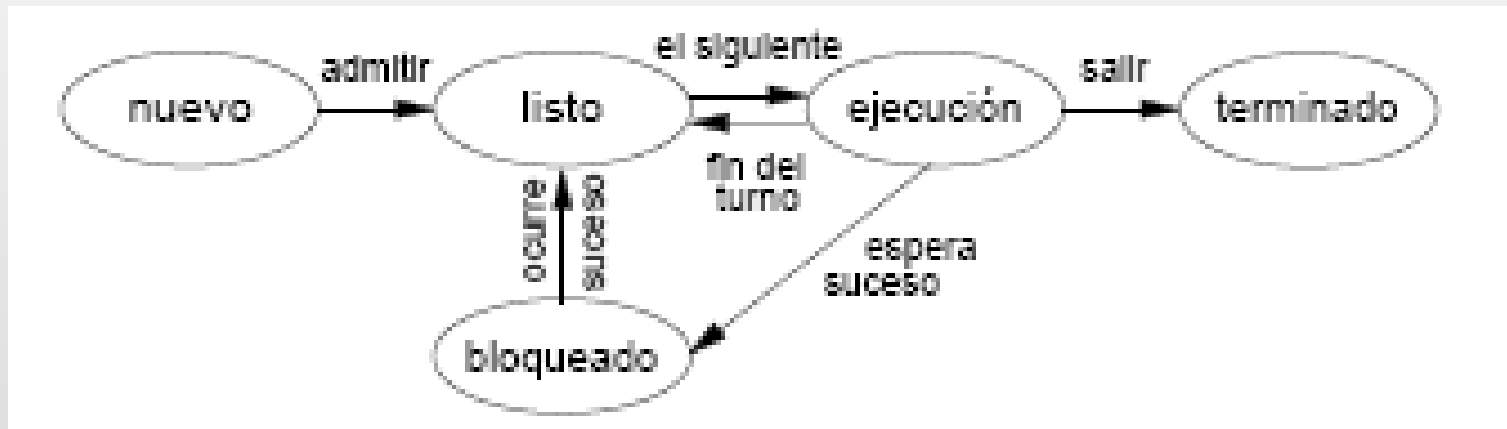
Planificación

Protección



Estados de un Thread

- ✓ Estados Mínimos:
 - Ejecución, Listo y Bloqueado
- ✓ Planificación: sobre los Threads
- ✓ Eventos sobre procesos afectaran todos sus Threads



Tipos - Usuario – ULT

- ☑ User Level Thread
- ☑ La aplicación, en modo usuario, se encarga de la gestión
 - Por medio de una Biblioteca de Threads
- ☑ El Kernel “no se entera” de la existencia de Threads.
- ☑ La Biblioteca deberá brindar funciones para:
 - ✓ Crear, destruir, planificar, etc.
- ☑ Ejemplos:
 - ✓ Java VM
 - ✓ POSIX Threads
 - ✓ Solaris Threads



Tipos - Usuario – ULT

☑ Ventajas

- ✓ Intercambio entre hilos
- ✓ Planificación independiente
- ✓ Portabilidad (no depende del SO multithreading)
- ✓ No requiere cambios modo para su “existencia”

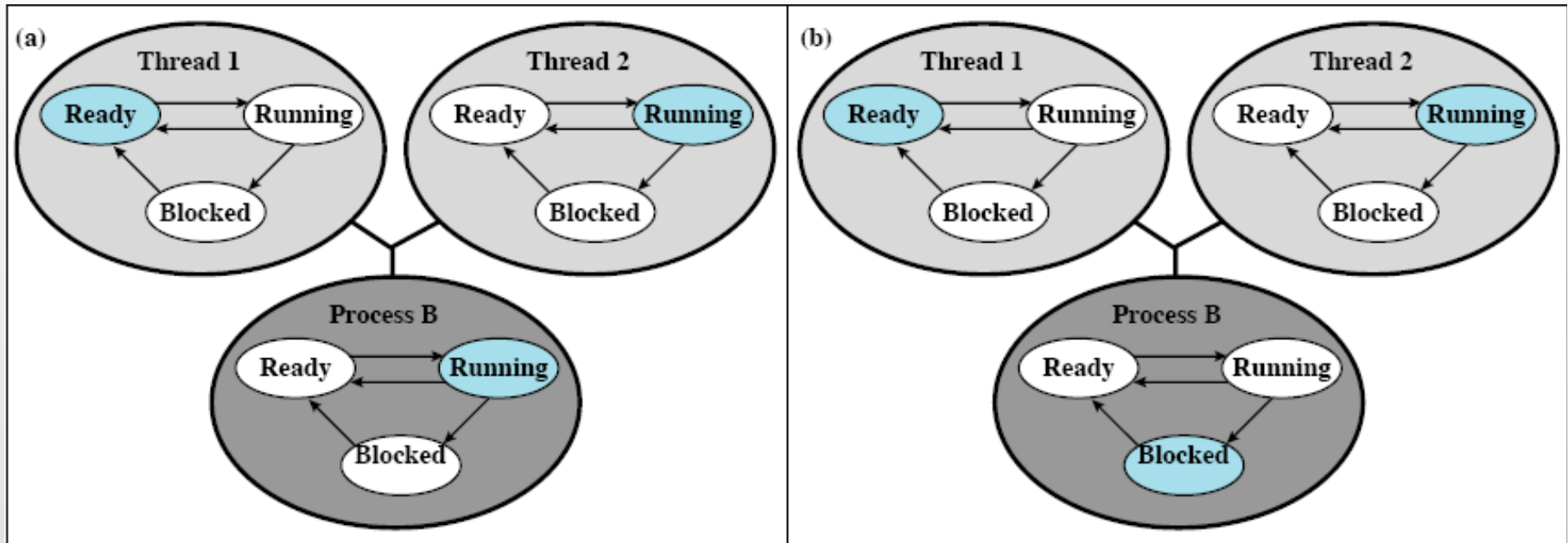
☑ Desventajas

- ✓ Bloqueo del proceso durante una System Call bloqueante
- ✓ No se puede ejecutar hilos del mismo proceso en distintos procesadores



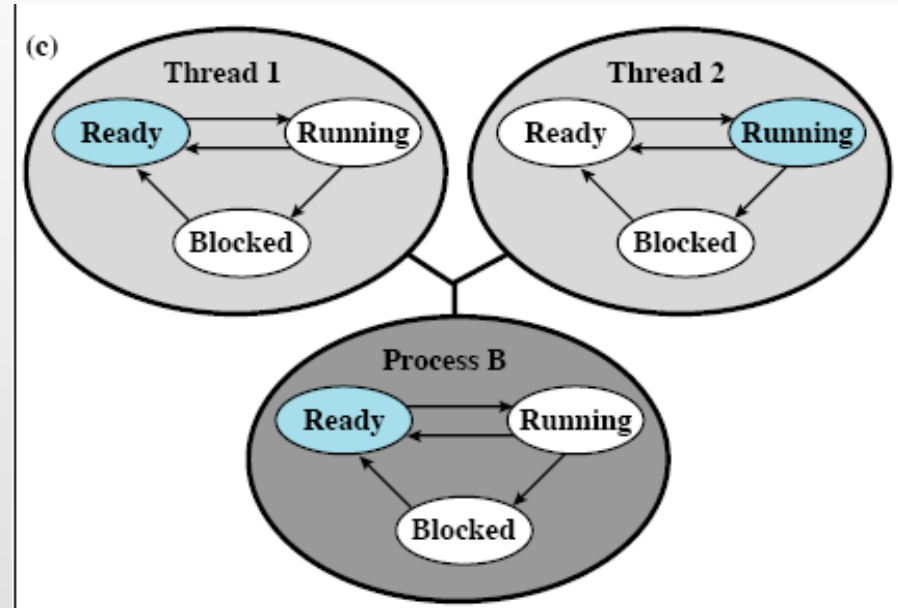
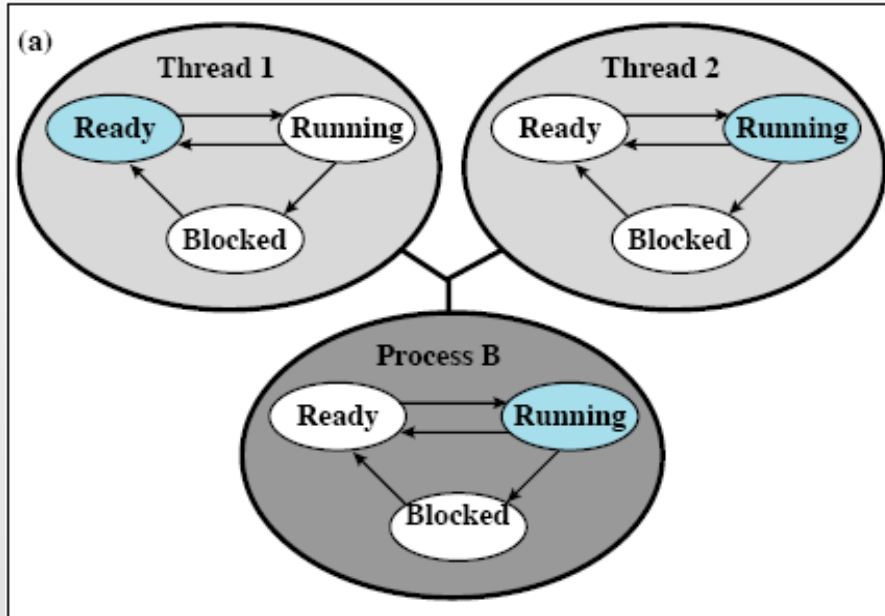
Tipos - Usuario - ULT

☑ Threads State's Vs. Process State's



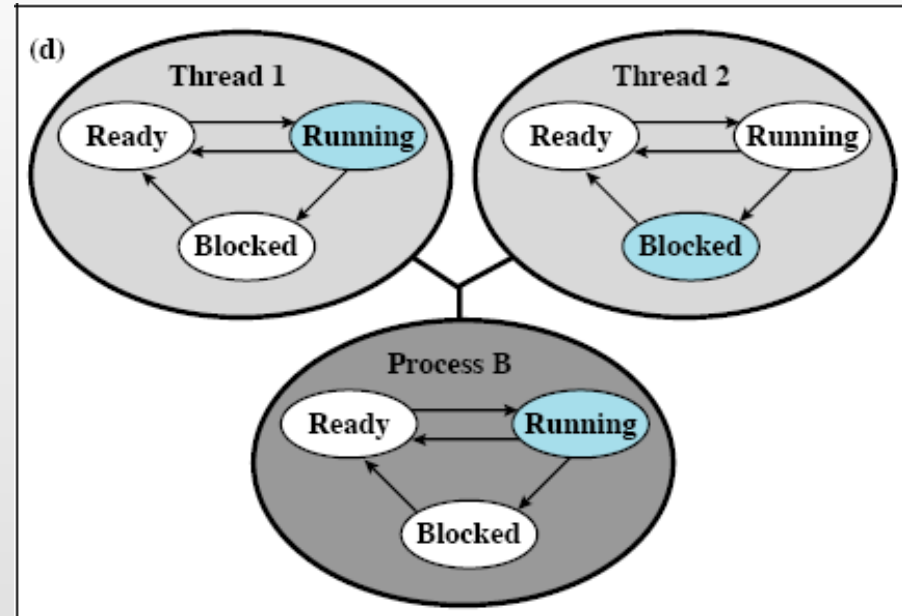
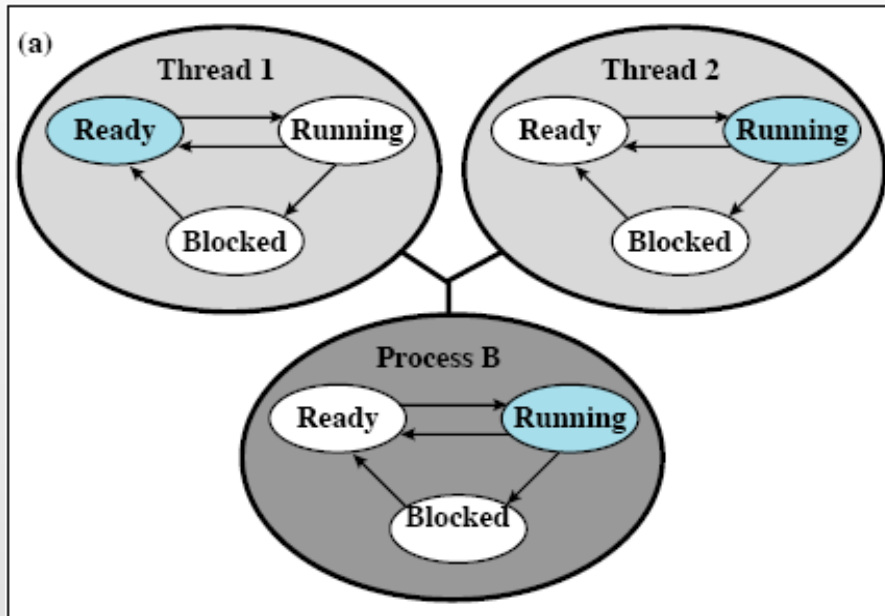
Tipos - Usuario - ULT

☑ Threads State's Vs. Process State's



Tipos - Usuario - ULT

☑ Threads State's Vs. Process State's



Tipos - Kernel - KLT

- ✓ Kernel Level Thread
- ✓ La gestión completa se realiza en modo Kernel
- ✓ Ventajas
 - ✓ Se puede multiplexar hilos del mismo proceso en diferentes procesadores
 - ✓ Independencia de bloqueos entre Threads de un mismo proceso
- ✓ Desventajas
 - ✓ Cambios de modo de ejecución para la gestión
 - Planificación, creación, destrucción, etc.
- ✓ Ejemplos:
 - ✓ Windows NT/2000
 - ✓ Linux



Tipos de Threads - Combinaciones

- ☑ Es posible combinar ULT y KLT
- ☑ ULT
 - ✓ Planificación
 - ✓ Sincronización
- ☑ LWP – Lightweight Process
 - ✓ Asocia ULT y KLT
- ☑ Este enfoque aprovecha las ventajas de ambos tipos



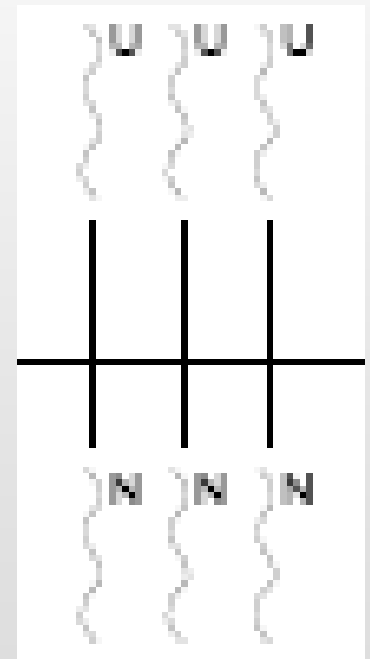
Modelos de Multithreading

- ☑ Relación entre ULT y KLT
- ☑ Tipos
 - ✓ Uno a Uno
 - ✓ Muchos a Uno
 - ✓ Muchos a Muchos



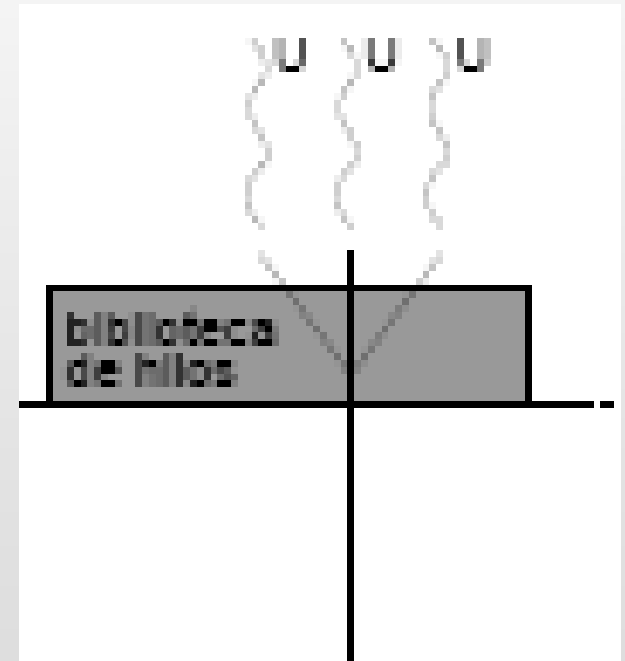
Modelos - Uno a Uno

- ✓ Cada ULT mapea con un KLT
- ✓ Cuando se necesita un ULT se debe crear un KLT
- ✓ Ejemplos: Windows – OS/2



Modelos - Muchos a Uno

- ✓ Muchos ULT mapean a un único KLT
- ✓ Usado en sistemas que no soportan KLT
- ✓ Si se bloquea un ULT, se bloquea el proceso
- ✓ Java sobre un sistema que no soporta KLT



Modelos - Muchos a Muchos

- ✓ Muchos ULT mapean a muchos KLT
- ✓ Ejemplo: Solaris

