Cap. 04 – Processes and Threads

4.1 – Processes and Threads
4.2 – Types of Threads
4.3 – Multicore and Multithreading
4.4 – Windows 7 Thread and SMP Management
4.5 – Solaris Thread and SMP Management
4.6 – Linux Process and Thread Management

Referências Bibliográficas

- Operating Sytems – Internals and Design Principles. William Stallings. 7th, Prentice-Hall 2012.
- Instructor Resources – Operating Systems - 7th http://williamstallings.com/OperatingSystems/OS7e-Instructor/

The basic idea is that the several components in any complex system will perform particular subfunctions that contribute to the overall function.

The Sciences of Artificial, Herbert Simon

4.1 – Processes and Threads

- “**concept of a process**” .. envolve a junção de 02 características:

- “**resource ownership**” .. inclui um espaço de endereço virtual para conter o imagem do processo, ou seja, programa, dados, pilha e atributos definidos no bloco de controle do processo.
- .. sistema operacional executa uma função de proteção para evitar interferência indesejada entre processos com respeito aos recursos.
- “**scheduling / execution**” .. define um caminho (rota) de execução em um ou mais programas e, assim possui um estado que a descreve.
- .. um processo tem um estado de execução, p.ex., “running”, “ready”, etc. e uma prioridade de despacho, bem como a entidade que é agendada e despachada pelo sistema operacional.

… 4.1 – Processes and Threads

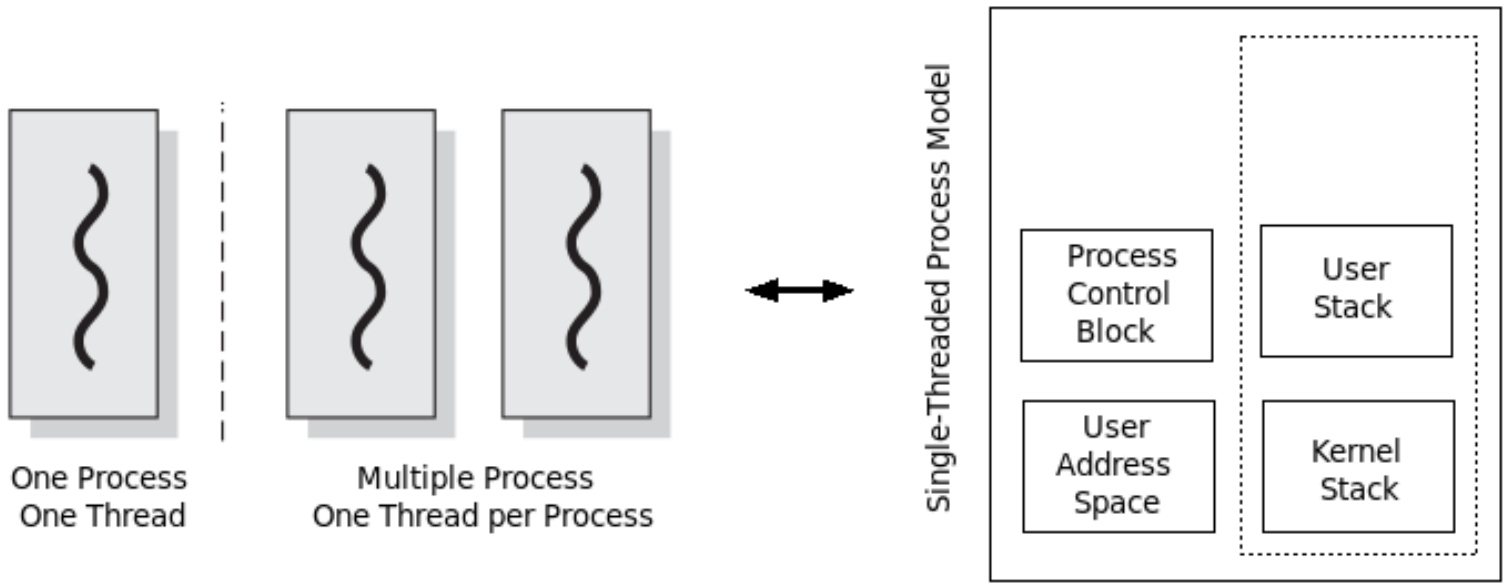
- “**thread or lightweight process**” .. surge da idéia de que em alguns sist. oper. estas duas características são distintas e, portanto, podem ser tratadas de forma independente.
- ... “lightweight processes” acomodados dentro de um processo são caracterizados pelo contador de programa, registradores e pilha.
- ... “lightweight processes” compartilham código, dados e recursos (espaço de endereçamento, variáveis globais, arquivos, etc.).
- ... as “threads” operam como processos e, portanto, possuem estado e podem criar outras “threads”, alocar recursos, etc.

4.1.1 – Multithreading

- “**single-threaded approach**” .. abordagem tradicional de um único encadeamento de execução por processo, em que o conceito de entrelaçamento não é reconhecido.
- “MS-DOS” .. sist. oper. “single-threaded” que oferece suporte a um único processo de usuário e um única “thread” por processo.
- “UNIX” .. oferece suporte a vários processos de usuário, mas apenas a um encadeamento ou “thread” por processo.

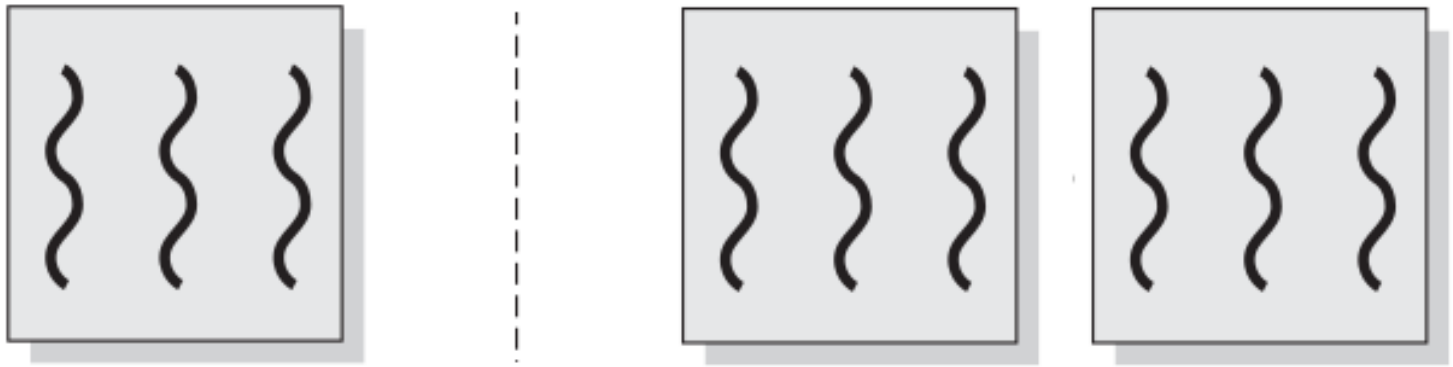
… 4.1.1 – Multithreading

- “**thread**” .. uma maneira de ver uma “thread” é visualizá-la como um contador de programa independente operando dentro de um processo.
- “MS-DOS” .. sist. oper. “single-threaded” que oferece suporte a um único processo de usuário e um única “thread” por processo.
- “UNIX” .. oferece suporte a vários processos de usuário, mas apenas a um encadeamento ou “thread” por processo.



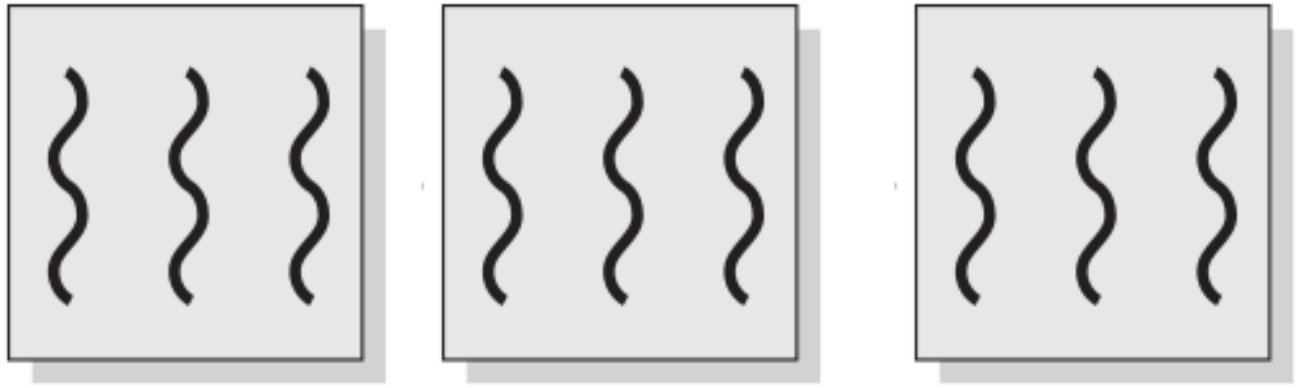
… 4.1.1 – Multithreading

- "**multithreading**" .. refere-se à capacidade de um sist. oper. de oferecer suporte a vários caminhos simultâneos de execução ou unidades escolonáveis em um único processo.
- .. este conceito contempla o uso de vários processos, cada um dos quais oferece suporte a vários threads.
- .. abordagem é usada no Windows, Solaris e em muitas versões modernas do UNIX, entre outros.



… 4.1.1 – Multithreading

- “**multithreaded environment**” .. processo é definido como a unidade de alocação de recursos e uma unidade de proteção.
- .. espaço de endereço virtual que contém a imagem do processo.
- .. acesso protegido a processadores e outros processos (p.ex., comunicação entre processos), arquivos e recursos de I/O, etc..
- “**lembrete**” .. uma maneira de ver uma “thread” é visualizá-la como um contador de programa independente operando dentro de um processo.



Multiple Process
Multiple Threads por Process

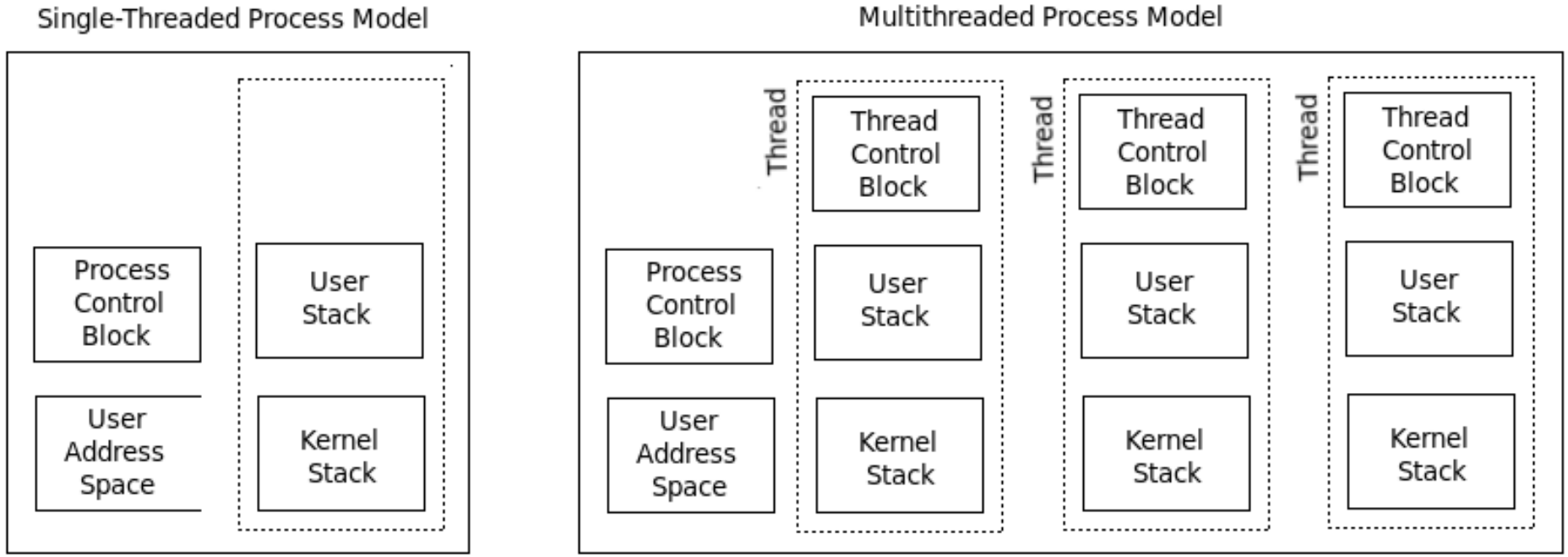
… 4.1.1 – Multithreading

- “**multithreaded environment**” .. em um processo, pode haver uma ou mais threads, cada “thead” com as seguintes características:
- (1) .. estado de execução da “thread” (“running”, “ready”, etc.).
- (2) .. contexto da “thread” salvo quando não estiver em execução.
- (3) .. pilha de execução da “thread”.
- (4) .. algum armazenamento estático por “thread” para variáveis locais.
- (5) .. acesso à memória e recursos do processo, compartilhado com todas as outras “threads” no mesmo processo.

- “**lembrete**” .. uma maneira de ver uma “thread” é visualizá-la como um contador de programa independente operando dentro de um processo.

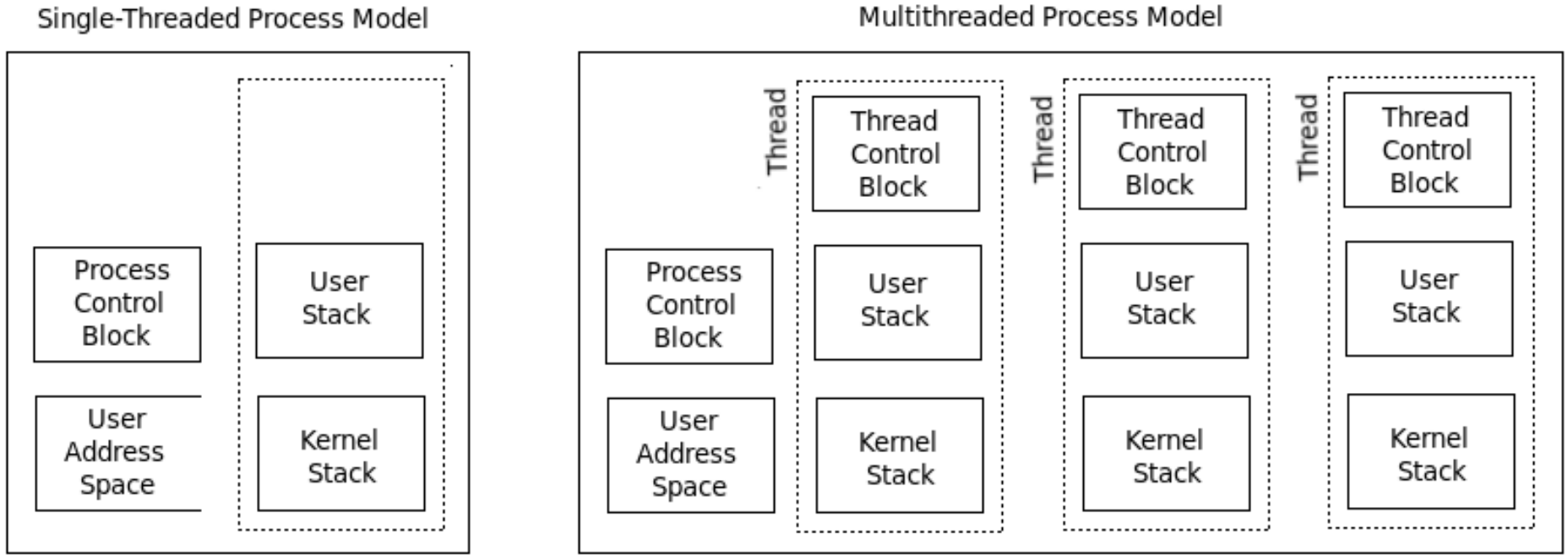
… 4.1.1 – Multithreading

- “**single-threaded model**” .. representação contempla o “process control block”, espaço de endereço do usuário, pilhas de usuário e kernel para gerenciar o comportamento de chamada / retorno.
- .. enquanto o processo está em execução, ele controla os registros do processador, que são salvos quando o processo não está em execução.



… 4.1.1 – Multithreading

- “**multithreaded model**” .. há um único “process control block” e endereço do usuário associado ao processo, mas agora há pilhas separadas para cada “thread”, bem como um controle separado.
- .. pilhas de cada “thread” contém valores de registro, prioridade e outros informações de estado relacionadas a “thread”.



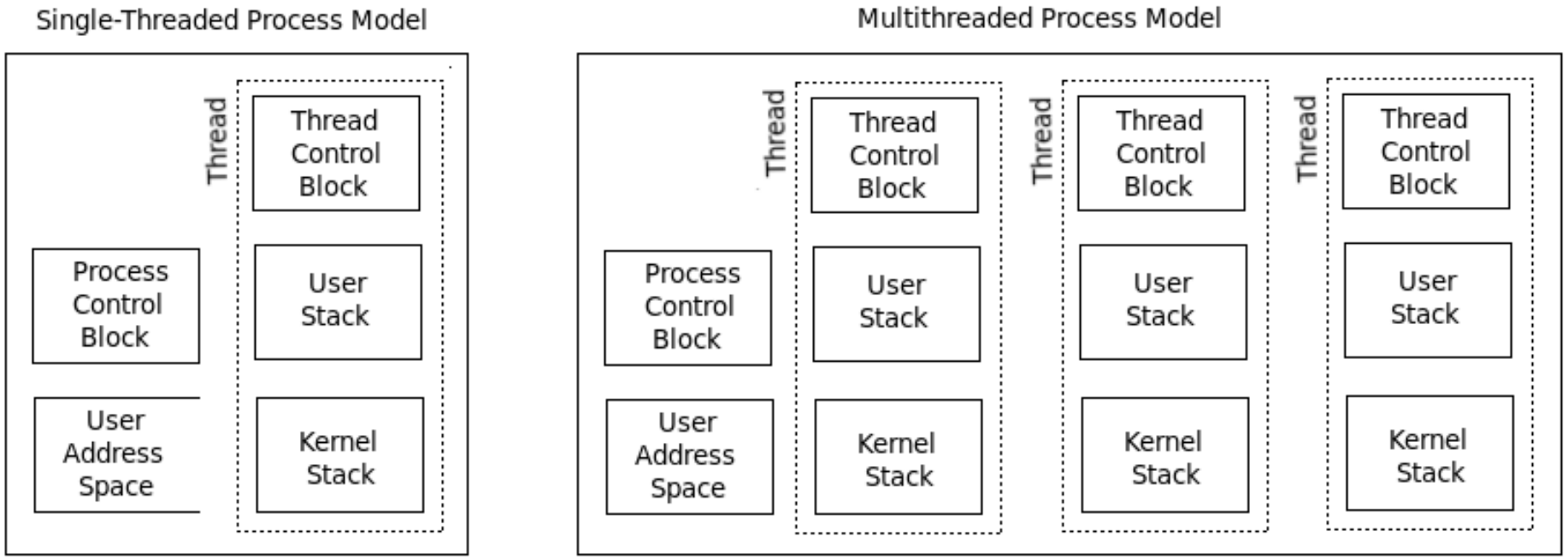
… 4.1.1 – Multithreading

- e.g., considere um “file server” que utiliza-se de “threads”, assim, a medida que um novo pedido chega, um nova “thread” pode ser gerada para o programa de gerenciamento de arquivos.
- .. como um servidor manipula muitas solicitações, muitos encadea-mentos são criados e destruídos em um curto período.
- .. se o “file server” for executado em um sistema multiprocessado, várias “threads” do mesmo processo poderão ser executadas simultaneamente em diferentes processadores.

- !? como processos ou “threads” em um servidor de arquivos podem compartilhar dados do arquivo e/ou coordenar suas ações !?
- “**vantagem**” .. é mais rápido usar “threads” e memória compartilhada do que processos e passagem de mensagens para essa coordenação.

4.1.2 – Thread Functionality

- “**thread states**” .. assim como os processos, as “threads” tem estado de execução e podem ser sincronizadas com outras “threads”.
- “**thread states**” .. principais estados são “running”, “ready” e “blocked”.
- .. geralmente, não faz sentido associar estado de “suspended” a uma “threads” porque tais estados são conceitos do nível de processo.



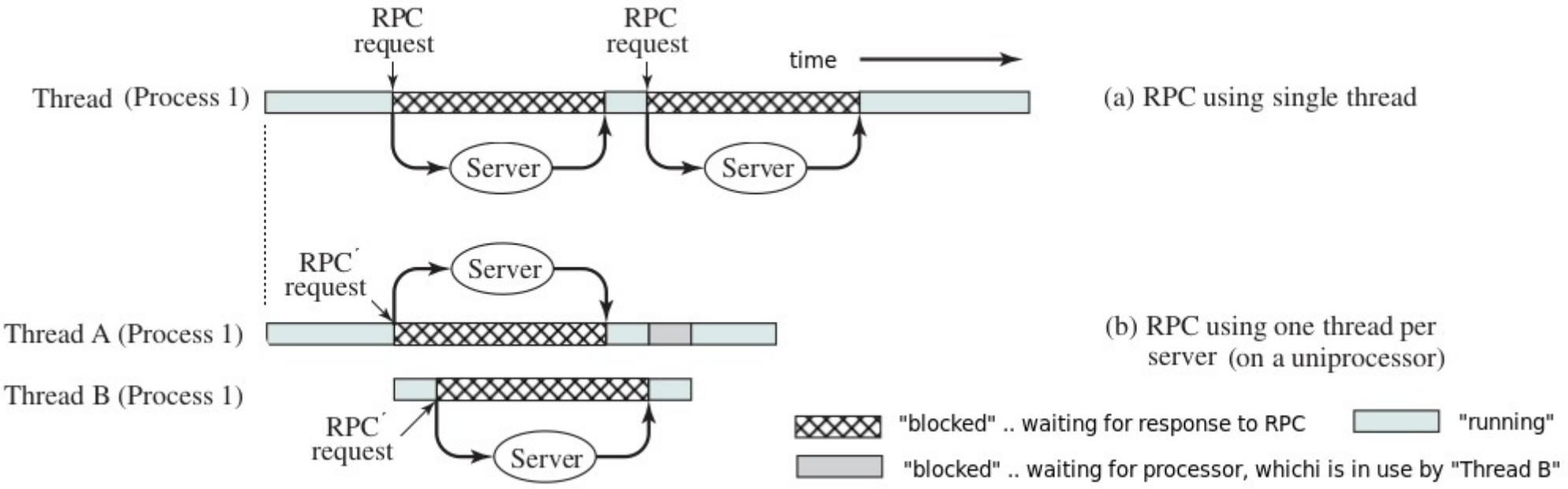
… 4.1.2 – Thread Functionality

- .. 04 operações básicas de “thread” associadas a mudança de estado:

- “**spawn**” .. uma “thread” (dentro de um processo) pode gerar outra “thread” dentro do mesmo processo, fornecendo um ponteiro de instrução e argumentos para a nova “thread”.
- “**blocked**” .. quando uma “thread” precisa esperar por um evento, a “thread” é bloqueada e, na sequência, o processador pode executar uma outra “thread” no estado de “ready” no mesmo ou outro processo.
- “**unblock**” .. “thread” muda para o estado ou fila “ready”, quando ocorre o evento para o qual ums “thread” se encontra bloqueada.
- “**finish**” .. quando uma “thread” é concluída, seu contexto de registro e pilhas são desalocados.

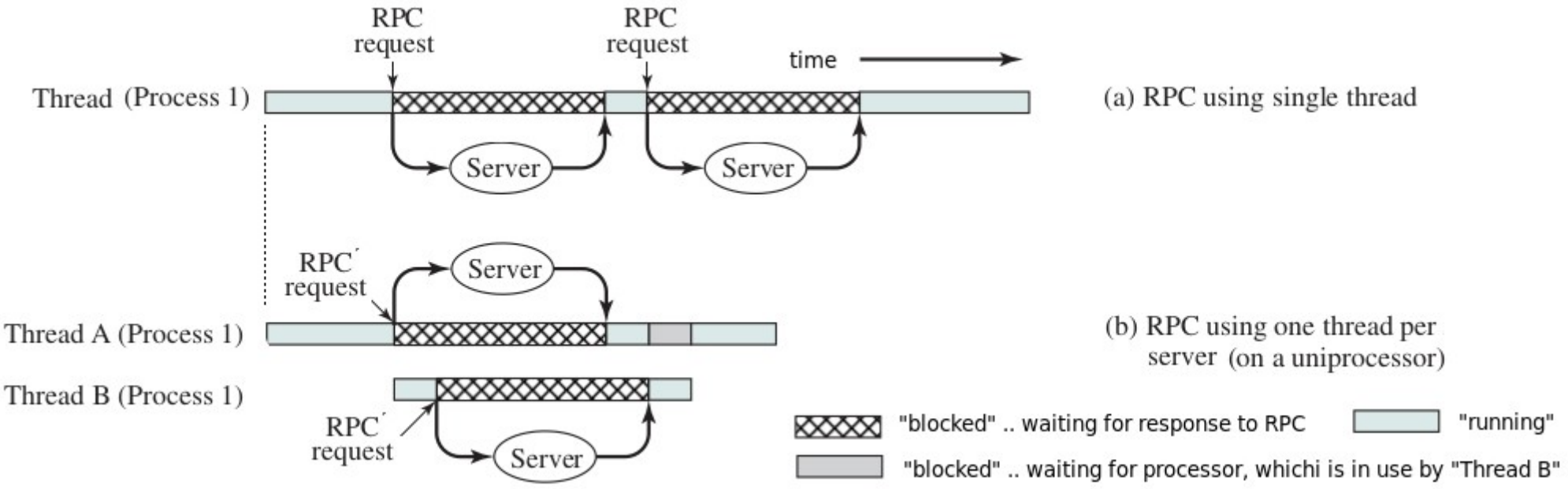
… 4.1.2 – Thread Functionality

- e.g., considere um processo no qual 02 chamadas RPC são executadas para 02 “hosts” diferentes.
- .. em um programa “single-threaded” e monoprocessado, os resultados são obtidos em sequência, de modo que o processo deve aguardar a resposta do 1ª invocação para dar prosseguimento a 2ª invocação.



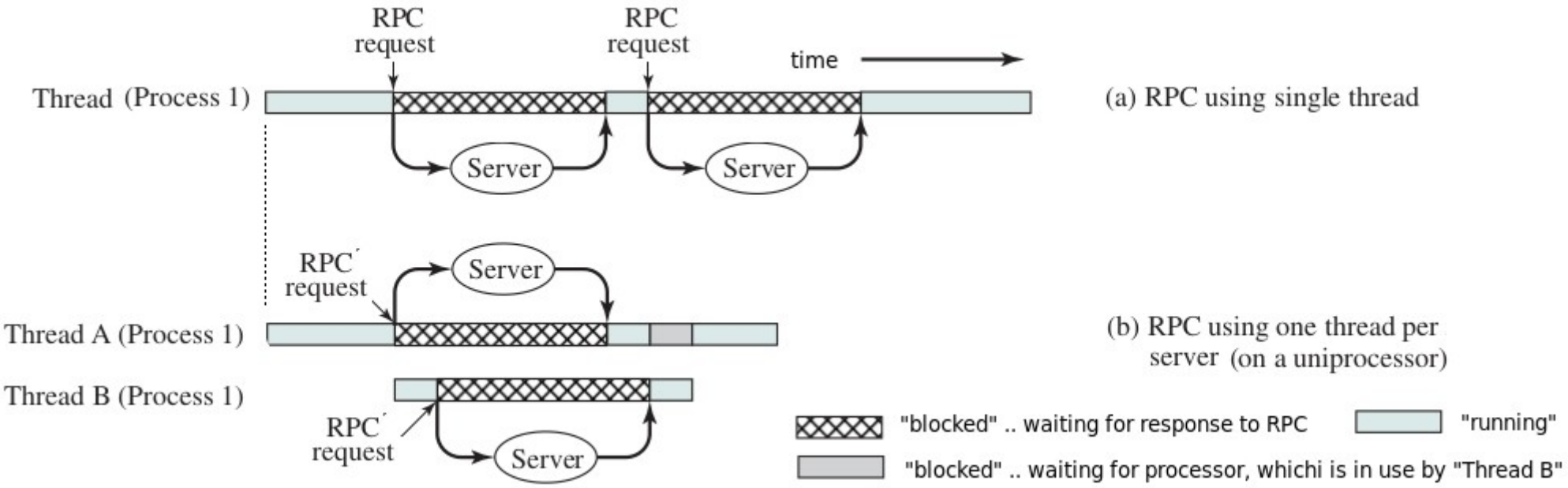
… 4.1.2 – Thread Functionality

- e.g., considere um processo no qual 02 chamadas RPC são executadas para 02 “hosts” diferentes.
- .. observe que se este processo opera em um sist. monoprocessado, as solicitações ou invocações devem ser geradas sequencialmente e os resultados processados em sequência.



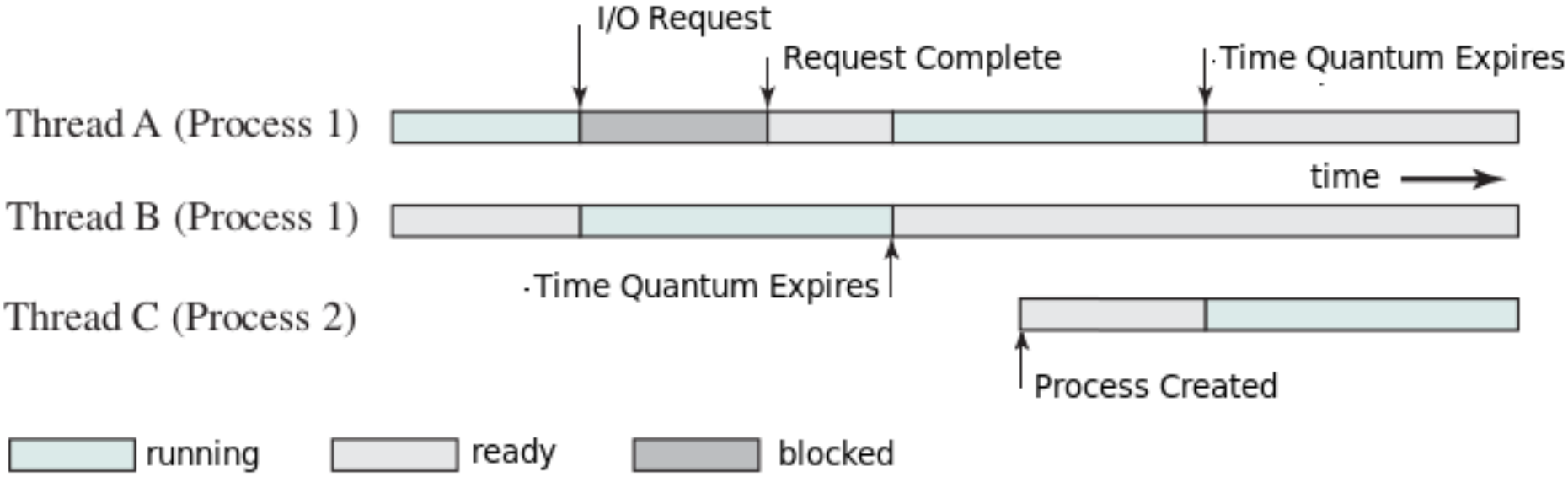
… 4.1.2 – Thread Functionality

- .. reescrever o programa para usar uma “thread” em separado para cada invocação RPC resulta em uma aceleração substancial.
- .. 02 “threads” dentro do mesmo processo passam a esperar simultaneamente pelas 02 respostas decorridas das 02 requisições.



… 4.1.2 – Thread Functionality

- "**multiprogramming**" .. em um sistema monoprocessado, a multiprogramação permite a intercalação de várias "threads" em vários processos como no exemplo (03 "threads" em 02 processos).
- .. execução passa de um encadeamento para outro quando o encadeamento atualmente em execução é bloqueado ou quando sua fatia de tempo se esgota.



… 4.1.2 – Thread Functionality

- “**thread synchronization**” .. todos os encadeamentos e/ou entrelaçamentos de um processo compartilham o mesmo espaço de endereço e outros recursos, como arquivos abertos.
- .. qualquer alteração de um recurso por um encadeamento afeta o ambiente dos outros encadeamentos no mesmo processo.

- “**conclusão**” .. faz-se necessário sincronizar as atividades dos vários encadeamentos para que não interfiram entre si ou corrompam as estruturas de dados.

4.2 – Types of Threads

- "**User-Level and Kernel-Level Threads**" .. existem 02 categorias amplas de implementação de thread, ou seja, User-Level Threads (ULTs) e Kernel-Level Threads (KLTs).

- **"User-Level Threads**" .. quando somente threads no nível do usuário estão presentes, todo o seu gerenciamento é feito pela aplicação e, assim, o kernel não toma conhecimento da existência das threads.
- .. qualquer aplicação pode ser programada para acomodar múltiplas "threads", para tanto, basta utilizar uma biblioteca de "threads".
- .. qualquer aplicação pode ser programada para ser "multithread" ao usar uma biblioteca de threads (roinas para gerenciamento ULT).
- "**thread library**" .. contém código para criar e destruir threads, para passar mensagens e dados entre threads, para agendar a execução de threads e para salvar e restaurar contextos de threads.

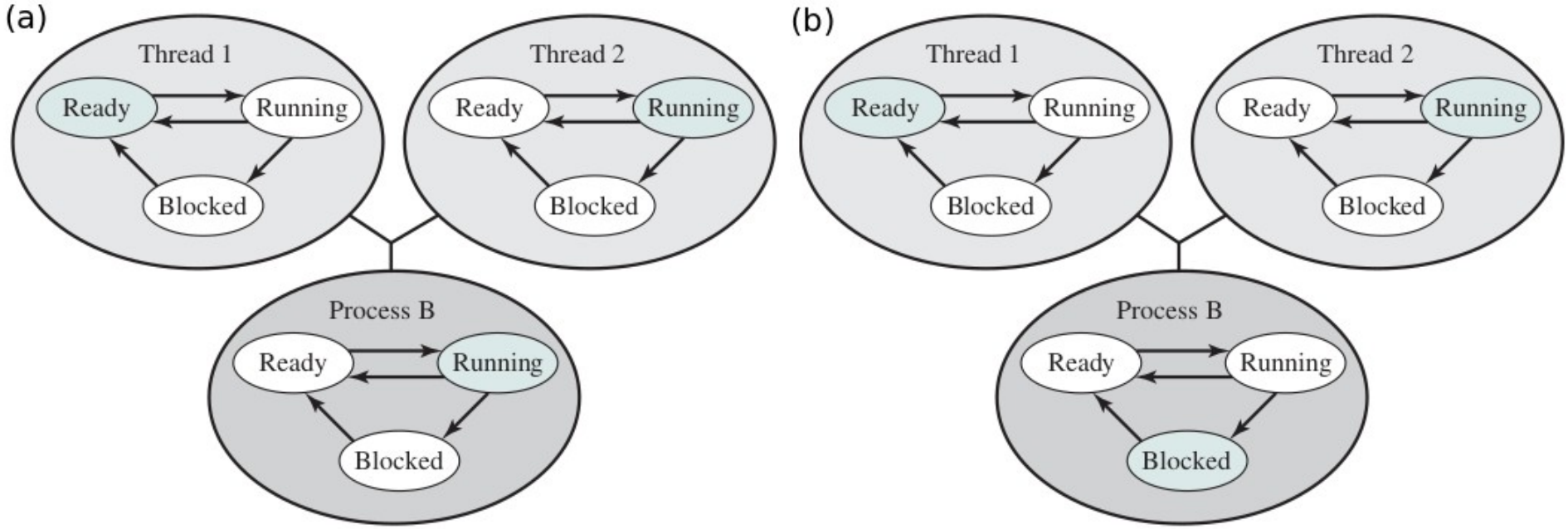
… 4.2 – Types of Threads

- … toda a atividade descrita no parágrafo anterior ocorre no espaço de endereçamento do usuário e dentro de um único processo, ou seja, “kernel” não tem conhecimento desta atividade.
- .. “kernel” escalona o processo como uma unidade e atribui um único estado de execução, p.ex., “ready”, “running”, ”blocked”, etc. para esse processo sem sequer tomar conhecimento das “threads”.
- .. exemplos a seguir de diagramas esclarecem a relação entre o escalonamento da “thread” e o escalonamento do processo.

- e.g., neste sentido, suponha que o Processo B encontra-se no estado de “running” com 02 (duas) “threads”, assim, seja a ilustração dos estados do processo e das 02 “threads” que fazem parte do processo.

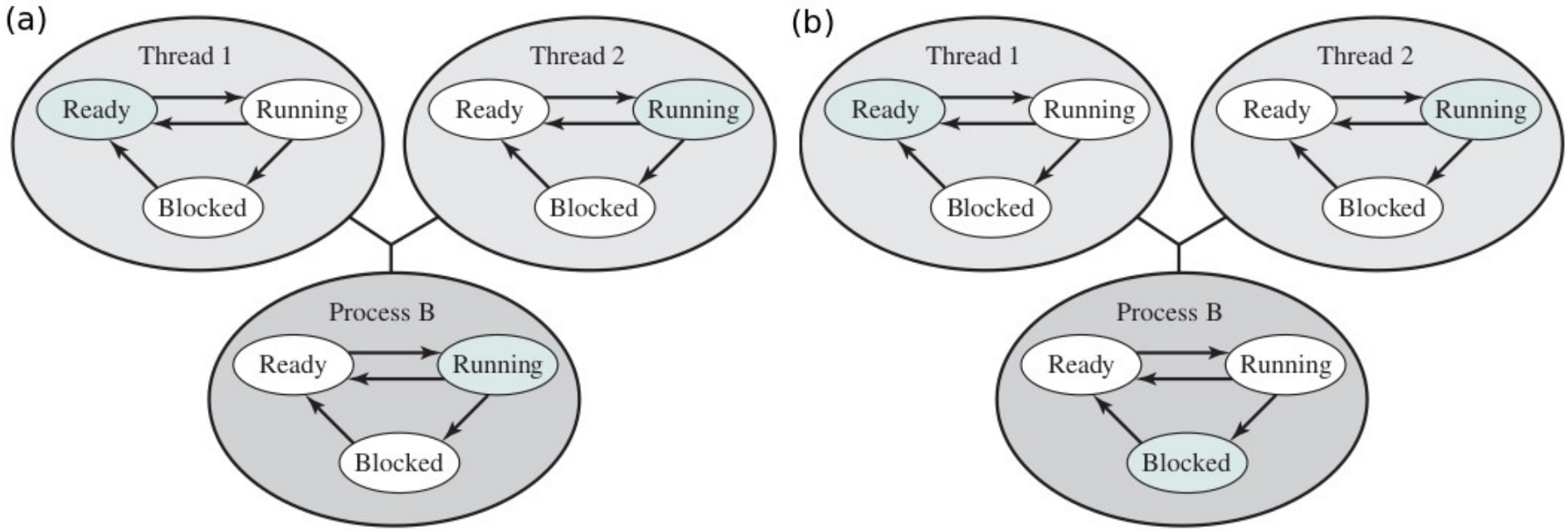
… 4.2 – Types of Threads

- e.g., .. suponha que o Processo B (P#B) encontra-se no estado de “running” com 02 (duas) “threads”, assim, seja a ilustração dos estados do processo e das 02 (duas) “threads” que fazem parte do processo.
- (a) .. aplicativo em execução em T#2 faz uma chamada de sistema que bloqueia P#B, p.ex., uma chamada de I/O » faz com que o controle seja transferido para o “kernel”.



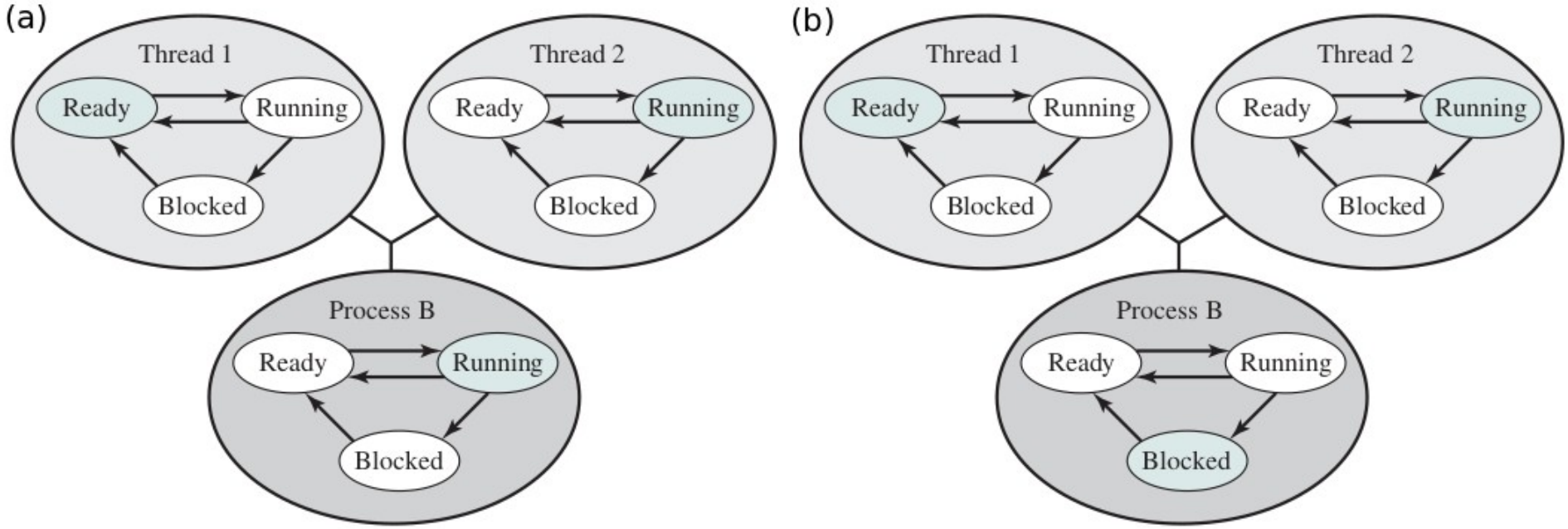
… 4.2 – Types of Threads

- (a) .. kernel invoca a ação de I/O, coloca o processo P#B no estado “blocked” e muda para outro processo, ou seja, (a) » (b).
- .. enquanto isso, de acordo com a estrutura de dados mantida pela biblioteca de “threads”, a T#2 do P#B ainda está no estado “running”.



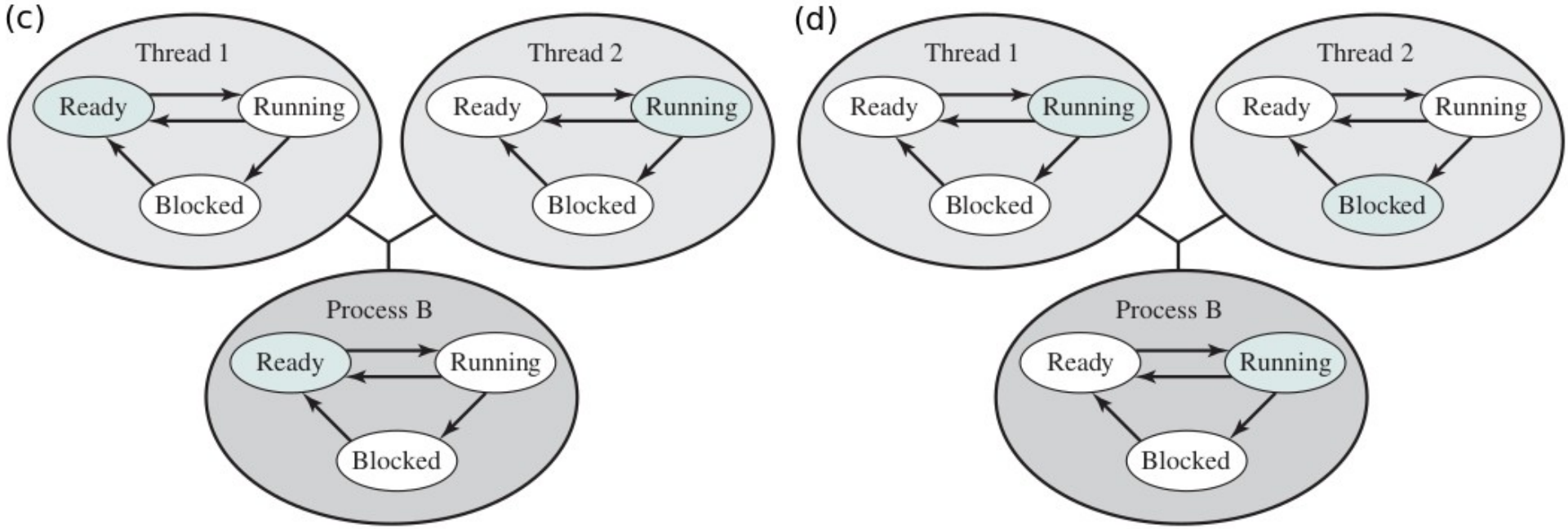
… 4.2 – Types of Threads

- (b) .. observe que T#2 não está realmente sendo executada no sentido de ser executada pelo processador, mas percebe-se como se estivesse no estado “running” pela biblioteca de “threads”.
- .. “kernel” escalona o processo como uma unidade e atribui um único estado de execução, p.ex., “ready”, “running”, ”blocked”, etc. para esse processo sem sequer tomar conhecimento das “threads”.



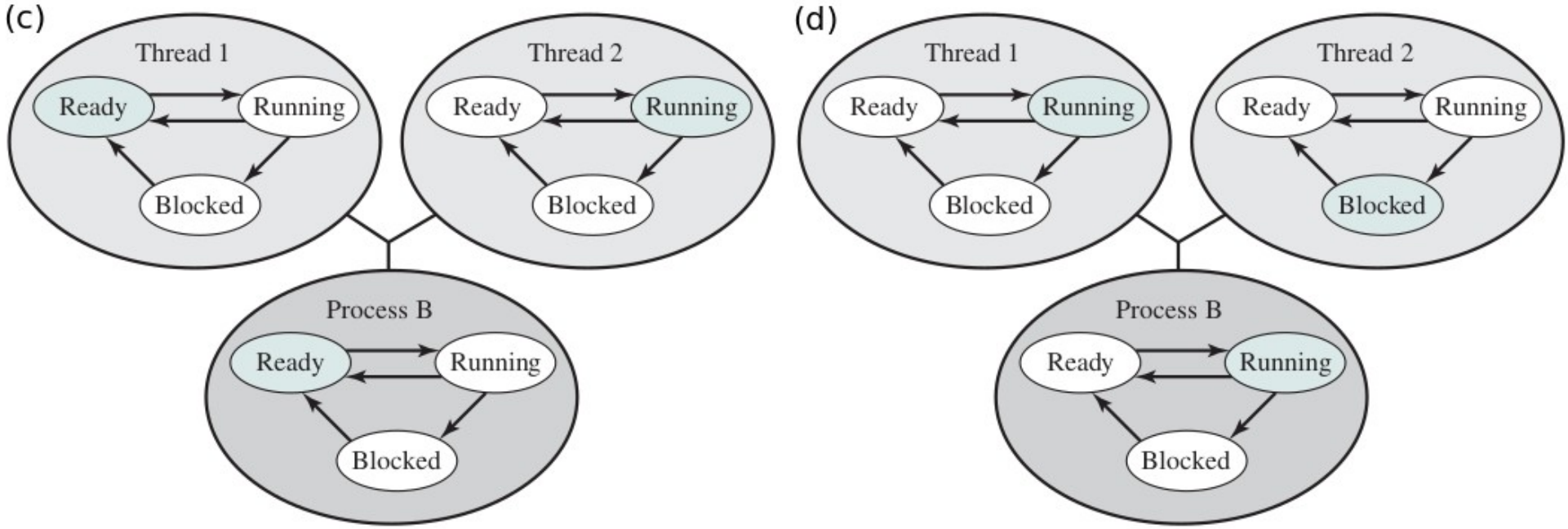
… 4.2 – Types of Threads

- (c) .. uma interrupção de relógio passa o controle para o “kernel”, que determina que a fatia de tempo do P#B esgotou-se » kernel coloca o P#B no estado “ready” e muda para outro processo.
- .. enquanto isso, de acordo com a estrutura de dados mantida pela biblioteca de “threads”, T#2 do P#B ainda está no estado “running”.



… 4.2 – Types of Threads

- (d) .. T#2 atingiu um ponto em que precisa de alguma ação realizada pelo T#1 de P#B, assim, T#2 entra em um estado de “blocked” e o T#1 faz a transição de “ready” para “running”.
- .. o próprio processo permanece no estado “running”, já os diagramas de estado correspondentes são mostrados na Fig. (d).



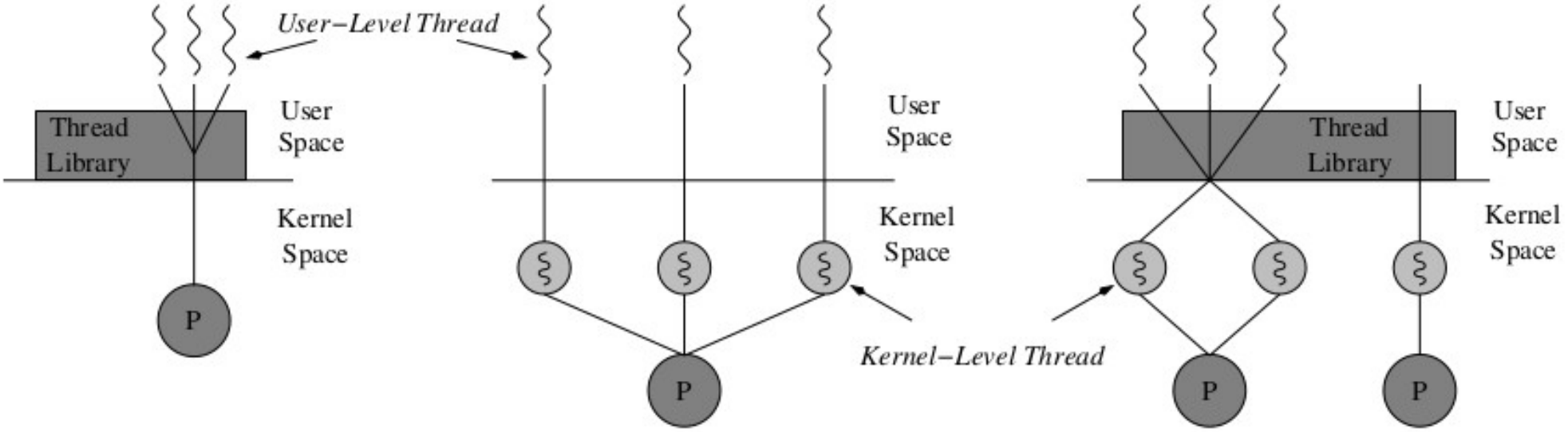
… 4.2 – Types of Threads

- “**algumas observações**“ .. acerca do cenário anterior ..

- .. nos casos da Figs. (b) e (c), quando o “kernel” muda o controle de volta para o P#B, a execução é retomada na “thread” T#2.
- .. observe que um processo pode ser interrompido, esgotando sua fatia de tempo ou interrompido por um processo de prioridade mais alta, enquanto executa o código na biblioteca de “threads”.
- .. deste modo, um processo pode se encontrar no meio de uma troca entre “threads” internas ao processo quando é interrompido.
- .. quando este processo é retomado, a execução continua dentro da biblioteca de threads, que conclui a troca de “thread” e transfere o controle para outra “thread” dentro desse processo.

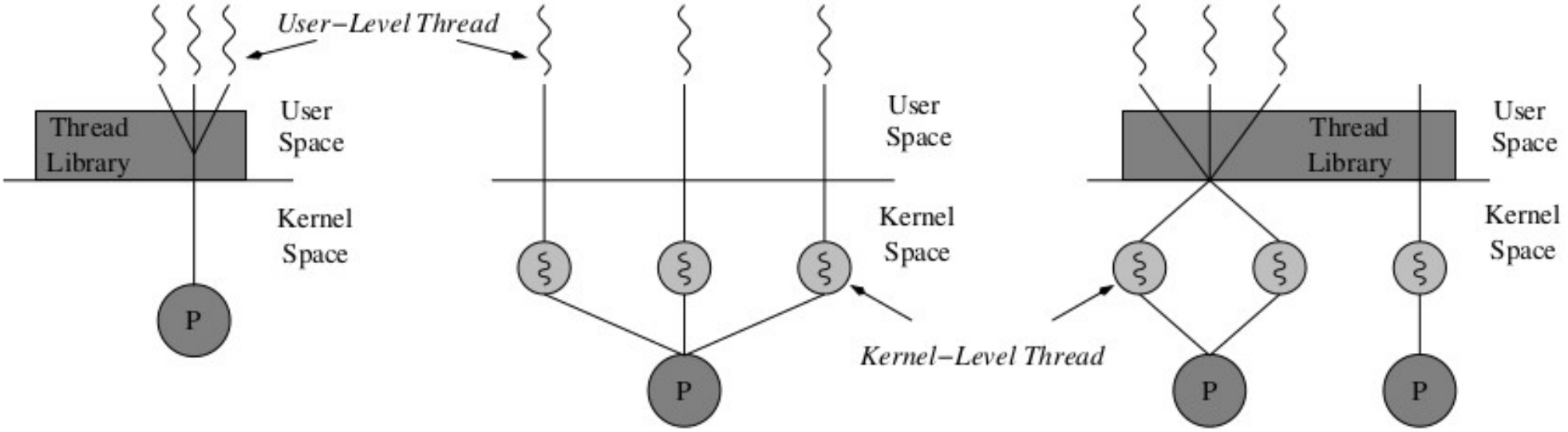
… 4.2 – Types of Threads

- "**vantagens**" .. ULTs sobre KLTs:
- (1) chaveamento de threads não requer privilégios do modo "kernel" pois todas as estruturas de dados para o gerenciamento das threads estão contidas no espaço de endereçamento de um único processo.
- … … (2) e (3)



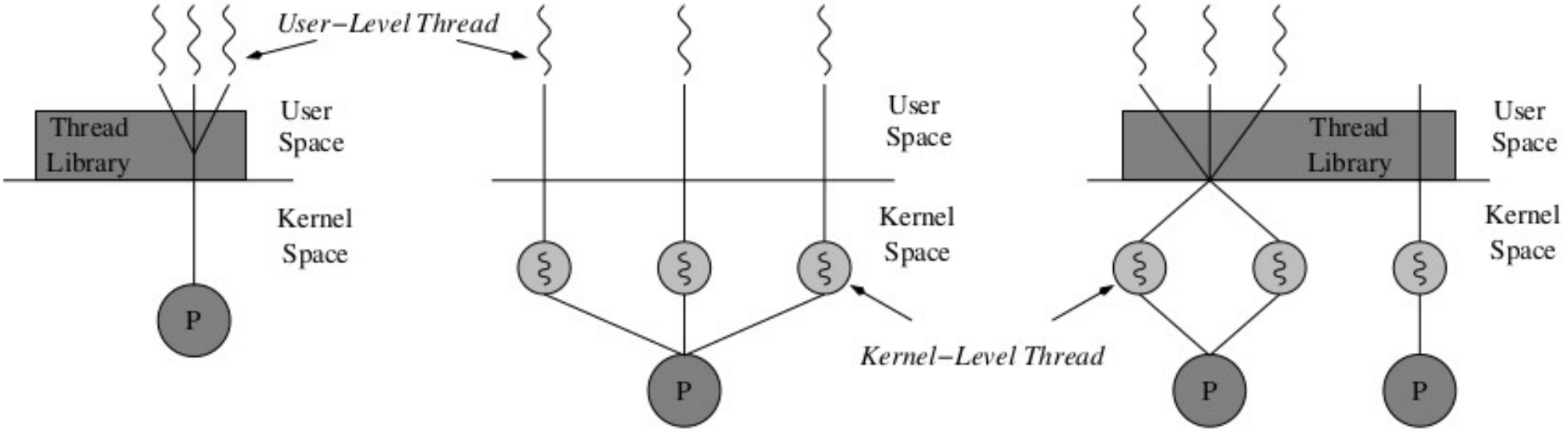
… 4.2 – Types of Threads

- “**vantagens**” .. ULTs sobre KLTs:
- (1) … …
- (2) escalonamento dependente da aplicação, o que possibilita variá-lo conforme as necessidades contempladas pela aplicação.
- (3) pode ser executada em qualquer sistema operacional, não exigindo nenhuma mudança no kernel de forma a suportar tal funcionalidade.



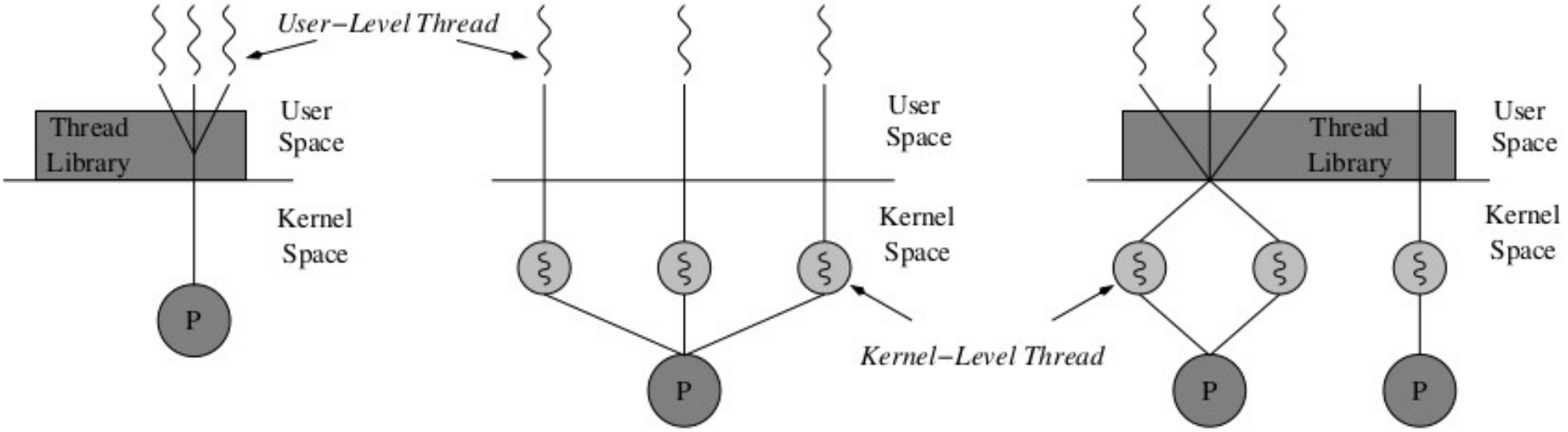
… 4.2 – Types of Threads

- "**desvantagens**" .. ULTs sobre KLTs:
- (1) na maioria dos sistemas operacionais, muitas das chamadas são bloqueantes o que leva ao bloqueio de todo o processo, não só da "thread" que executou a chamada.
- (2) aplicações "multithreaded" na abordagem de threads no nível do usuário não tiram vantagem de "sistemas multiprocessados".



… 4.2 – Types of Threads

- **“Kernel-Level Threads**” .. quando somente threads no nível do kernel estão presentes, nenhum código de gerenciamento está presente na aplicação, ou seja, encontramos somente um API de threads.
- nesta abordagem, uma aplicação com suporte a múltiplas threads deixa a cargo do sistema operacional toda a manutenção e gerenciamento das informações relacionadas às “threads” e aos processos.



… 4.2 – Types of Threads

- “**vantagens**” .. KLTs sobre ULTs :
- (1) “kernel” pode simultaneamente escalonar múltiplas threads para o mesmo processo sobre uma plataforma multiprocessada.
- (2) se uma “thread” em um processo é bloqueada, o kernel escalona uma outra thread para o mesmo processo, ou seja, o processo não é inteiramente bloqueado.
- (3) as próprias rotinas do “kernel” podem utilizar-se da abordagem “multithreaded” internamente.

- “**desvantagens**” .. KLTs sobre ULTs :
- (1) transferência do controle de uma “thread” para outra dentro do mesmo processo requer a mudança do modo de execução, ou seja, mudança “user-mode” para “kernel-mode”.

… 4.2 – Types of Threads

- Para ilustrar as diferenças, a Tab. mostra os resultados das medições feitas em um VAX com 01 processador executando “UNIX-like OS”.
- “**fork(...)**” .. quando se invoca a chamada de sistema “fork”, o sist. oper. cria uma cópia ou duplica o seu próprio processo que tem seu próprio espaço de endereçamento, dados e stack.
- .. isto permite que várias tarefas sejam executadas independentemen-te uma da outra, como se cada uma delas tivesse a memória completa da máquina.

Operation	User-Level Threads	Kernel-Level Threads	Processes
Null Fork	34	948	11,300
Signal Wait	37	441	1,840

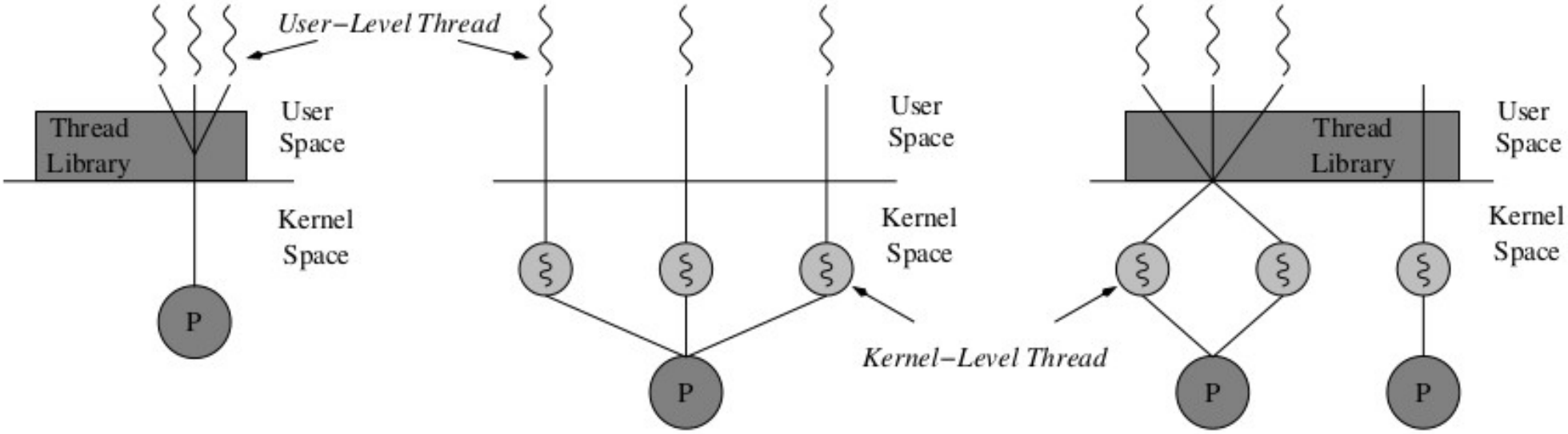
… 4.2 – Types of Threads

- Para ilustrar as diferenças, a Tab. mostra os resultados das medições feitas em um VAX com 01 processador executando “UNIX-like OS”.
- “**null fork**” .. tempo para criar, agendar, executar e concluir um processo ou thread que invoca o procedimento nulo (ou seja, a sobrecarga de bifurcação de um processo ou thread).
- “**signal-wait**” .. tempo para um processo / thread sinalizar um processo ou thread em espera e, em seguida, aguardar uma condição (ou seja, a sobrecarga de sincronizar dois processos ou threads juntos).
- .. que há uma ordem de magnitude ou mais de diferença entre ULTs e KLTs e de forma semelhante entre KLTs e processos.

Operation	User-Level Threads	Kernel-Level Threads	Processes
Null Fork	34	948	11,300
Signal Wait	37	441	1,840

… 4.2 – Types of Threads

- "**combined approaches**" .. criação de "threads" se dá no espaço do usuário, assim como a maior parte do agendamento e sincronização de "threads" em um aplicativo.
- .. vários ULTs de um único aplicativo são mapeados em algum nro. menor ou igual de KLTs, embora o programador posa ajustar o nro. de KLTs para um determinado aplicativo e processador.



… 4.2 – Types of Threads

- "**other arrangements**" .. recentemente, tem havido interesse em fornecer várias "threads" dentro de um único processo, ou seja, relacionamento "many-to-one".
- .. no entanto, outras combinações também são investigadas, a saber, um relacionamento "many-to-many" e "one-to-many".

Threads: Processes	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:N	Combines attributes of M:1 and 1:M cases.	TRIX

4.3 – Multicore and Multithreading

- “**multicore system**” .. oferece suporte para aplicativos com várias “threads”, como em console de videogame ou “desktop” pessoal.
- .. incialmente são discutidos algumas das implicações de desempenho de um aplicativo “multithread” em um sistema “multicore”.
- .. na sequência, descreve-se um aplicativo projetado para explorar recursos de vários núcleos – “multicore system”

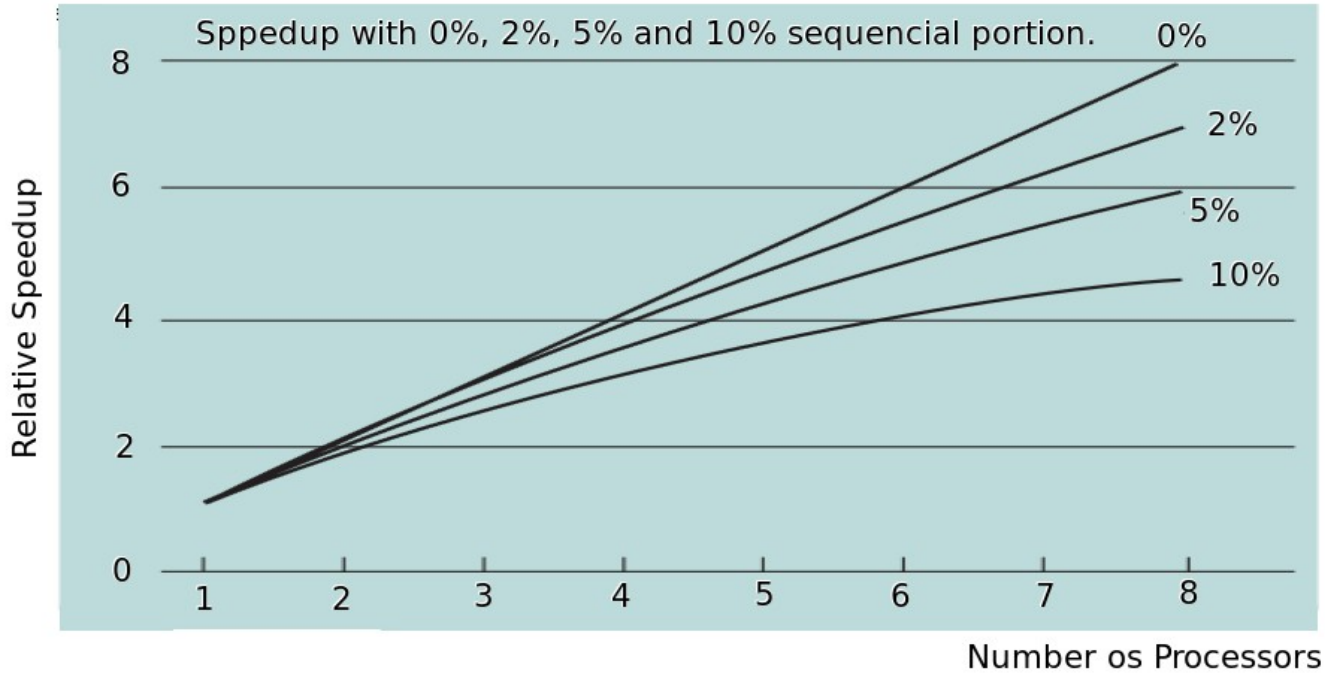
… 4.3.1 – Performance of Software on Multicore

- "**potencial benefits**" .. depende da capacidade de explorar efetiva-mente os recursos paralelos disponíveis para o aplicativo.
- e.g., seja o cenário de um único aplicativo em um sistema "multicore" e, no qual, possa-se aplicar a Lei de Amdahl.
- "**speedup**" = "time to execute program on a single processor" / "time to execute program on N parallel processors"
- "**speedup**" = 1 / [(1 - f) + f/N]

- onde:
- (1 - f) = tempo de execução de código inerentemente serial.
- f = fração de código paralelizável sem sobrecarga de programação.

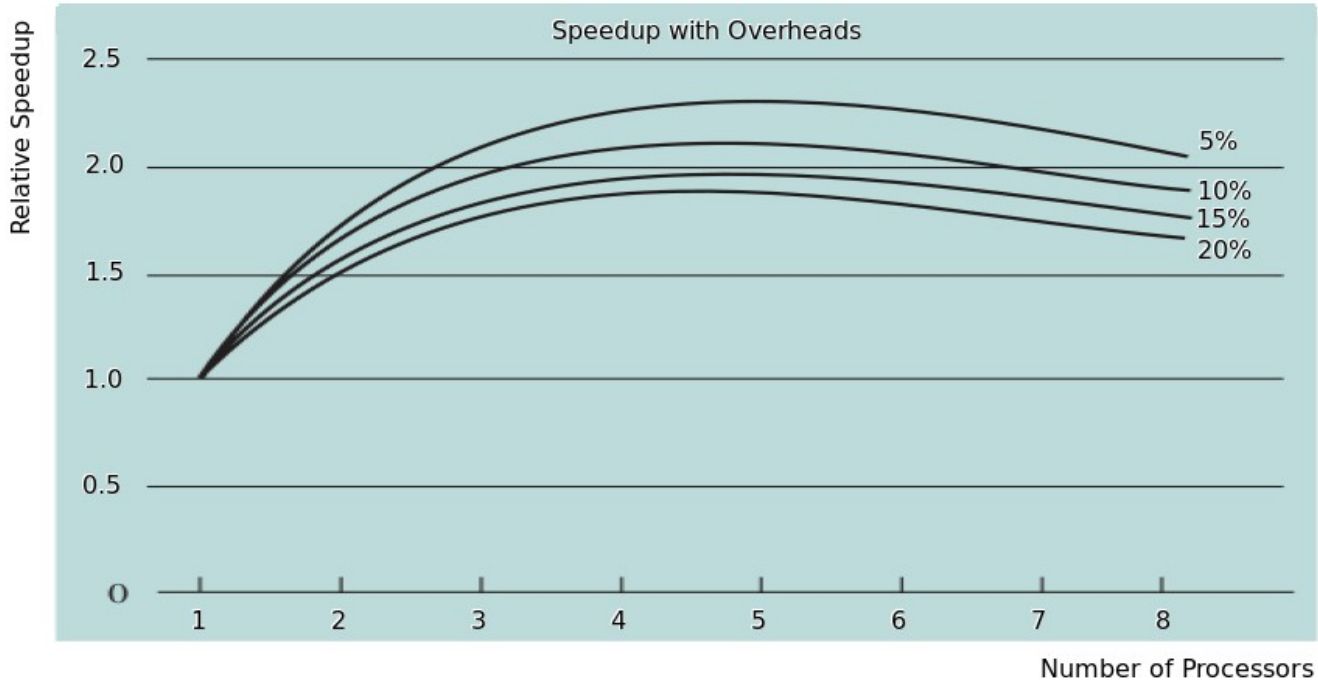
… 4.3.1 – Performance of Software on Multicore

- "**speedup with sequential portion**" .. pequeno percentual de código serial, p.ex., 2%, 5% ou 10% há um impacto perceptível.
- .. se apenas 10% do código for inerentemente serial (f = 0,9), executar o programa em um sistema "multicore" com 08 processadores resulta em um ganho de desempenho de apenas um fator de 4,7.



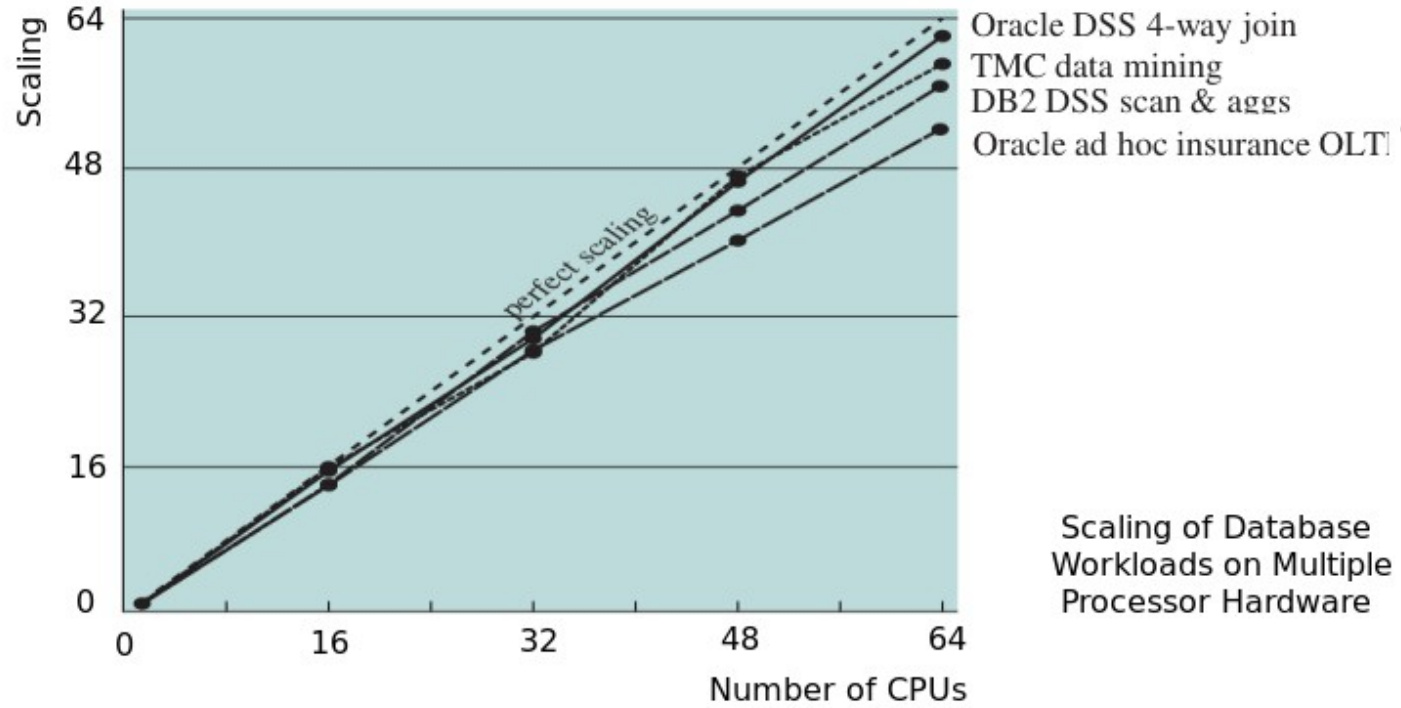
… 4.3.1 – Performance of Software on Multicore

- "**speedup with overhead**" .. software incorre em sobrecarga como resultado da comunicação e distribuição de trabalho para vários processadores bem como pela sobrecarga de coerência da "cache".
- "**conclusão**" .. performance aumenta até um certo nro. de processadores, a partir deste ponto ou pico, a sobrecarga pelo uso de mais processadores degrada a performance.



… 4.3.1 – Performance of Software on Multicore

- "**database application**" .. é possível explorar efetivamente um sistema "multicore" especialmente em aplicações de bancos de dados.
- .. muitos tipos de servidores podem usar com eficácia a organização paralela de vários núcleos, uma vez que lidam com várias transações relativamente independentes em paralelo.



Scaling of Database Workloads on Multiple Processor Hardware

… 4.3.1 – Performance of Software on Multicore

- "**classes of applications**" .. além do software de servidor de uso geral, várias classes de aplicativos se beneficiam diretamente da capacidade de dimensionar a taxa de transferência com o número de núcleos.

- "**multithreaded native applications**" .. são caracterizados por terem um pequeno número de processos altamente segmentados.
- e.g., exemplos de aplicativos encadeados incluem Lotus Domino ou Siebel CRM (Customer Relationship Manager).
- "**multiprocess applications**" .. aplicativos multiprocessos são caracterizados pela presença de muitos processos de thread único.
- e.g., exemplos de aplicativos multiprocessos incluem o banco de dados Oracle, SAP e PeopleSoft.

… 4.3.1 – Performance of Software on Multicore

- "**classes of applications**" .. além do software de servidor de uso geral, várias classes de aplicativos se beneficiam diretamente da capacidade de dimensionar a taxa de transferência com o número de núcleos.

- "**java applications**" .. adota o "threading" de uma maneira fundamental e, não apenas na linguagem Java mas também na JVM que fornece agendamento e gerenciamento de memória para aplicativos Java.
- … dentre os aplicativos que se beneficiam diretamente de recursos de vários núcleos destaca-se o servidor de aplicativos Java da Sun, o Weblogic da BEA, o Websphere da IBM e o Tomcat de código aberto.
- .. todos os aplicativos que usam um servidor de aplicativos Java 2 Platform, Enterprise Edition (plataforma J2EE) podem se beneficiar imediatamente da tecnologia "multicore".

… 4.3.2 – Application Valve Game Software

- “**Valve Game Software**” .. empresa de entretenimento e tecnologia que desenvolveu uma série de jogos populares, bem como o “source engine”, um dos mais jogados e disponíveis no mercado.
- .. “**source**” é um mecanismo de animação usado para seus jogos e licenciado para outros desenvolvedores de jogos.

- “**source engine**” .. contempla “multithreading” para explorar o poder dos chips de processador multicore da Intel e AMD.
- .. código do motor de origem revisado fornece suporte mais poderoso para jogos Valve, como Half Life 2.

… 4.3.2 – Application Valve Game Software

- Há 03 opções de granularidade de “threads”:

- “**coarse threading**” .. módulos individuais, chamados sistemas, são atribuídos a processadores individuais.
- .. no caso do “source engine”, isso significa colocar renderização em um processador, IA em outro, física em outro e assim por diante.
- “**fine-grained threading**” .. muitas tarefas semelhantes ou idênticas são distribuídas por vários processadores.
- e.g., “loop” que itera sobre uma matriz de dados pode ser dividido em vários loops paralelos menores em “threads” individuais que podem ser agendados em paralelo.
- “**hybrid threading**” .. envolve o uso seletivo de “fine-grained thread” para alguns sistemas e “single threading” para outros sistemas.

… 4.3.2 – Application Valve Game Software

- “**alguns resultados**” .. por meio do “coarse threading” pode-se atingir até 02 vezes o desempenho em 02 processadores em comparação com a execução em um único processador.
- .. no entanto, esse ganho de desempenho só pode ser alcançado com gabinetes planejados.

- “**alguns resultados**” .. “hybrid threading” é mais promissor e escalona melhor, à medida que sistemas “multicore” com 8 ou 16 processadores se tornaram disponíveis.
- .. identifica-se sistemas que operam de forma muito eficaz, sendo atribuídos de forma permanente a um único processador.

4.4 – WINDOWS 7 Thread and SMP

- "**process design**" .. é impulsionado pela necessidade de fornecer suporte para uma variedade de ambientes de sistema operacional.
- .. os processos suportados por diferentes ambientes de sistema operacional diferem de várias maneiras, incluindo o seguinte:

- .. como os processos são nomeados;
- .. se os threads são fornecidos dentro dos processos;
- .. como os processos são representados;
- .. como os recursos do processo são protegidos;
- .. mecanismos para comunicação e sincronização entre processos;
- .. como os processos são relacionados entre si.

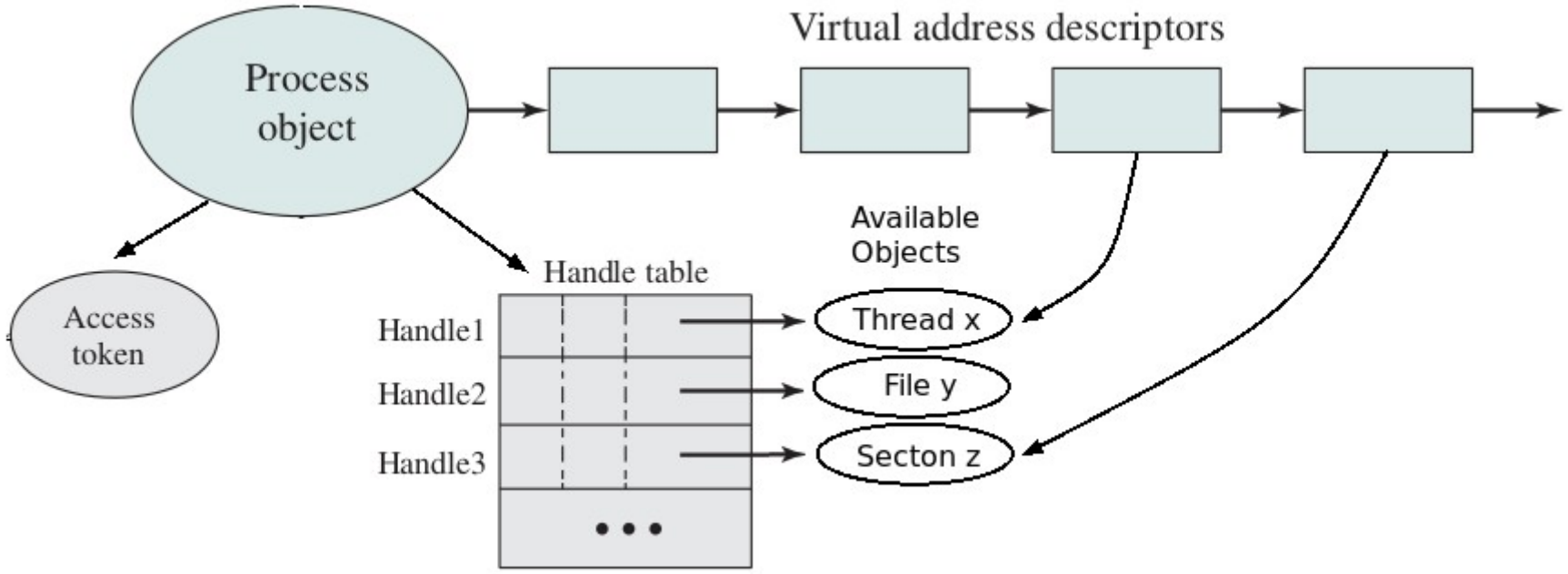
… 4.4 – WINDOWS 7 Thread and SMP

- "**Windows Kernel**" .. contempla estrutura de processos e serviços de modo relativamente simples e geral, permitindo que cada subsistema emule uma determinada estrutura e funcionalidade de processo.
- .. as características dos processos do Windows são as seguintes:

- 1) .. processos do Windows são implementados como objetos;
- 2) .. processo pode ser criado como um novo processo ou como uma cópia de um processo existente.
- 3) .. processo executável pode conter um ou mais threads.
- 4) .. processo e thread têm recursos de sincronização integrados.

… 4.4 – WINDOWS 7 Thread and SMP

- .. na relação entre processos e seus recursos, cada processo recebe um token de acesso de segurança, chamado de “primary token”.
- “**access token**” .. quando um usuário efetua logon, cria-se um token de acesso que inclui o ID de segurança para o usuário e cada processo criado ou executado em nome desse usuário possui uma cópia.

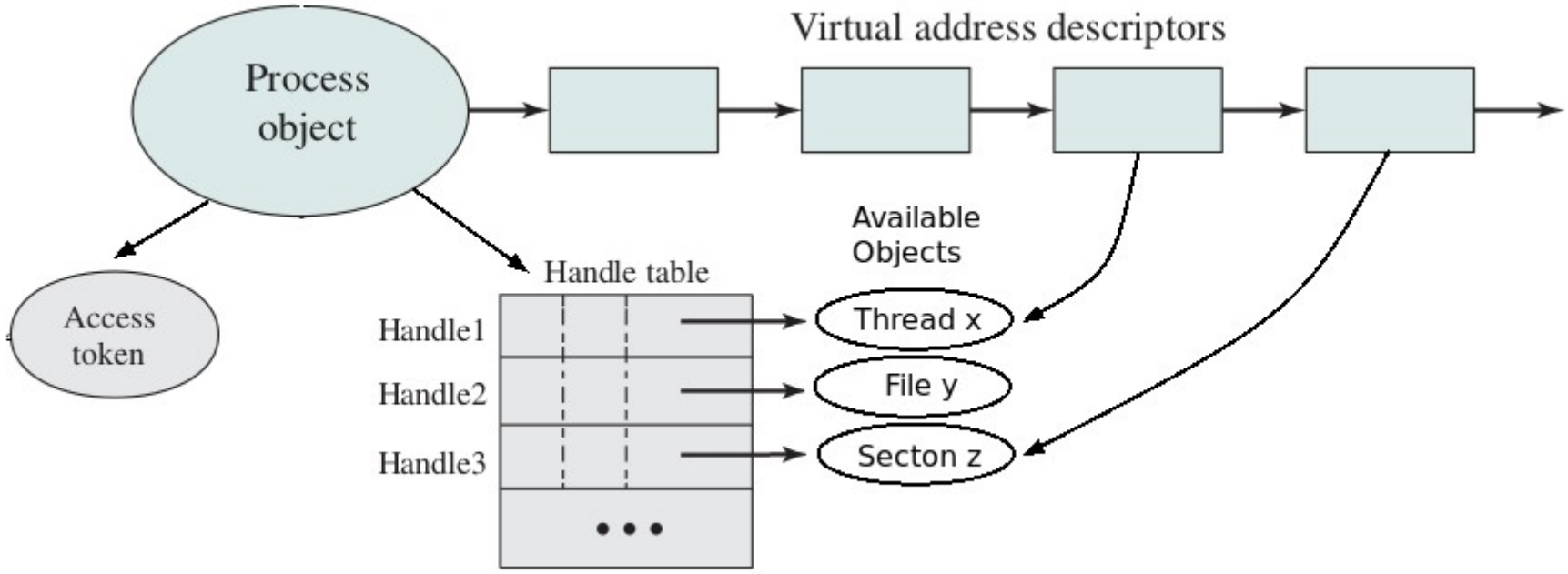


… 4.4 – WINDOWS 7 Thread and SMP

- “**access token**” .. quando um usuário efetua logon, cria-se um token de acesso que inclui o ID de segurança para o usuário e cada processo criado ou executado em nome desse usuário possui uma cópia.
- .. “access token” é utilizado para validar a capacidade do usuário de acessar objetos protegidos ou de executar funções restritas no sistema bem como acesso de objetos protegidos.
- .. “access token” também controla se o processo pode alterar seus próprios atributos, ou seja, o processo não tem um identificador aberto para seu token de acesso.
- .. caso o processo tente abrir esse identificador, o sistema de segurança determina se isso é permitido e, portanto, se o processo pode alterar seus próprios atributos.

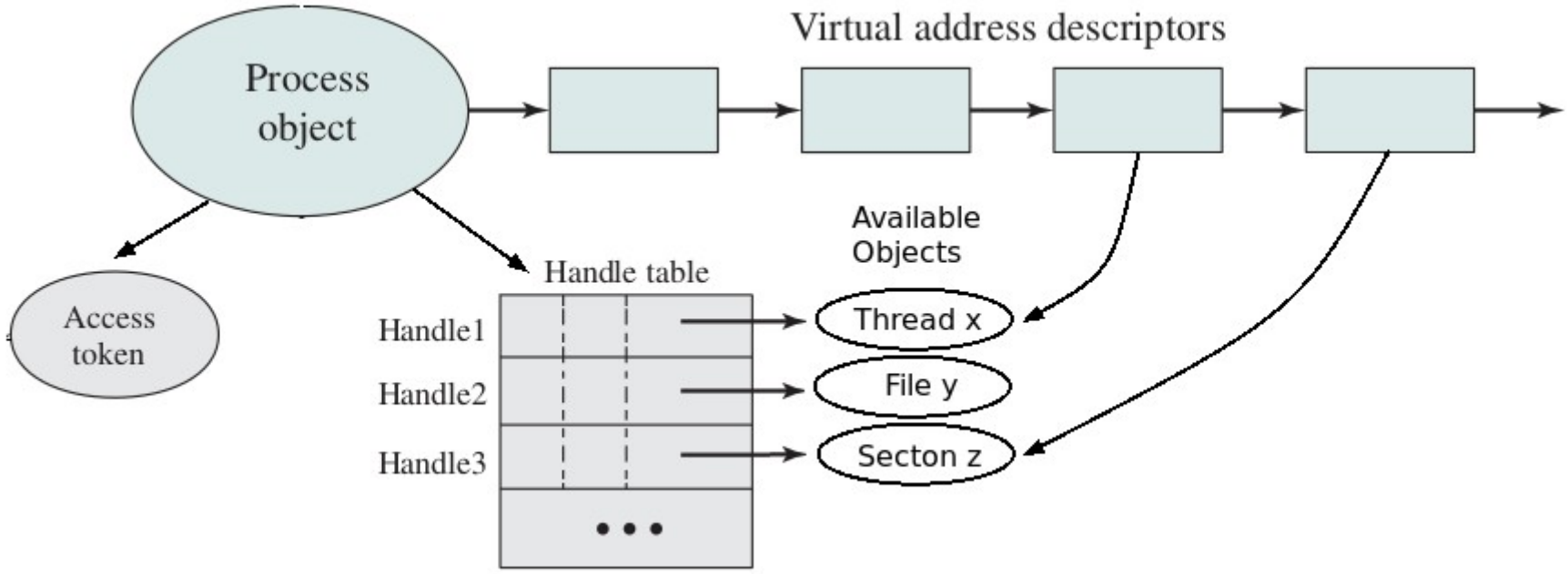
… 4.4 – WINDOWS 7 Thread and SMP

- .. há também uma série de blocos relacionados ao processo que definem o espaço de endereço virtual atualmente atribuído a ele.
- .. processo não pode modificar diretamente essas estruturas, mas conta com o gerenciador de memória virtual, que fornece um serviço de alocação de memória para o processo.



… 4.4 – WINDOWS 7 Thread and SMP

- .. finalmente, o processo inclui uma tabela de objetos, com alças para outros objetos conhecidos por este processo.
- .. a figura abaixo ilustra um processo com acesso a uma única “thread”, a um objeto de arquivo e a um objeto de seção que define uma seção de memória compartilhada.



4.4.1 – Processes and Thread Objects

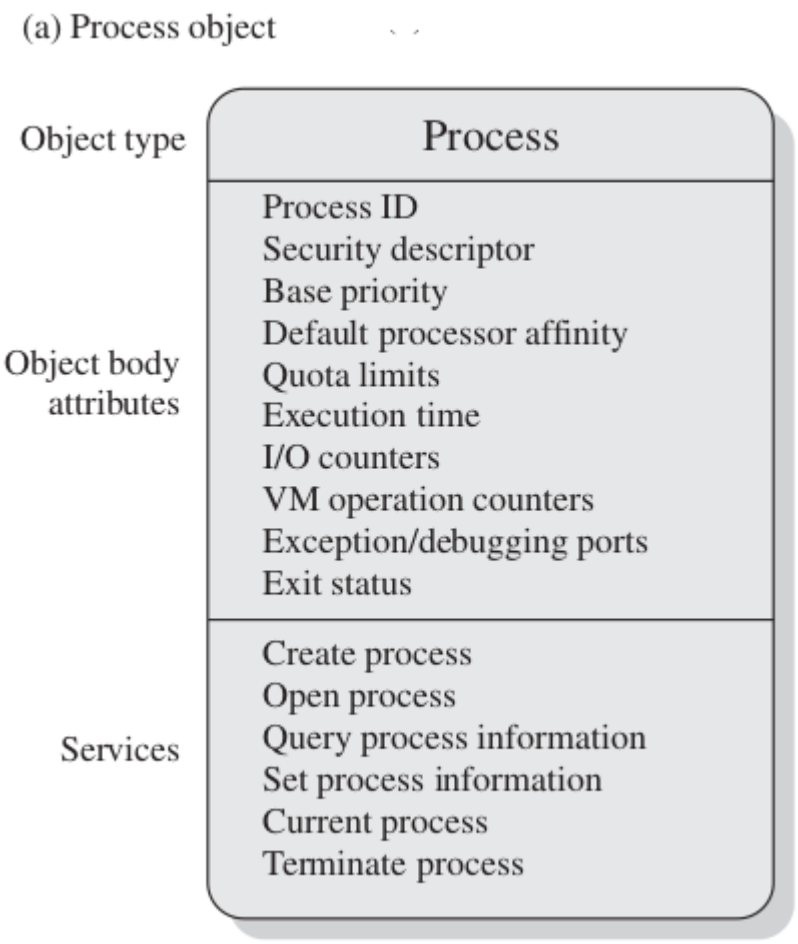
- “**object-oriented structure**” .. Windows usa 02 tipos de objetos relacionados a processos, ou seja, processos e threads.

- .. processo é uma entidade correspondente a um “job” ou aplicativo do usuário que possui recursos, como memória e arquivos abertos.
- .. “thread” é uma unidade escalonável que é executada sequencial-mente e passível de interrupção pelo sistema operacional.

… 4.4.1 – Processes and Thread Objects

- .. cada processo do Windows é representado por um objeto cuja estrutura geral acomoda alguns atributos.

- .. cada processo é definido por uma série de atributos e encapsula uma série de ações, ou serviços, que pode executar.
- .. um processo executa um serviço quando do chamado por meio de um conjunto de métodos de interface publicado.



… 4.4.1 – Processes and Thread Objects

- .. quando o Windows cria um novo processo, ele usa a classe de objeto, ou tipo, definido para o processo do Windows como um modelo para gerar uma nova instância de objeto.
- .. momento da criação, os valores dos atributos são inicializados, ou seja, atributos de objeto para um objeto processo.

- "**process id**" .. um valor exclusivo que identifica o processo para o sistema operacional, ou seja, um identificador único.
- "**security descriptor**" .. descreve quem criou um objeto, quem pode obter acesso ou usar o objeto e quem tem acesso negado ao objeto.

Process ID	A unique value that identifies the process to the operating system.
Security descriptor	Describes who created an object, who can gain access to or use the object, and who is denied access to the object.

… 4.4.1 – Processes and Thread Objects

- "**base priority**" .. define a prioridade de execução de linha de base para os "threads" do processo.
- "**default processor affinity**" .. conjunto padrão de processadores nos quais as "threads" do processo podem ser executadas.
- "**quota limits**" .. quantidade máxima de memória paginada e não paginada, espaço de arquivo de paginação e tempo de processador que os processos de um usuário podem usar.
- "**execution time**" .. quantidade total de tempo que todos as "threads" no processo foram executadas.

Base priority	A baseline execution priority for the process's threads.
Default processor affinity	The default set of processors on which the process's threads can run.
Quota limits	The maximum amount of paged and nonpaged system memory, paging file space, and processor time a user's processes can use.
Execution time	The total amount of time all threads in the process have executed.

… 4.4.1 – Processes and Thread Objects

- "**I/O counters**" .. variáveis que registram o número e tipo de operações de I/O que as "threads" do processo realizaram.
- "**virtual memory operation counters**" .. variáveis que registram o número e os tipos de operações de memória virtual que as "threads" do processo realizaram.

I/O counters	Variables that record the number and type of I/O operations that the process's threads have performed.
VM operation counters	Variables that record the number and types of virtual memory operations that the process's threads have performed.
Exception/debugging ports	Interprocess communication channels to which the process manager sends a message when one of the process's threads causes an exception.
Exit status	The reason for a process's termination.

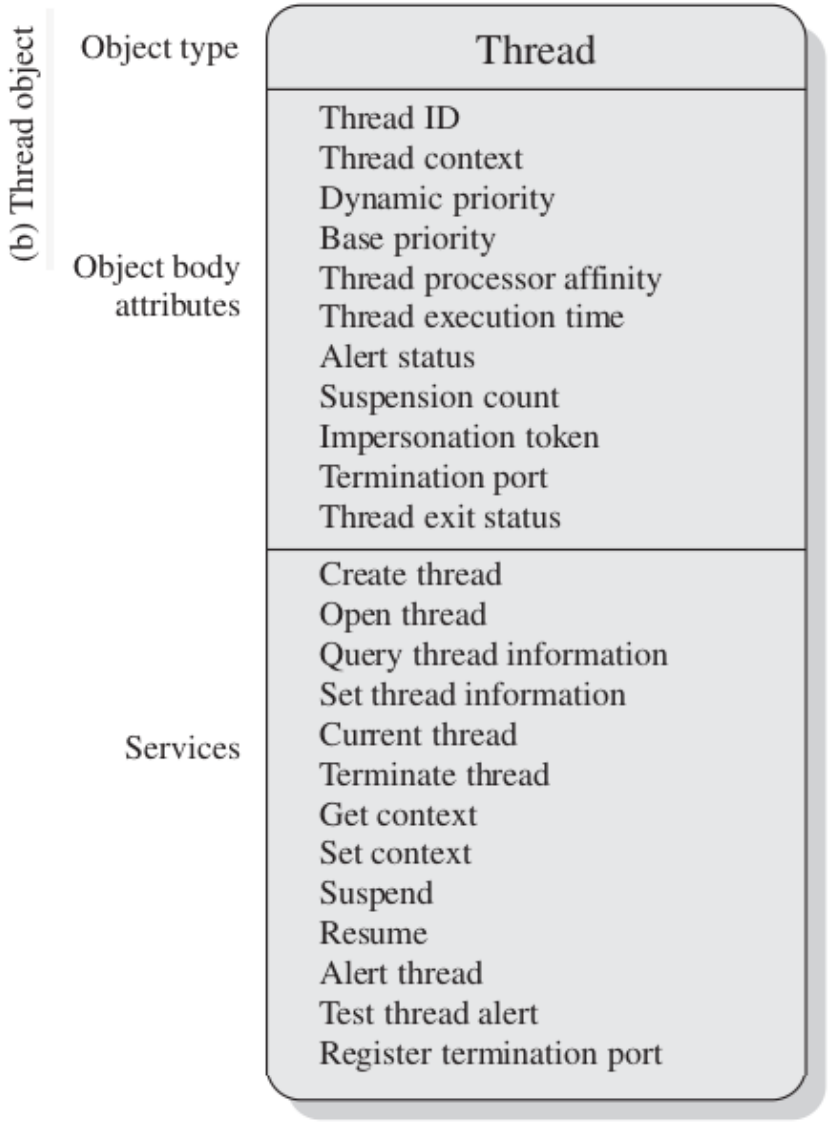
… 4.4.1 – Processes and Thread Objects

- "**exception/debugging ports**" .. canais de comunicação entre processos para os quais o gerenciador de processos envia uma mensagem quando uma das "threads" do processo causa uma exceção.
- "**exit status**" .. motivo do encerramento de um processo.

I/O counters	Variables that record the number and type of I/O operations that the process's threads have performed.
VM operation counters	Variables that record the number and types of virtual memory operations that the process's threads have performed.
Exception/debugging ports	Interprocess communication channels to which the process manager sends a message when one of the process's threads causes an exception.
Exit status	The reason for a process's termination.

… 4.4.1 – Processes and Thread Objects

- .. um processo do Windows deve conter pelo menos uma “thread” para ser executado, que por sua vez pode criar outras “threads”.
- .. em um sistema multiprocessado, vários threads do mesmo processo podem ser executados em paralelo.
- .. diagrama ao lado descreve a estrutura do objeto para um objeto thread e para os quais há atributos específicos.



… 4.4.1 – Processes and Thread Objects

- Como será visto, alguns dos atributos de “thread” se assemelham aos atributos de um processo, uma vez o valor do atributo da “thread” é derivado do valor do atributo do processo.
- .. p.ex., “thread processor afinity” refere-se aos processadores que uma “thread” pode executar, ou seja, este conjunto é mesmo ou parte do conjunto dos processadores para “default processor affinity”.

- “**threads attributes**” .. seja alguns atributos de “threads” ..

- “**thread id**” .. valor exclusivo que identifica uma thread.
- “**thread context**” .. conjunto de valores de registro e outros dados voláteis que definem o estado de execução de uma “thread”.

… 4.4.1 – Processes and Thread Objects

- “**dynamic priority**” .. prioridade de execução da “thread”.
- “**base priority**” .. limite inferior da prioridade dinâmica da “thread”.
- “**thread processor afinity**” .. conjunto de processadores nos quais o thread pode ser executado, que é um subconjunto ou toda a afinidade do processador do processo do thread.

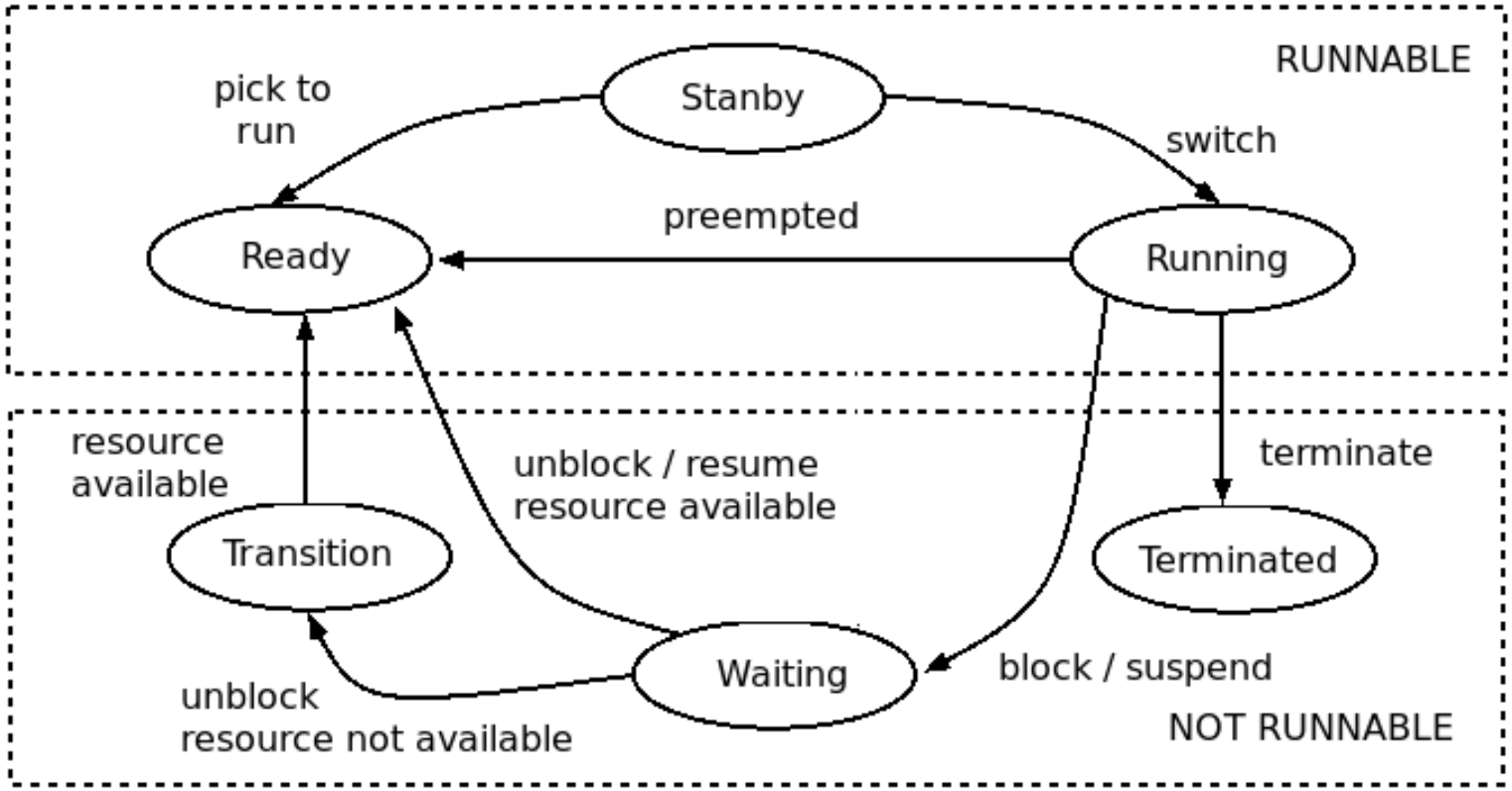
- “**thread execution time**” .. quantidade cumulativa de tempo que uma “thread” foi executada no modo usuário e no modo kernel.
- “**alert status**” .. um sinalizador que indica se uma “thread” em espera pode executar uma chamada de procedimento assíncrono.
- “**suspension count**” .. número de vezes que a execução da “thread” foi suspensa sem ser retomada.

… 4.4.1 – Processes and Thread Objects

- “**impersonation token**” .. um token de acesso temporário que permite a uma “thread” realizar operações em nome de outro processo.
- “**termination port**” .. canal de comunicação entre processos para o qual o gerenciador de processos envia uma mensagem quando a “thread” termina (usado por subsistemas).
- “**thread exti status**” .. motivo do encerramento de uma “thread”.

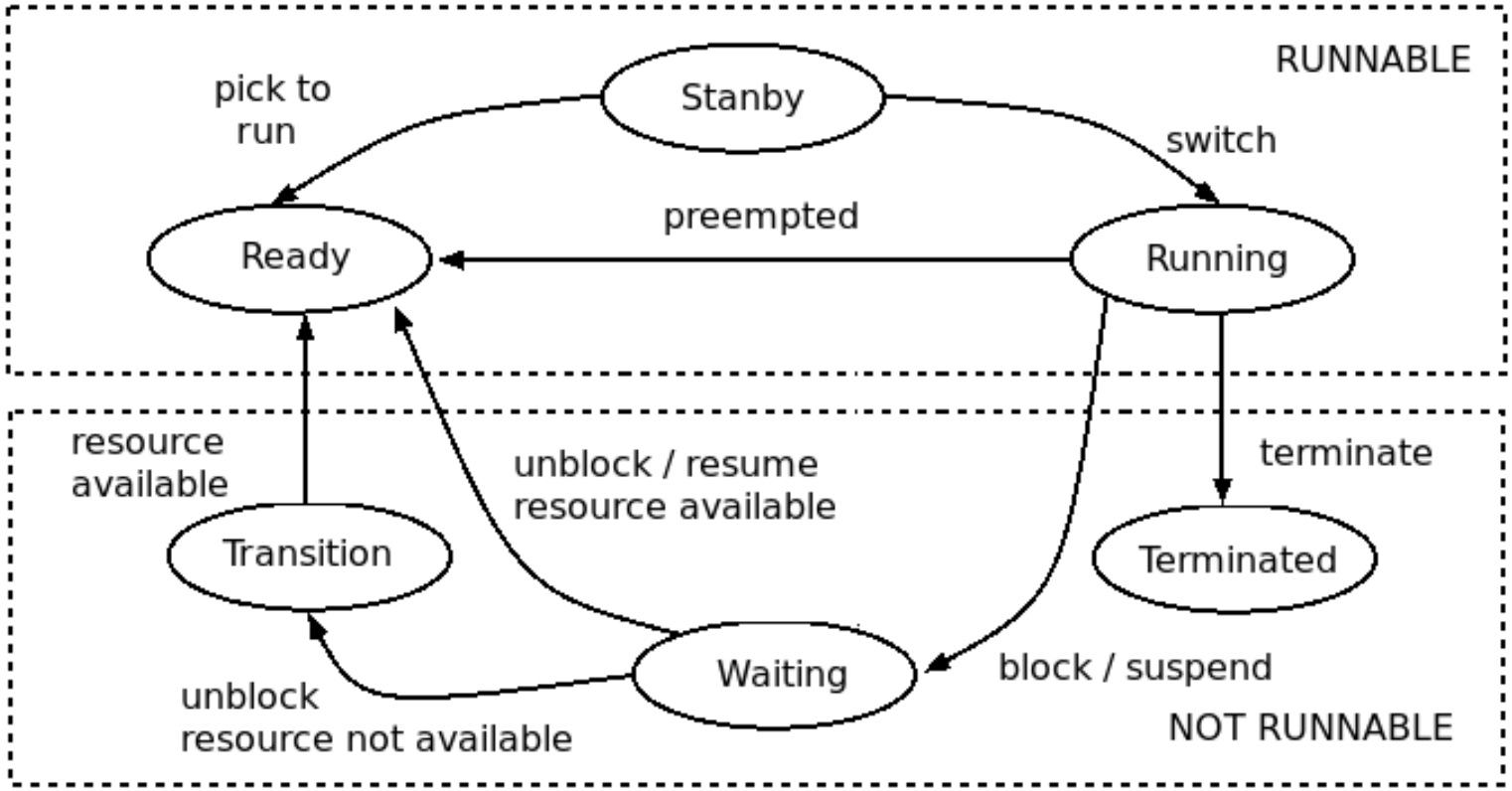
… 4.4.2 – Thread States

- “**ready**” .. a “thread” pode ser agendada para execução, para tanto, o .. “kernel dispatcher” mantém registro de todas as threads prontas e as programa em ordem de prioridade.



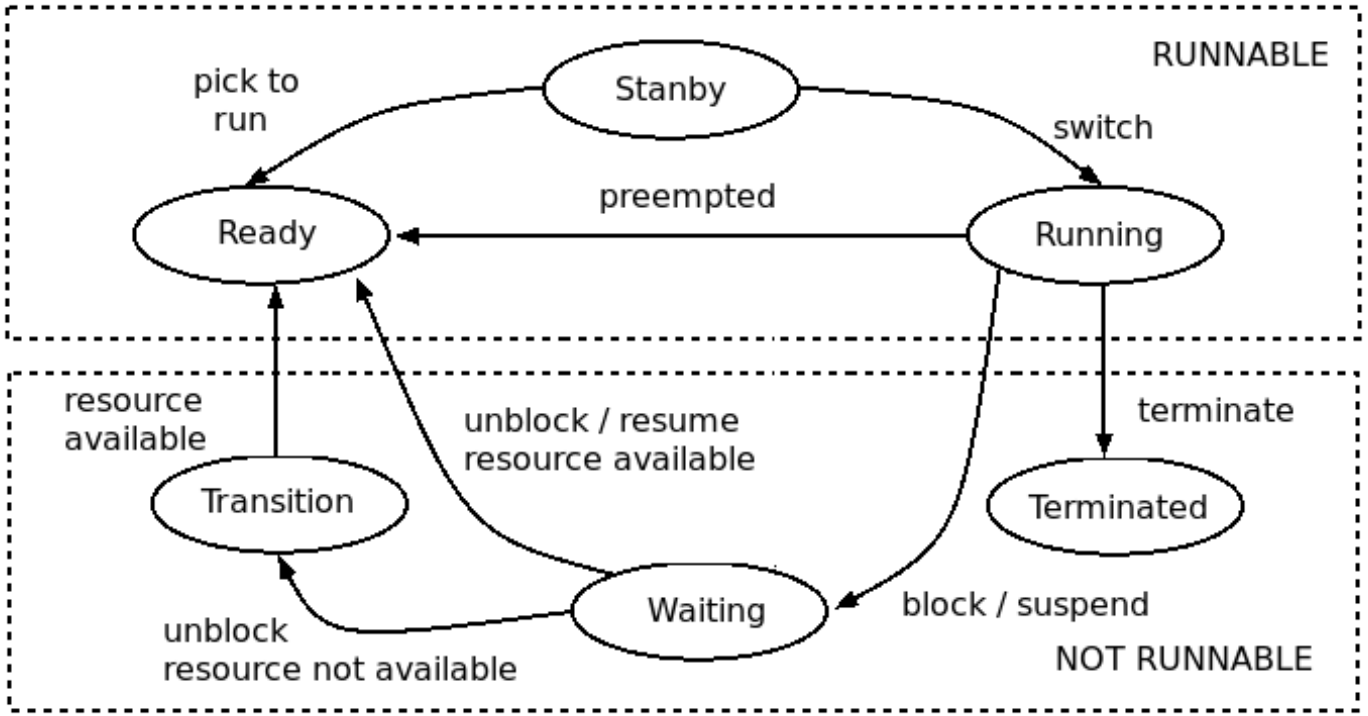
… 4.4.2 – Thread States

- “**standby**” .. uma “thread” em “standby” foi selecionada para ser executada em um processador específico, ou seja, “thread” espera neste estado até que o processador seja disponibilizado.



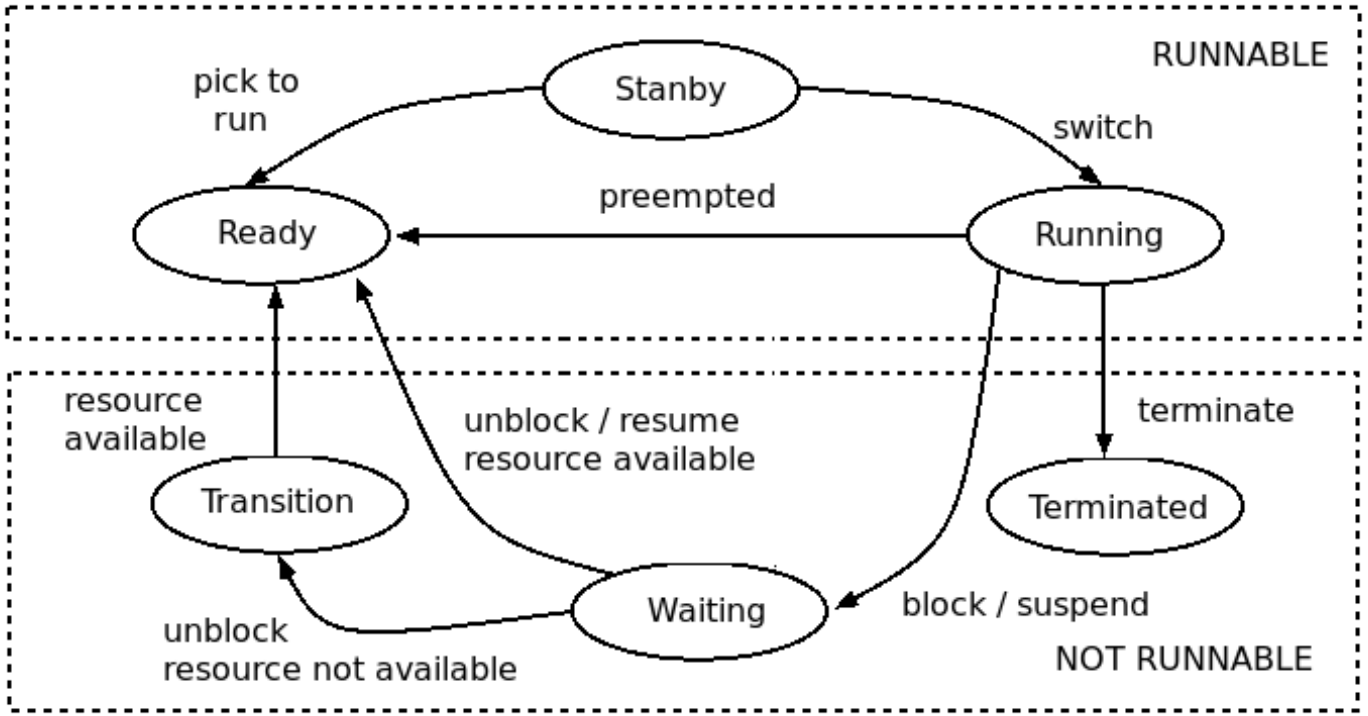
… 4.4.2 – Thread States

- "**running**" .. uma vez que o "kernel dispatcher" realiza uma troca de "thread", a "thread" em espera entra no estado "running"
- .. neste estado inicia a execução e continua até que seja interrompido por uma "thread" com prioridade mais alta, ou quantum de tempo se esgote, ou bloqueie ou termine.



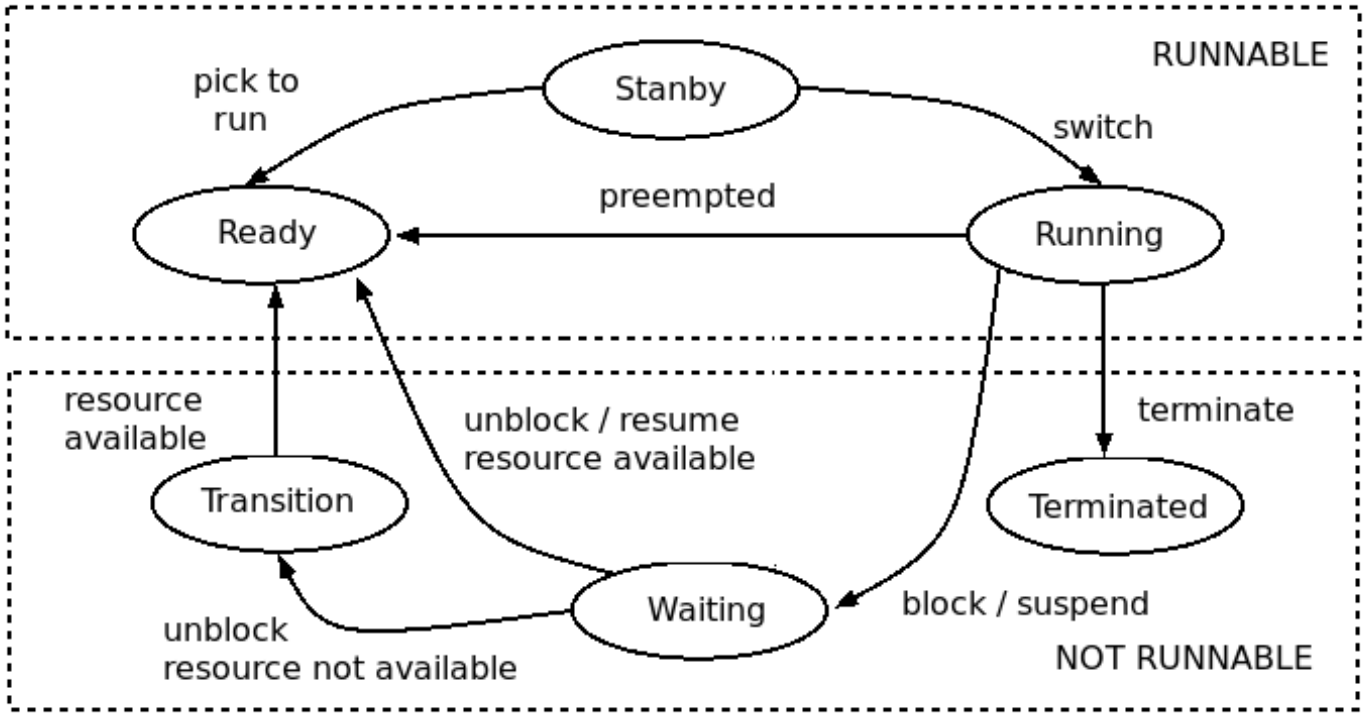
… 4.4.2 – Thread States

- “**waiting**” .. uma “thread” entra no estado de espera quando ... (1) é bloqueada em um evento, (2) espera voluntariamente para fins de sincronização ou (3) a ”thread” suspende a si mesma.
- .. quando a condição de espera é satisfeita, a “thread” passa para o estado “ready” se todos os seus recursos estiverem disponíveis.



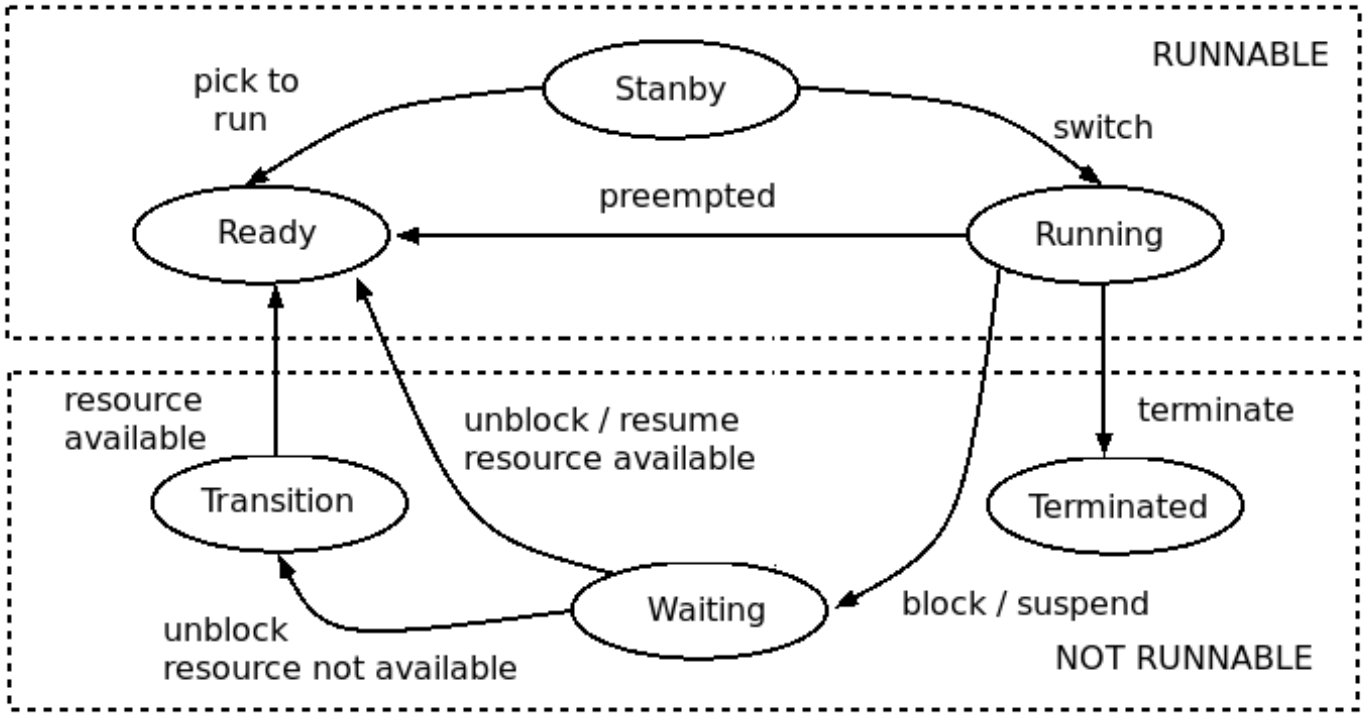
… 4.4.2 – Thread States

- “**transition**” .. uma “thread” entra neste estado depois de esperar se está pronta para ser executada, mas os recursos não estão disponíveis.
- .. p.ex., pilha da “thread” pode ser paginada sem memória, mas quando os recursos estiverem disponíveis, o thread vai muda para “ready”.



… 4.4.2 – Thread States

- "**terminated**" .. uma "thread" pode ser encerrada por si mesma, por outra "thread" ou quando seu processo pai termina.
- .. depois que as tarefas de limpeza são concluídas, a "thread" é removida do sistema ou pode ser retida para reinicialização futura.



4.5 – SOLARIS Thread ans SMP Management

- .. LEITURA COMPLEMENTAR ..

4.6 – LINUX Process and Thread Management

- .. LEITURA COMPLEMENTAR ..