Programação Paralela e Distribuída

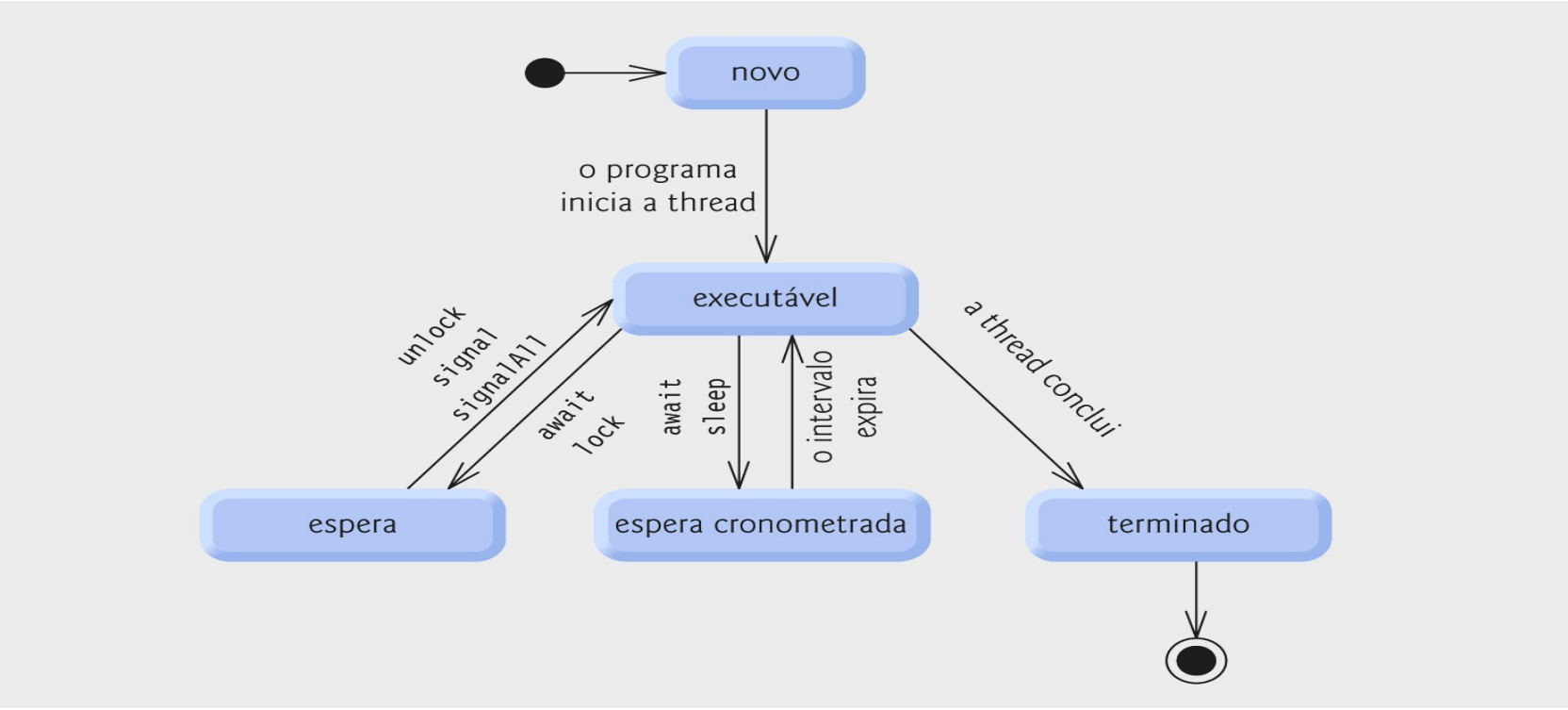
Prof. Hugo Alberto Perlin

Modelo Multi-thread – Threads

- Todo processo possui ao menos um fluxo de execução (thread)
- Nos SOs modernos é possível criar múltiplas threads
- Cada thread é parte de um processo
  - Compartilham recursos: memória e arquivos abertos
- Cada thread tem seu estado e segue um ciclo de vida
- Por que utilizar múltiplas threads?

Thread

- Ciclo de vida de uma thread



Compartilhamento de Recursos

- Threads podem facilmente compartilhar recursos (memória, arquivos abertos, etc)
- Isso é bom e ruim…
- Por que?

Condição de corrida (race condition)

- Se o desenvolvedor não gerenciar corretamente o acesso a recursos compartilhados, resultados indesejados podem ocorrer
- Lembre como ocorre o processo de leitura/escrita na memória
- Exemplo

Sincronização com Lock

- Tipo de objeto que permite regular o acesso a recursos compartilhados
- Necessário indicar a aquisição (.acquire()) e liberação (.release()) do lock
- Regras:
  - se está liberado: adquirir lock muda o estado para bloqueado
  - se está bloqueado: adquirir lock aguarda até outra thread liberar
  - se está liberado: liberar gera uma RuntimeError
  - se está bloqueado: liberar muda o estado para desbloqueado
- Aquisição consecutiva pela mesma thread são bloqueantes
- *Exemplo*

DeadLock

- Situação onde threads necessitam de aquirir locks de forma alternada
- Exemplo

RLock

- Tipo de lock que permite aquisições sucessivas pela mesma thread
- Somente a thread detentora do lock pode fazer a liberação
- O número de liberações deve ser igual ao número de aquisições
- Exemplo

Condição

- Usa um objeto interno lock para gerenciar o acesso aos recursos compartilhados
- Threads interessadas em um determinada estado devem chamar o método wait() repetidamente até que o estado desejado ocorra
- Threads que modificam o estado devem chamar o método notify() ou notifyAll() para indicar a mudança de estado para as threads aguardantes
- Exemplo

Semáforo

- Conceito proposto pelo cientista da computação Edsger W. Dijkstra
- Objeto que permite o controle de recursos compartilhados
- Permite controlar o acesso a recursos com capacidade limitada (ex. servidor)
- Matém um contador interno para determinar o estado do semáforo
- Regras:
  - contador é decrementado a cada aquisição
  - se o contador é igual 0, a aquisição bloqueia
  - contador é incrementado a cada liberação
- A thread pode liberar um semáforo sem ter adquirido
- Exemplo

Evento

- Um dos mecanismos mais simples para comunicação entre threads
- Uma thread emite um sinal
- Outras threas aguardam pelo sinal
- Para emitir um sinal utiliza-se os métodos set() e clear() em sequência
- Para aguardar um sinal utiliza-se o método wait(), que irá bloquear até que o sinal seja recebido
- Exemplo

Queue

- Implementa filas multi-produtor e multi-consumidor
- Útil quando dados devem ser trocados de maneira segura entre múltiplas threads
- A classe Queue implementa todas as necessidades de controle de concorrência para garantir a integridade dos dados
- São disponibilizadas três tipos de filas: FIFO, LIFO e Prioridade
- Permite indicar o tamanho máximo do buffer
- Para colocar um item, utiliza-se o método put(), que irá incluir um item se o buffer não estiver cheio, caso contrário aguarda até liberar espaço
- Para retirar um item, utiliza-se o método get(), que irá pegar um item se o buffer não estiver vazio, caso contrário aguarda até existir um item
- Ao concluir um tarefa, após a chamada do método get(), deve-se invocar o método task_done(), para indicar a finalização
- O método principal pode bloquear até que todos os items da fila sejam processados, utlizando o método join()
- Exemplo

Exercício

- Utilizando os mecanismos de controle e sincronização, crie um software em Python que dada uma url inicial, procure por links naquela página que levam a páginas filhas. Faça isso até encontrar todas as páginas filhas (limite a busca somente para o domínio da url inicial), cuidando para não cair em um loop infinito. Para cada página conte o número de tags html. O processo deve ser feito utilizando no máximo 5 threads.
- Utilize a biblioteca urllib2 para acessar o conteúdo de urls