PUCPR
GRUPO MARISTA
ESCOLA
POLITÉCNICA

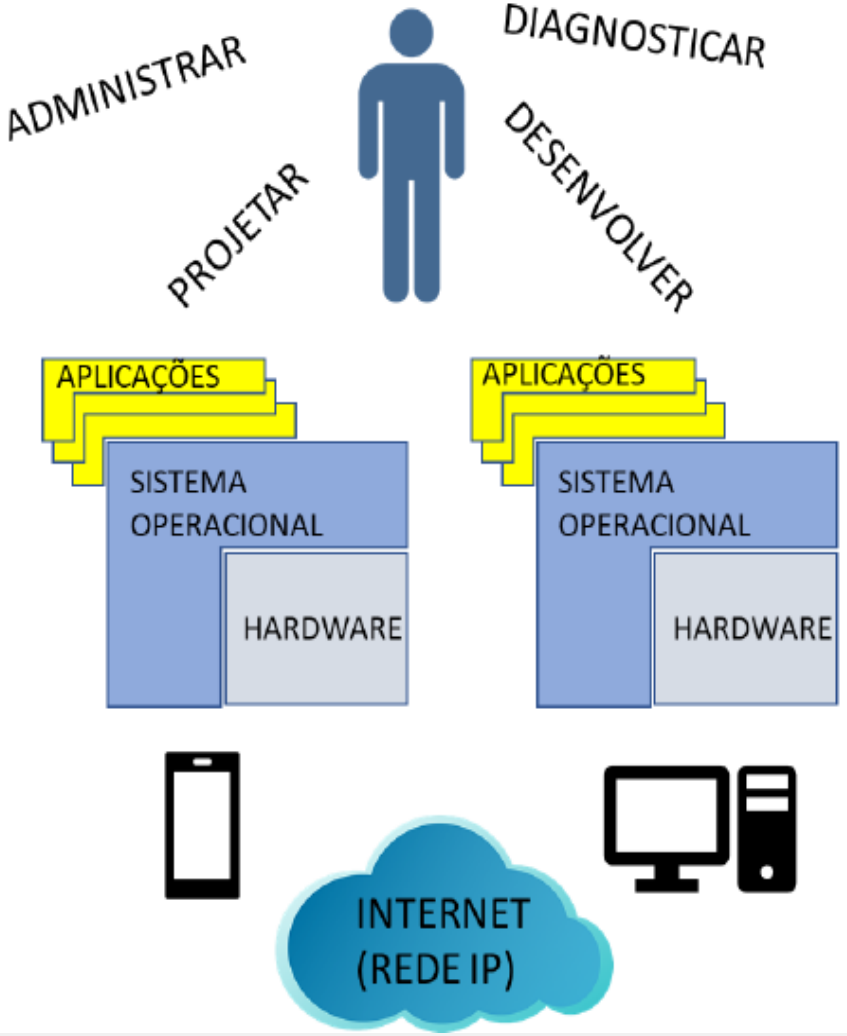
Gerência de Processos

Sistema Operacional

Sistema Operacional

A grande maioria dos dispositivos utilizados pelos usuários (desktop, celulares, tablets, etc.) possuem Sistemas Operacionais.

O Sistema Operacional disponibiliza funções que são **úteis as várias aplicações** do dispositivo.



2

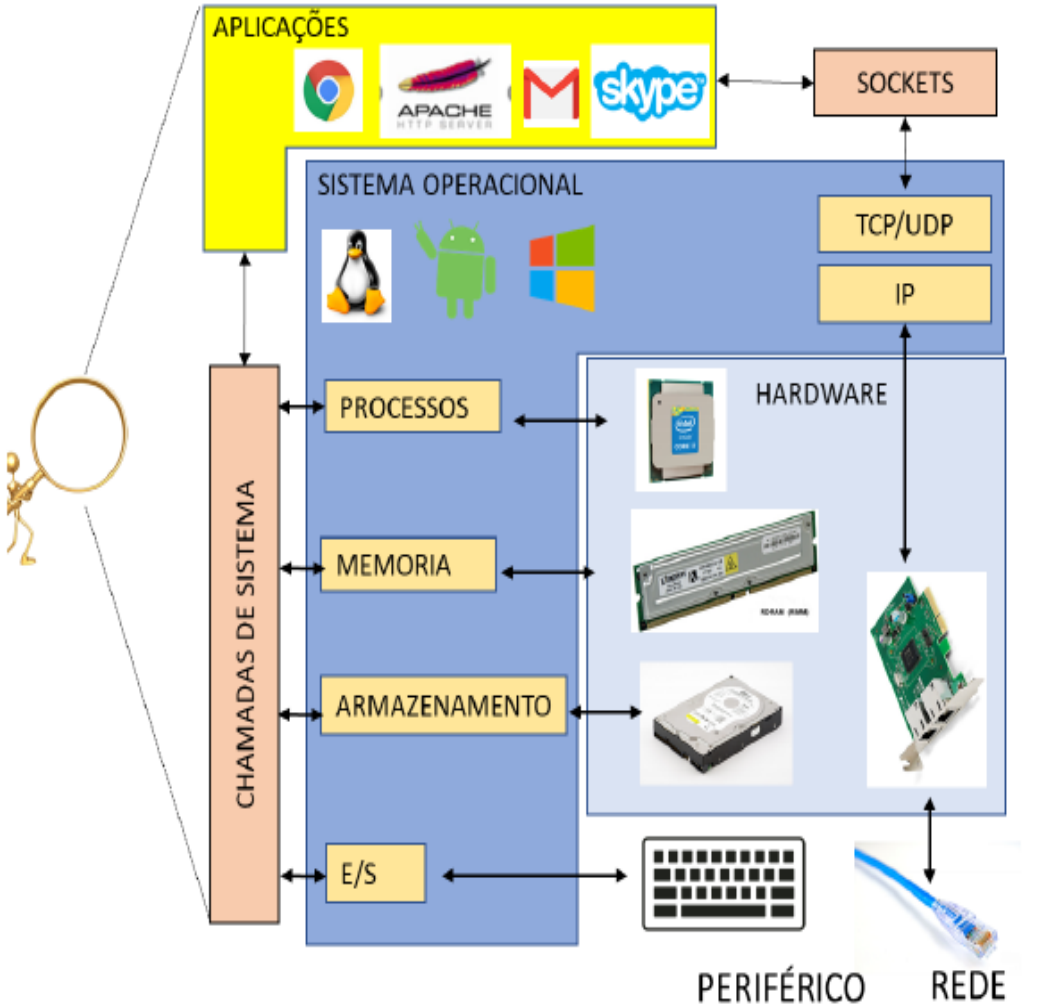
Sistema Operacional

O que faz um sistema operacional?

Muitas aplicações precisam utilizar recursos de hardware: CPU, Memória, Disco e periféricos.

Se as aplicações tiverem acesso direto aos recursos, elas poderiam causar conflitos e erros.

O S.O. intermedia o acesso entre as aplicações e os recursos de hardware, evitando que esses conflitos aconteçam.



3

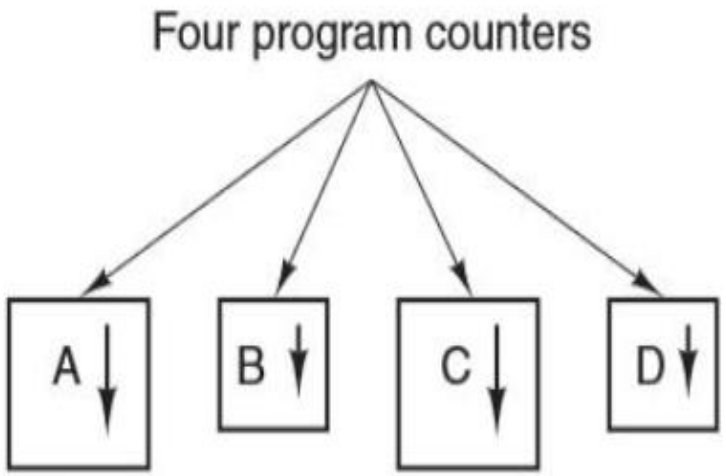
Gerência de Processos

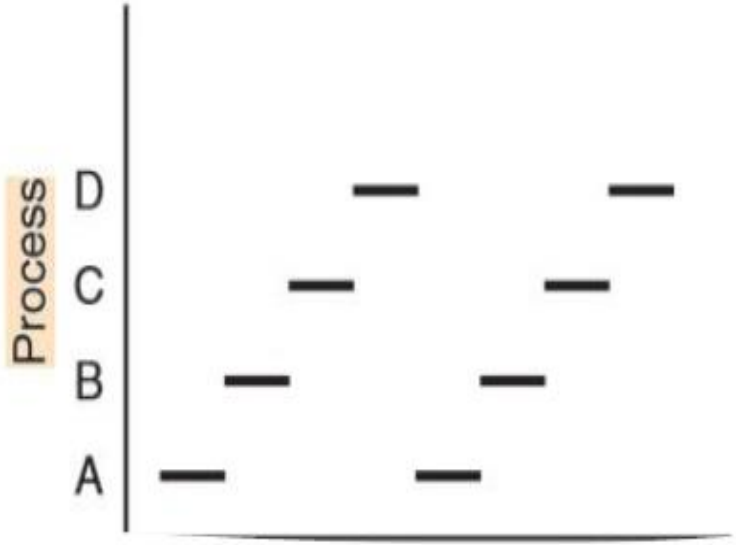
- Implementa a abstração de PROCESSO, que permite que vários programas sejam executados de forma concorrente em uma única CPU ou de forma paralela em várias CPUs.
- Pode implementar uma abstração de THREAD, que permite que partes de um mesmo processo rodem de forma concorrente.
- Determina cada instante qual PROCESSO ou THREAD pode usar a CPU.
- Determina a cada instante qual PROCESSO ou THREAD pode usar um core diferente da CPU.

4

Processo

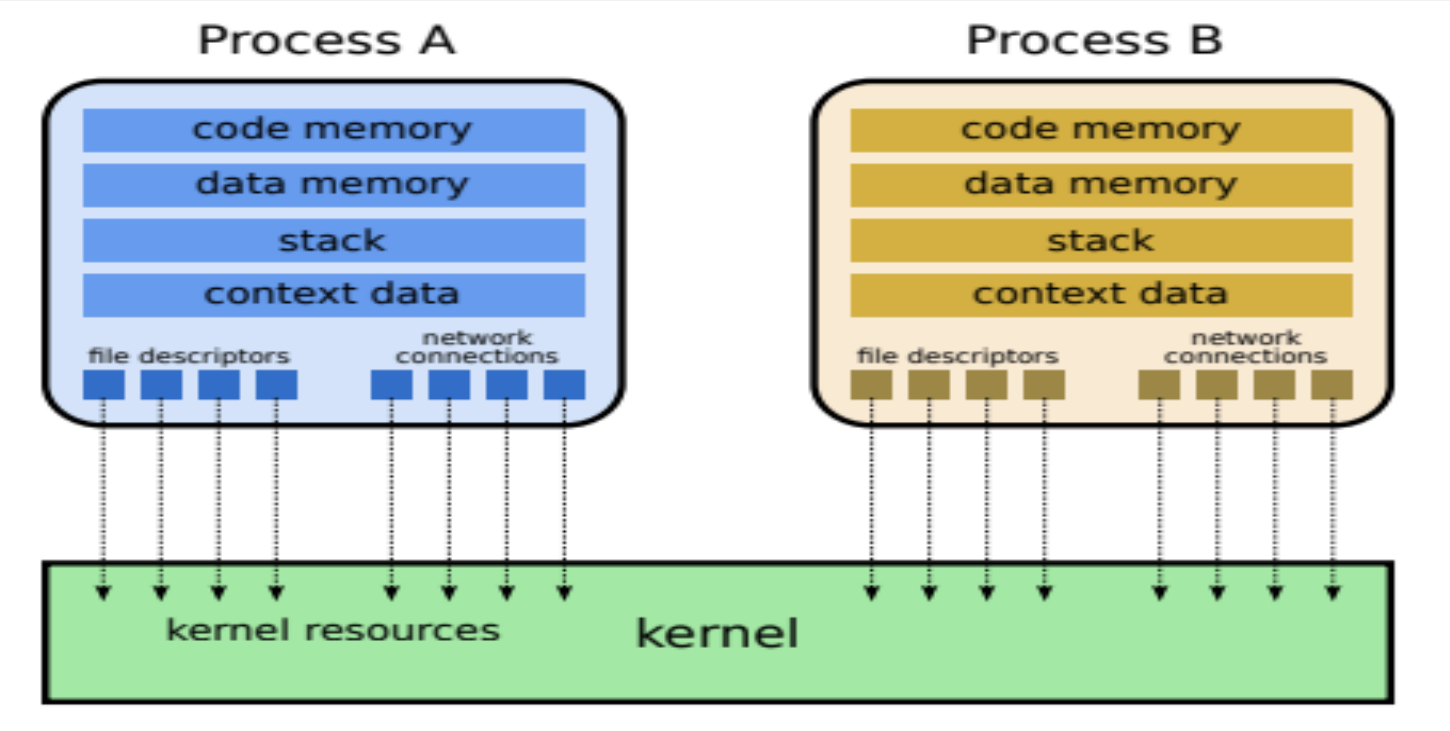
- Abstração do sistema operacional que permite que vários programas sejam EXECUTADOS de forma concorrente ou paralela.
- Em um sistema com uma única CPU esse paralelismo é virtual, pois cada programa é executado em um instante diferente de tempo por vez





Processo

- Processos e seus conjuntos de recursos



6

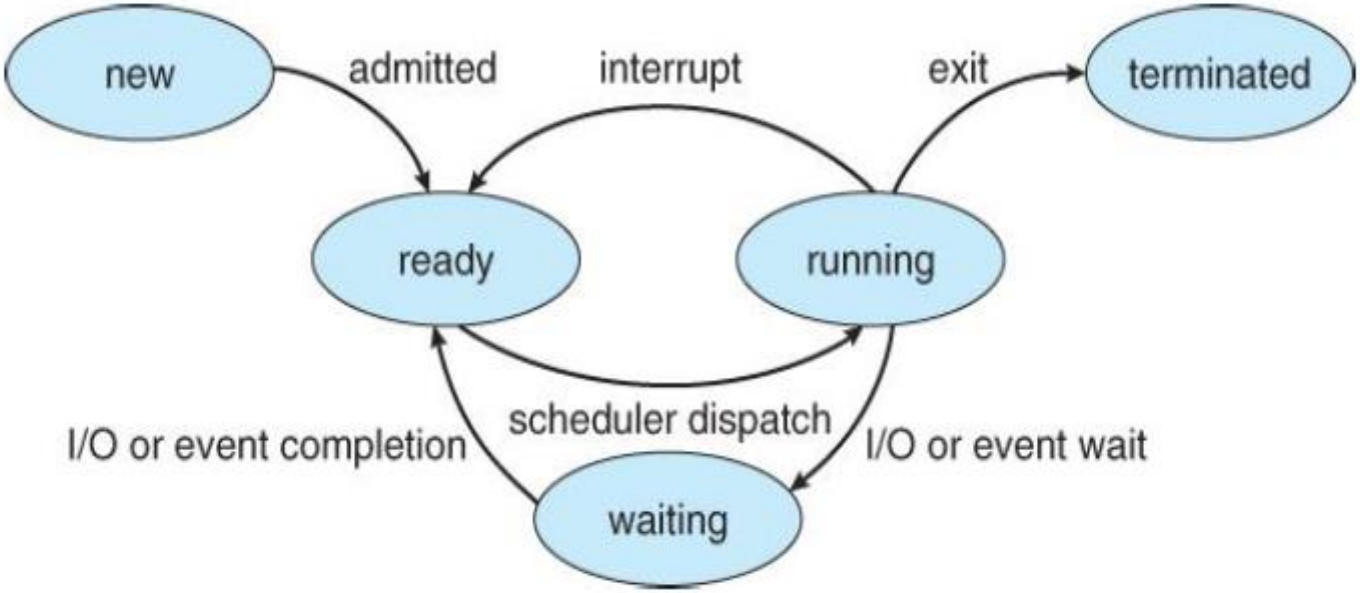
Processo

- Durante a vida do sistema, processos são criados e destruídos.
- Essas operações são disponibilizadas às aplicações através de chamadas de sistema; cada sistema operacional tem suas próprias chamadas para a gestão de processos
- Exemplos de chamadas:

Ação	Windows	Linux
Criar um novo processo	CreateProcess()	fork(),
Encerrar o processo corrente	ExitProcess()	exit()
Encerrar outro processo	TerminateProcess()	kill()
Obter o ID do processo corrente	GetCurrentProcessId()	getpid()

7

Estados de um Processo



READY = Pronto para ser executado e ganhar tempo de CPU
RUNNING = Sendo executado pela CPU
WAITING = Esperado que uma operação de E/S seja completada ou um Sleep (temporizado)
TERMINATED = Removido da memória e do mecanismos de escalonamento

8

Quiz

O que um programa faz enquanto espera uma operação de E/S ser completada pela controladora?

A. Continua a execução normalmente, ocupando memória e CPU.

B. Continua ocupando memória, mas não ocupa CPU.

C. Não ocupa nem memória, nem CPU.

D. Impede que outros programas sejam executados até que sua operação de E/S seja completada.

9

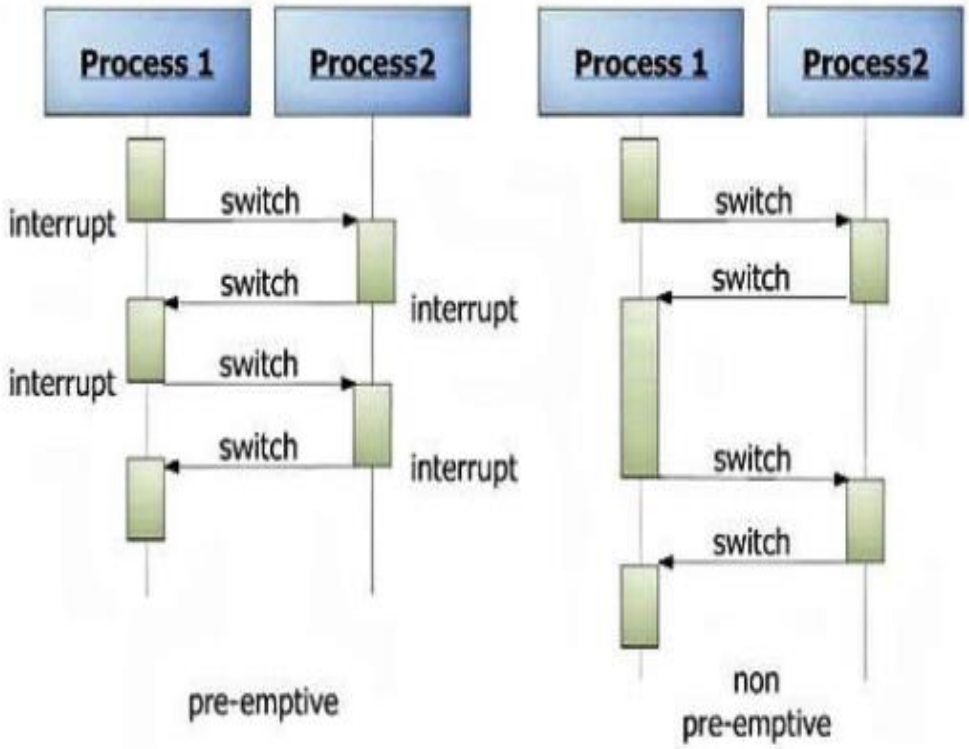
Tipos de Processos

- Processos que usam muita E/S
  - Programas que leem e escrevem em disco
  - Programas que se comunicam em rede
  - Programas que interagem com o usuário
- Processos que usam muita CPU
  - Inteligência artificial
  - Processamento de imagem
  - Programas de otimização (pesquisa operacional)

10

Escalonadores Cooperativos vs Preemptivos

- Preemptivo: os processos são interrompidos em intervalos regulares de tempo.
- Cooperativo: os processos precisam ceder a vez, seja de forma explicita, seja efetuando uma operação de E/S ou "SLEEP".

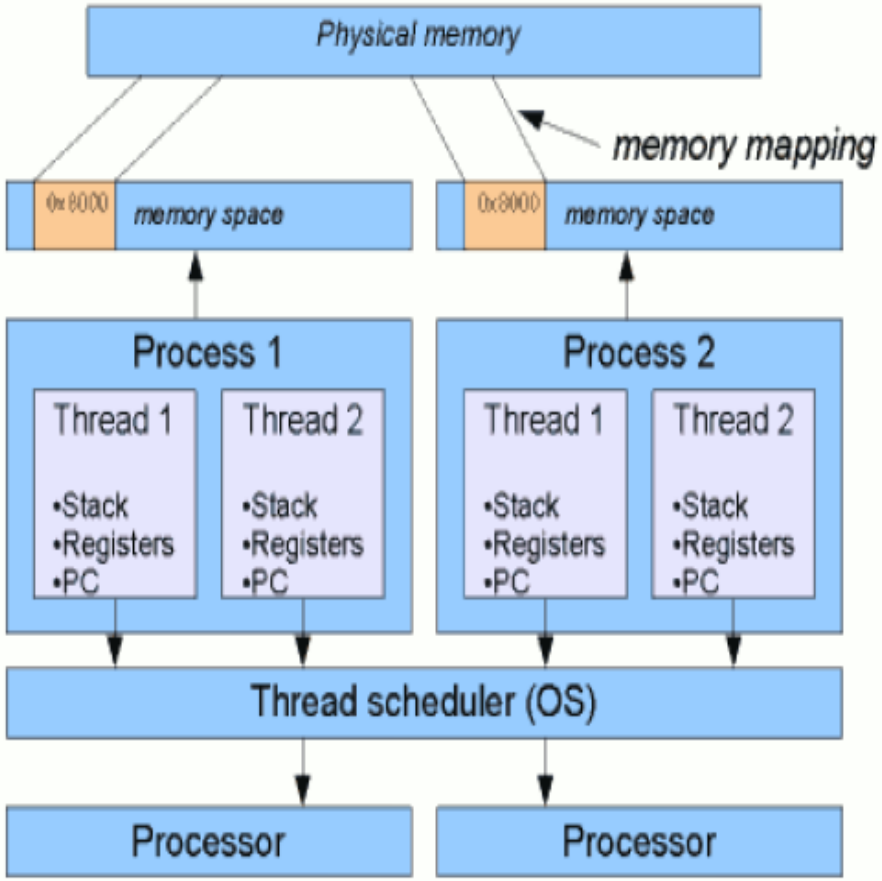


11

Processos e Threads

- Processos são programas que rodam de forma independente, e não compartilham um espaço de endereçamento (não podem ter variáveis em comum).

- Threads são divisões de um processo, que podem ser executados em paralelo, e compartilham o mesmo espaço de endereçamento.



12

Quando usar Threads

- Quando for vantajoso executar partes de um programa (funções) em paralelo.
- Quando for necessário compartilhar variáveis entre as funções que rodam em paralelo.
- Quando for necessário criar dinamicamente muitas vezes a mesma instância de uma função.
- Dependendo do sistema, criar uma Thread pode ser de 10 a 100 vezes mais rápido que criar um processo.

13

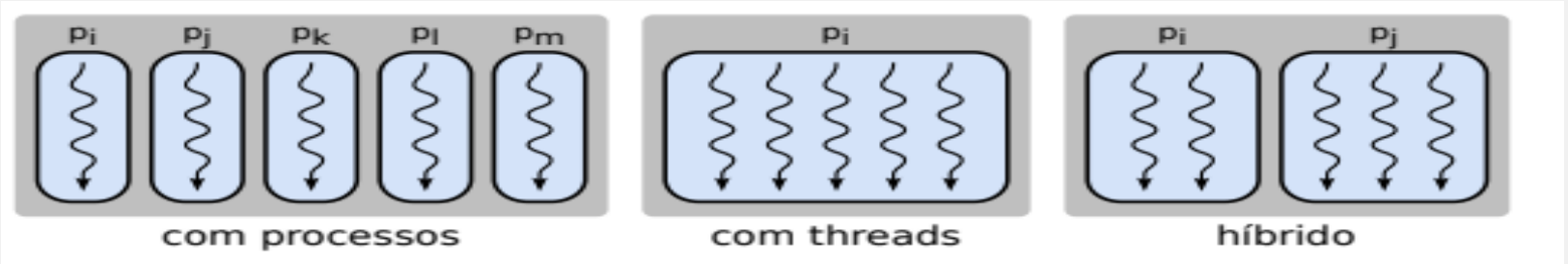
Quando usar Processos

- Quando for necessário desenvolver um sistema que pode ser executado em várias máquinas, pois o processo de comunicação entre processos (e.g. sockets) pode acontecer em máquinas diferentes.

- Para evitar que falhas interrompam o sistema completamente.
  - Um processo pode falhar e os demais continuarem funcionando.
  - Uma thread pode falhar e derrubar o processo, e por consequência todas as demais threads.

14

Abordagens de Desenvolvimento de Aplicações



Característica	Com processos	Com *threads* (1:1)	Híbrido
Custo de criação de tarefas	alto	baixo	médio
Troca de contexto	lenta	rápida	variável
Uso de memória	alto	baixo	médio
Compartilhamento de dados entre tarefas	canais de comunicação e áreas de memoria compartilhada.	variáveis globais e dinâmicas.	ambos.
Robustez	alta, um erro fica contido no processo.	baixa, um erro pode afetar todas as *threads*.	média, um erro pode afetar as *threads* no mesmo processo.
Segurança	alta, cada processo pode executar com usuários e permissões distintas.	baixa, todas as *threads* herdam as permissões do processo onde executam.	alta, *threads* que necessitam as mesmas permissões podem ser agrupadas em um mesmo processo.