SISTEMAS OPERACIONAIS

Processos e Threads

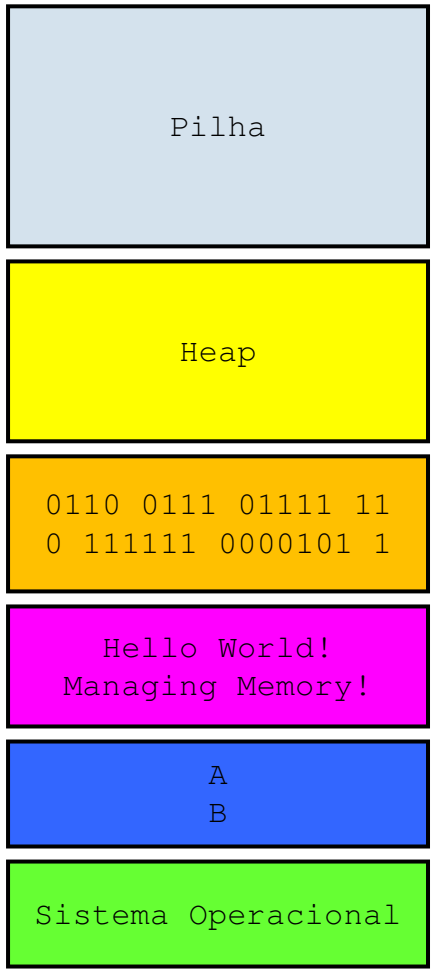
Andreza Leite
andreza.leite@univasf.edu.br

Plano de Aula

- Gerenciamento de Processos
  - Threads
  - Aplicações com múltiplas Threads
  - Concorrência e Compartilhamento
  - Modelos de Sistemas

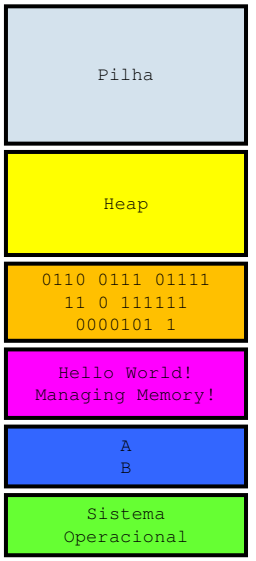
Processos Vs Threads

- Processo em memória:
  - **Pilha**:
    - Memória para alocação:
      - Dados de variáveis locais a sub-rotinas
      - Dados do endereço de retorno de uma sub-rotina.
  - **Heap**:
    - Memória para alocação sob-demanda durante a execução
      - Alocação dinâmica
  - **Código Objeto:**
    - Contém as instruções binárias do código executável do processo.
  - **Dados**:
    - Espaço para as variáveis do processo, declaradas como globais no programa.
  - Área do usuário vs. área do sistema.



Processos Vs Threads

- Processo:
  - Um programa em execução
    - Uma unidade de escalonamento
    - Um fluxo de execução
  - Um conjunto de recursos gerenciadospelo S.O.
    - Registradores (PC, SP, ...)
    - Memória
    - Descritores de arquivos
    - Etc...
  - A troca de contexto é uma operação pesada
    - Deve acontecer cada vez que há decisão de escalonamento

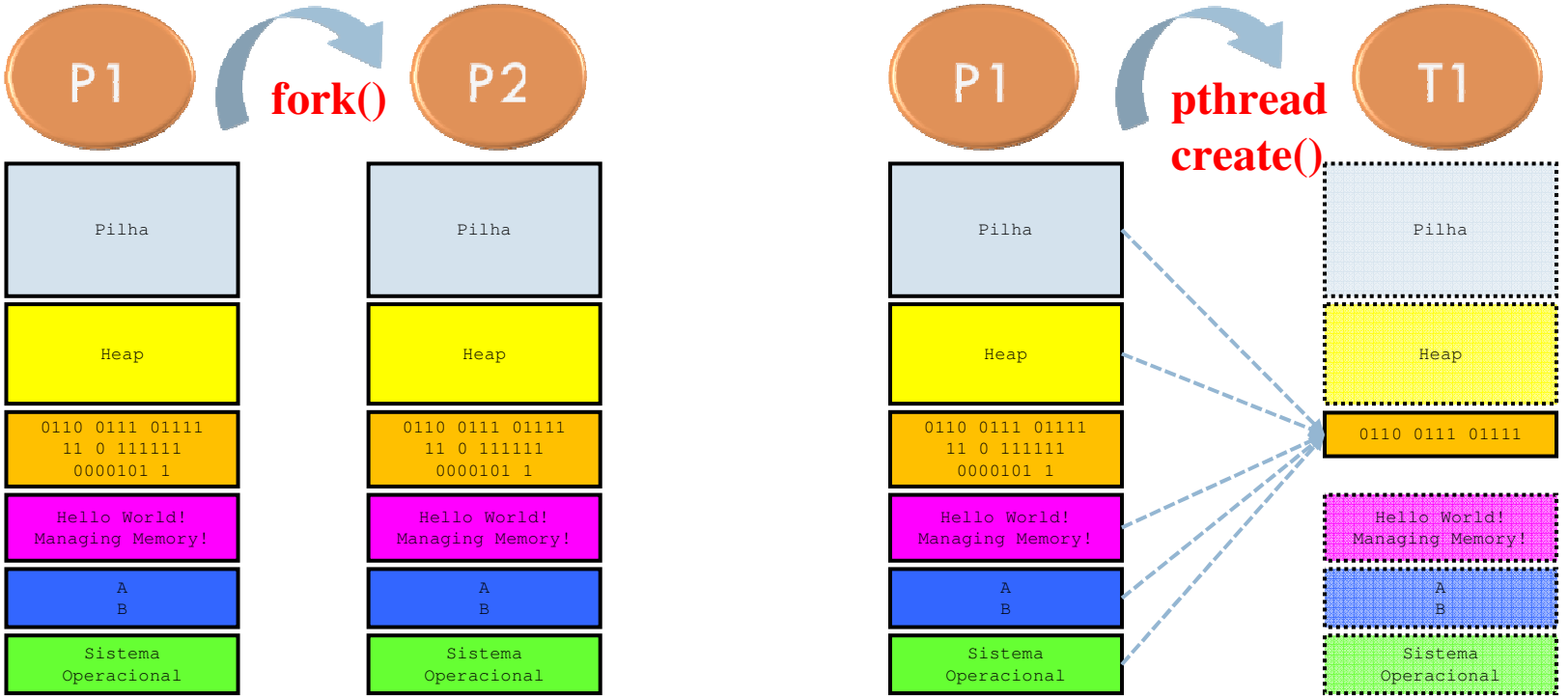


Processos Vs Threads

- Thread:
  - Linha de execução dentro de um processo
    - Seqüência independente de comandos de um programa.
      - Fluxo de Execução
  - Compartilham recursos do processo
  - Idéia simples: associar mais de um fluxo de execução a um processo:
    - Compartilhamento de recursos do processo (PCB - *Process Control Block*)
    - O PCB deve incluir uma lista de threads!

Processos Vs Threads

- Processos & Threads



Processos Vs Threads

- Thread:
  - Vantagens:
    - São processos “leves”
    - Troca de contexto mais rápida;
    - Tempo de criação menor
    - Diminui o tempo de resposta do sistema;
    - Maior facilidade para mesclar threads I/O-bound com threads CPU-bound.
    - Usa eficientemente as arquiteturas multi-processadas/multicores.

Processos Vs Threads

- Threads:
  - Principais aplicações:
    - Aplicações de processamento “paralelo”
      - CPU bound: ganho com várias CPU
      - I/O bound: ganho sempre
        - Se uma thread está bloqueada a outra pode executar
    - Aplicações que acessam dispositivos secundários lentos (disco local ou remoto) e não querem ficar esperando – síncronos - pela resposta para poder continuar executando.

Processos Vs Threads

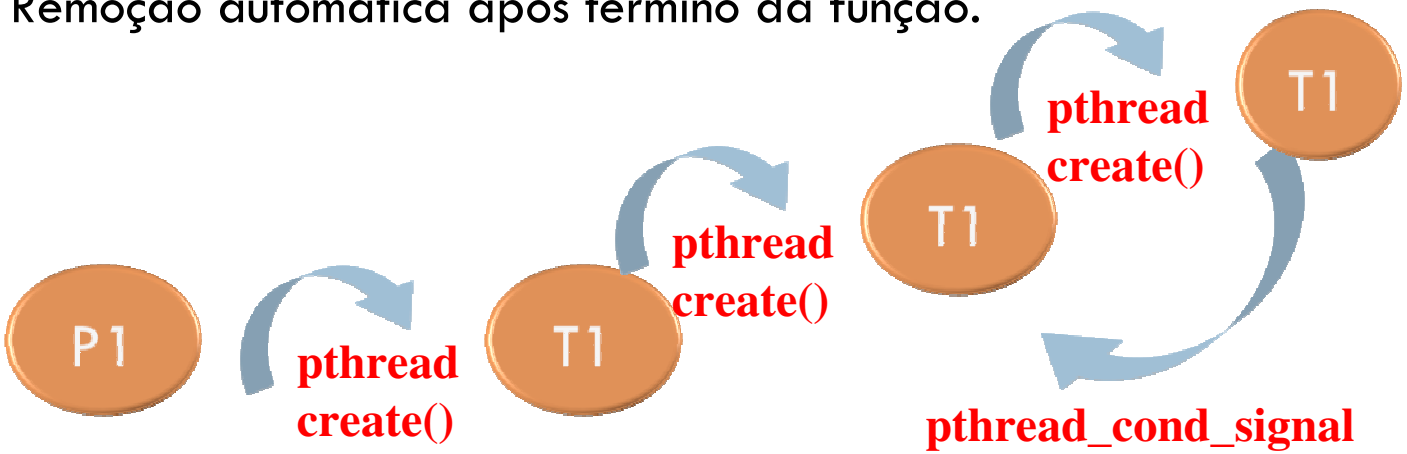
- Threads:
  - Principais aplicações:
    - Aplicações que controlam múltiplos servidores ou múltiplos clientes.
    - Melhorar funcionalidade e performance da interface gráfica.

Processos Vs Threads

- Threads:
  - Características:
    - Herança de recursos alocados pelo processo.
      - Menos custo ao SO para instanciação.
        - Arquivos, Portas de Comunicação, Memória, Ponteiros
    - Possuem contador de programas, ponteiro, contexto, prioridade, etc;
    - Disputam, **entre sí**, pelo processador.
      - Podem executar em paralelismo real – multiprocessamento – ou virtual – multiprogramação.

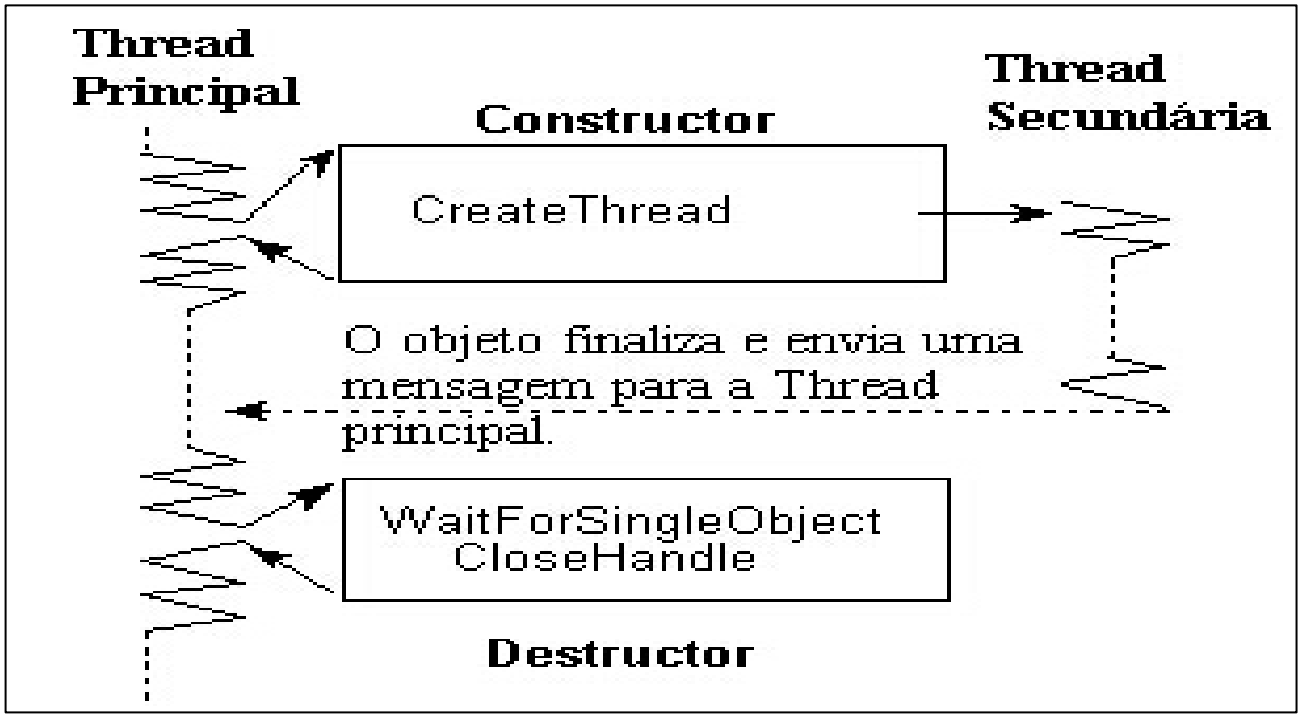
Processos Vs Threads

- Threads:
  - Funcionamento:
    - Pode criar threads filhas (secundárias).
      - Remoção automática após término da função.



Processos Vs Threads

- Threads:
  - Funcionamento:



Processos Vs Threads

- Threads:
  - Diferenças de Processos Convencionais
    - THREADS de um mesmo processo compartilham o mesmo Espaço de Endereçamento.
    - Não existe proteção garantida entre threads de um mesmo processo
      - Uma Thread pode ler, escrever, ou até mesmo eliminar uma pilha de outra Thread do mesmo processo

Processos Vs Threads

- Threads:
  - Diferenças de Processos Convencionais
    - Threads de um mesmo processo **herdam** desde os arquivos abertos, temporizadores, sinais, semáforos, variáveis globais, etc.
  - Uso de THREADS
    - Paralelismo combinado com execução serial

Processos Vs Threads

- Threads:
  - Metas:
    - Performance
      - Tempo total de computação
      - Criação de Threads e Troca de Contexto
        - OVERHEAD
      - CPU-Bound
        - Única Thread ou Múltiplas?
        - Sincronização.
    - Vazão (Throughput)
      - Produtividade

Processos Vs Threads

- Threads:
  - Modelos de Alocação
    - Dinâmica
      - Thread é alocada por demanda
        - Cria, usa e descarta
      - Boa escalabilidade
        - Pode criar novas até o limite do SO
      - Desempenho é reduzido com tempo de criação da Thread

Processos Vs Threads

- Threads:
  - Modelos de Alocação
    - Estática
      - Threads são pré-alocadas
        - Aumenta o desempenho eliminando o overhead de criação antes do uso
        - Baixa escalabilidade pois o número de Threads é predeterminado

Processos Vs Threads

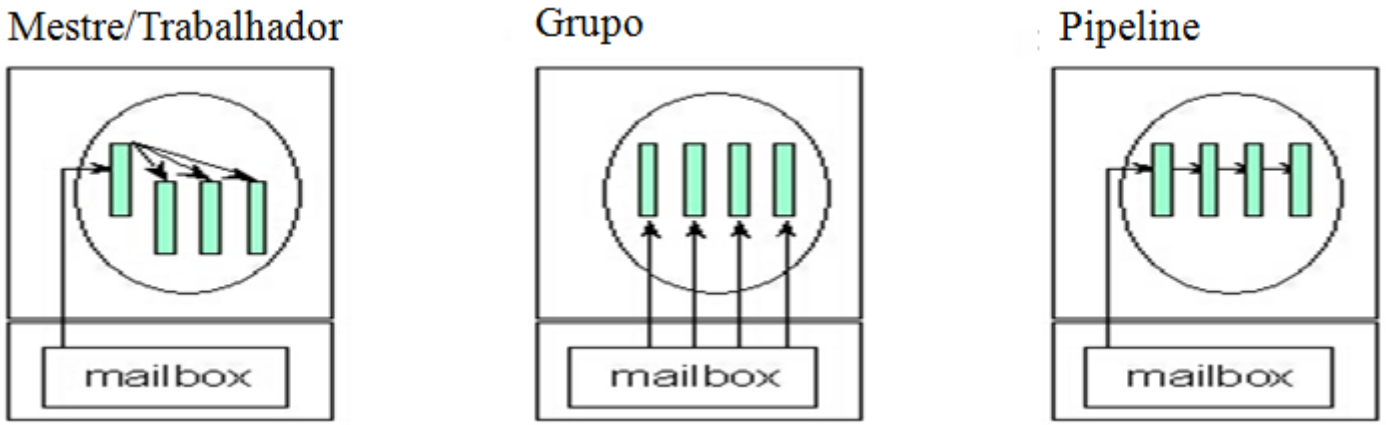
- Threads:
  - Modelos de Alocação
    - Híbrido
      - Pré-aloca n Threads
      - Se todas forem usadas, aloca mais por demanda

Processos Vs Threads

- Threads:
  - Modelos de Programação
    - **Mestre/ Trabalhador**
      - Thread principal – Mestre
      - Atribuição de tarefas às Threads Trabalhadoras
      - Thread Trabalhadora sinaliza estado – Livre ou Ocupada.
      - Filas – Disponibilização, pelo Mestre, de novas tarefas.
    - **Trabalho em Grupo**
      - Tarefas divididas
      - Macro-Tarefas são divididas em micro-tarefas (Threads) concorrentes
    - **Linha de Montagem**
      - Tipo Pipeline

Processos Vs Threads

- Threads: Fluxos de Execução
  - Modelos de Programação



Processos Vs Threads

- Threads: Fluxos de Execução
  - Modelos de Implementação
    - Nível do Kernel
    - Nível do Usuário
    - Multiplexação

Processos Vs Threads

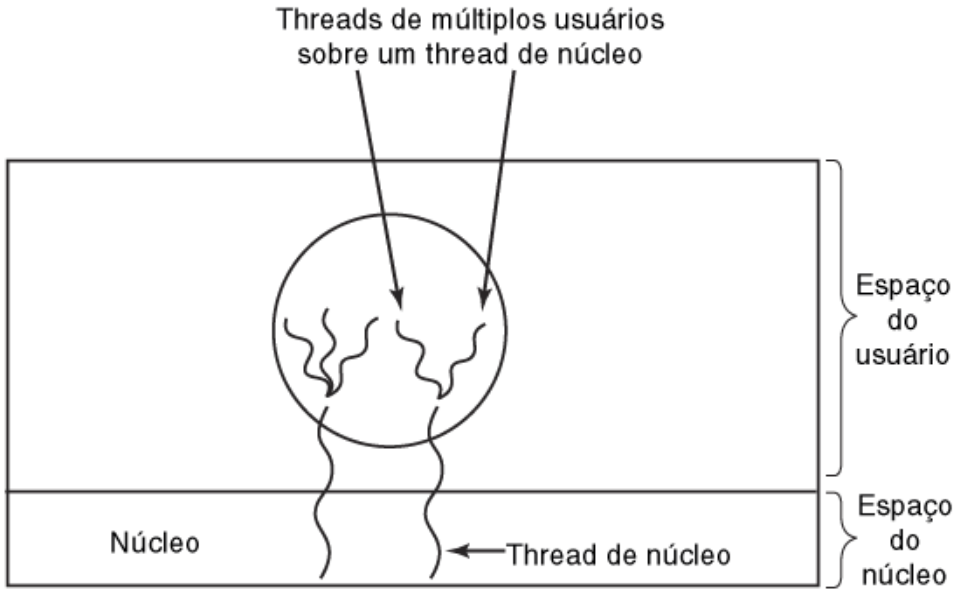
- Threads:
  - Modelos de Implementação
    - Nível do Kernel
      - Modelo 1:1
        - (Ex. Windows 2000, XP, Vista, Unix)
      - Thread visível ao processo possui uma Thread correspondente no Kernel.

Processos Vs Threads

- Threads:
  - Modelos de Implementação
    - Nível do Usuário
      - Modelo N:1
        - Exemplo: DOS + Windows 3.11
      - Threads visíveis ao processo são mapeadas como uma simples Thread no Kernel.
    - Multiplexação
      - M threads de processo multiplexadas para N thread Kernel.
      - Multiplexação de threads de usuário sobre threads de núcleo

Processos Vs Threads

- Threads:
  - Modelos de Implementação
    - Multiplexação
      - O programador pode decidir quantas threads de núcleo usar e quantas threads de usuário multiplexar sobre cada uma.
      - Nesse modelo cada thread de núcleo possui um conjunto de threads de usuário que aguarda por sua vez para usá-lo.



Processos Vs Threads

- Threads:
  - Implementação no Esp. End. do usuário (N:1)
    - O sistema operacional não precisa suportar threads
      - Por exemplo, SO monoprogramado
    - Threads executam sobre o sistema de run-time
      - O Kernel possui uma única Thread
      - O run-time escalona as Threads visíveis ao programador para a Thread do Kernel

Processos Vs Threads

- Threads:
  - Implementação no Esp. End. do usuário (N:1)
    - A troca de contexto é extremamente rápida - entre threads de um mesmo processo - até que outro seja escalonado
    - Escalonamento personalizado
    - Melhor escalabilidade
      - Facilidade de alocação de espaço para tabelas e pilhas no espaço do usuário

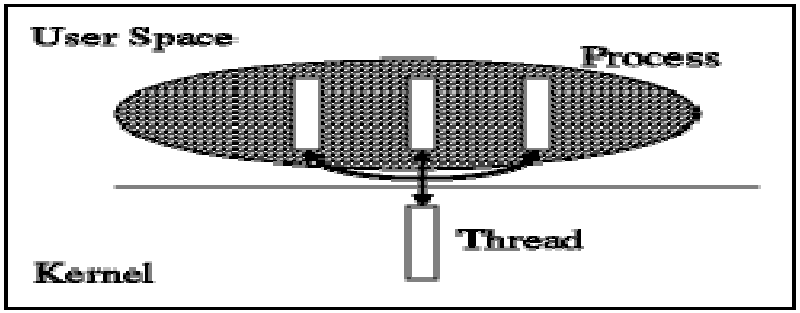
Processos Vs Threads

- Threads:
  - Implementação no Esp. End. do usuário (N:1)
    - Problemas no bloqueio de threads
    - Problemas na utilização do tempo do processador

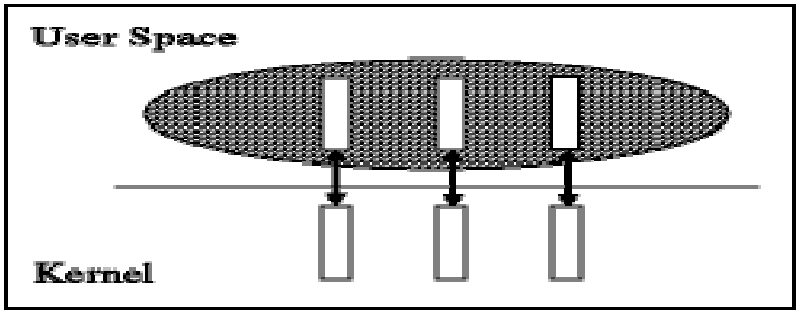
Processos Vs Threads

- Threads:
  - Implementação no Núcleo (1:1)
    - Não é necessário um sistema de run-time
    - Uma tabela de threads por processo
    - Troca de contexto pode ser feita entre threads de processos diferentes
    - Bloqueio de threads simplificado
    - Execução preemptiva

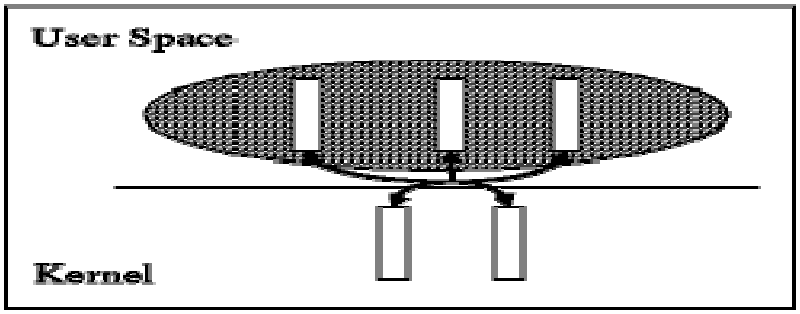
Processos Vs Threads



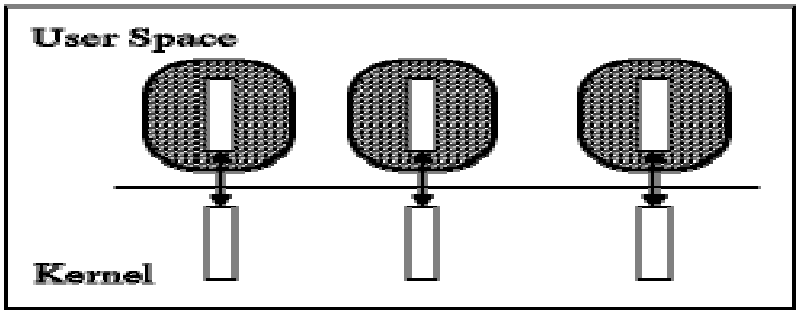
(a)



(b)



(c)



(d)

(a) N: 1

(b) 1:1, com um contexto de processo

(c) M:N

(d) Múltiplos contextos de processo

Processos Vs Threads

- Threads: Exemplos de Utilização
- Implementação de Servidor de Arquivos.
  - Thread única
    - Requisições atendidas em série
    - Baixo Throughput
  - Múltiplas Threads
    - Modelo Mestre/Trabalhador
    - O Mestre recebe as requisições dos clientes e as encaminha para as Threads de trabalho
    - Cada Thread pode atender a uma requisição diferente

Processos Vs Threads

- Threads: Exemplos de Utilização
- Implementação de Servidor de Arquivos.
  - Múltiplas Threads
    - Várias Threads podem executar ao mesmo tempo, compartilhando o processador
    - Enquanto uma thread está bloqueada, outra thread pode ser posta para executar
    - Executam em paralelo em sistemas multiprocessados
    - Todas as Threads compartilham uma mesma cache de arquivo

Processos Vs Threads

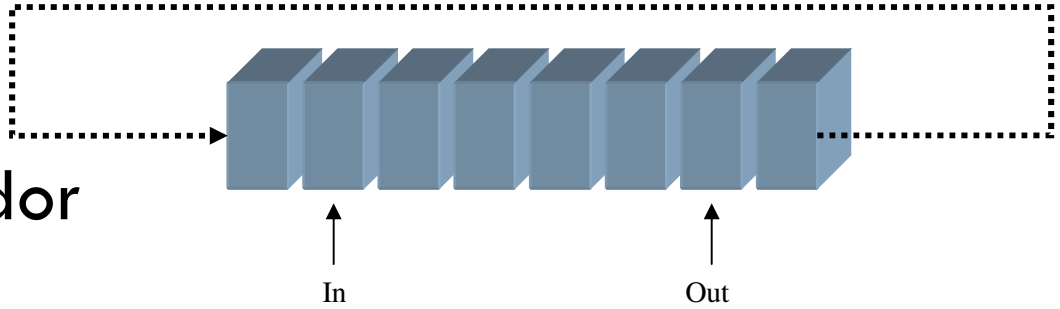
- Sincronização de Threads
  - Sincronização via sinais ou semáforos
    - Thread fica bloqueada esperando sinais
    - O bloqueio da Thread – caso essa não seja a única do processo – **NÃO** bloqueia o processo

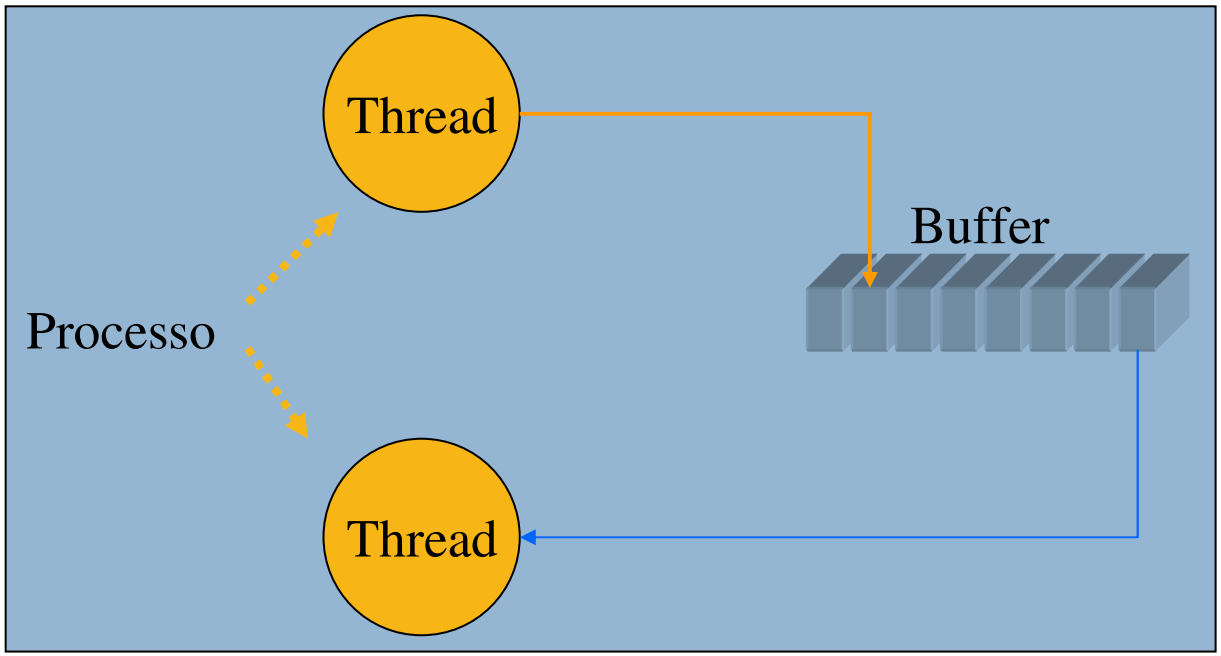
Processos Vs Threads

- Sincronização de Threads
  - Modelo Produtor-Consumidor
    - Paralelismo Inerente
    - Compartilhamento de recurso
      - Buffer de mensagens
    - Utilização de Threads maximiza a utilização do Buffer

Processos Vs Threads

- Produtor/Consumidor





Processos Vs Threads

- Threads: Gerência de Threads
  - Threads Estáticas
    - O número de threads é definido quando o programa é escrito ou compilado
    - Uma pilha de tamanho fixo é associada a cada thread
    - Menor flexibilidade

Processos Vs Threads

- Threads: Gerência de Threads
  - Threads Dinâmicas
    - Threads criadas e destruídas em tempo de execução
      - Alocação por demanda
    - O tamanho da pilha é passado como parâmetro para a chamada de sistema de criação de thread

Processos Vs Threads

- Compartilhamento de Dados
  - Área de dados global do Processo é compartilhada por suas Threads
  - Dados armazenados num espaço de endereçamento comum