Sistemas Operacionais

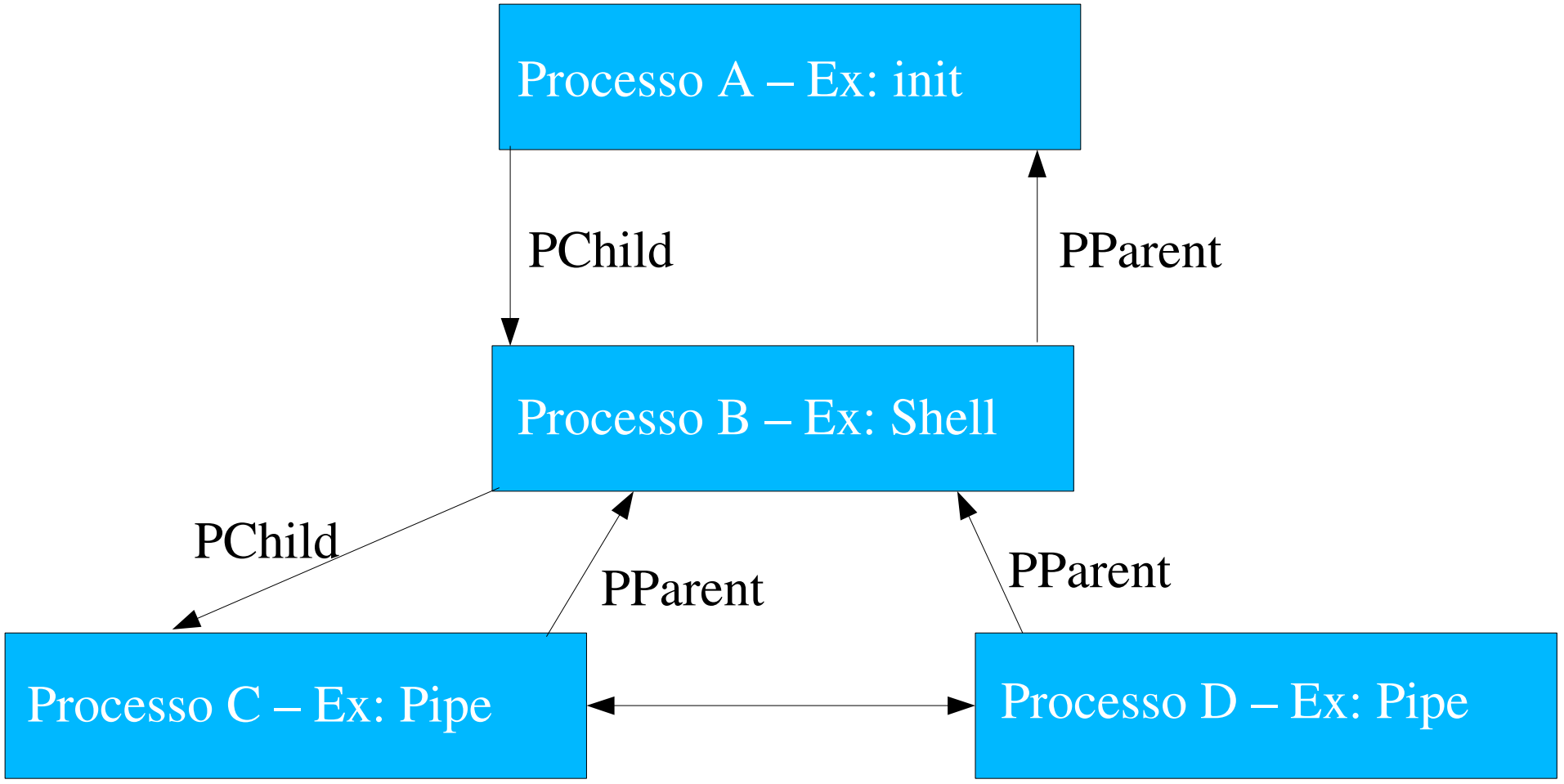
Rodrigo Rubira Branco
rodrigo@kernelhacking.com
rodrigo@fgp.com.br

Hierarquia de Processos

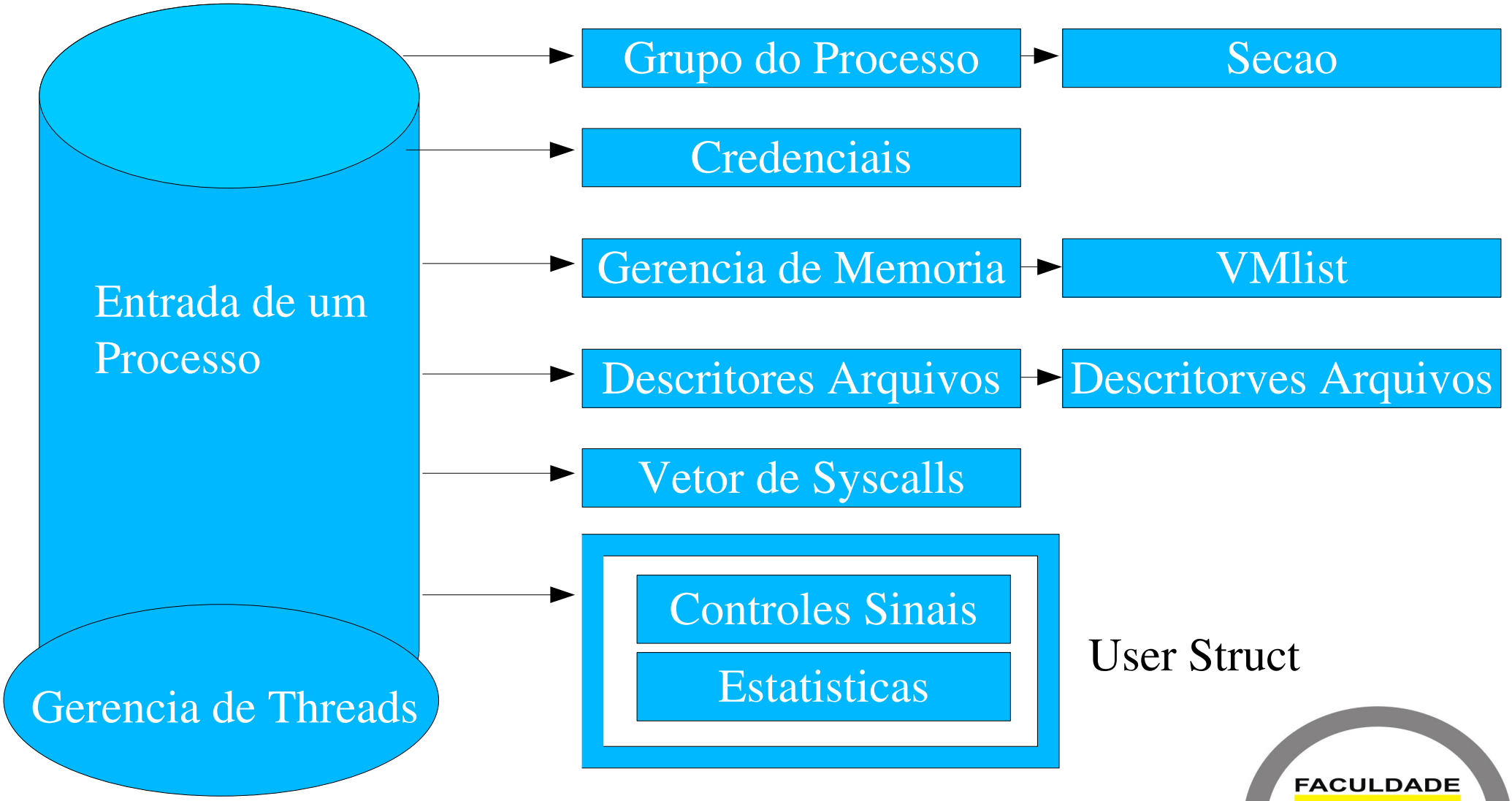
Em sistemas operacionais Unix-like, existe uma clara hierarquia de processos, onde todos os processos provem de um mesmo processo pai, conhecido como init.

Cada processo possui uma estrutura de memoria responsavel pelo gerenciamento dos dados referentes a este processo, e conhecida como task_struct.

Hierarquia de Processos



Estrutura de Gerencia de Processos



Fork()

A criacao de novos processos se da por uma chamada a fork(), que gera um “clone” do processo atual, compartilhando inclusive o espaco de enderecamento (com recurso de COW – copy-on-write, ou seja, se modificado ira gerar uma copia real) e o timeslice (tempo restante de processador).

Tal chamada (fork()) ira retornar para o processo pai, o pid do novo processo criado (processo filho), e ao processo filho, retornara 0, permitindo assim que no codigo, faca-se clara a diferenciacao entre a execucao do processo pai e do processo filho.

A chamada execve() pode ser utilizada para substituir a imagem do processo atual, por um novo processo que sera entao executado.

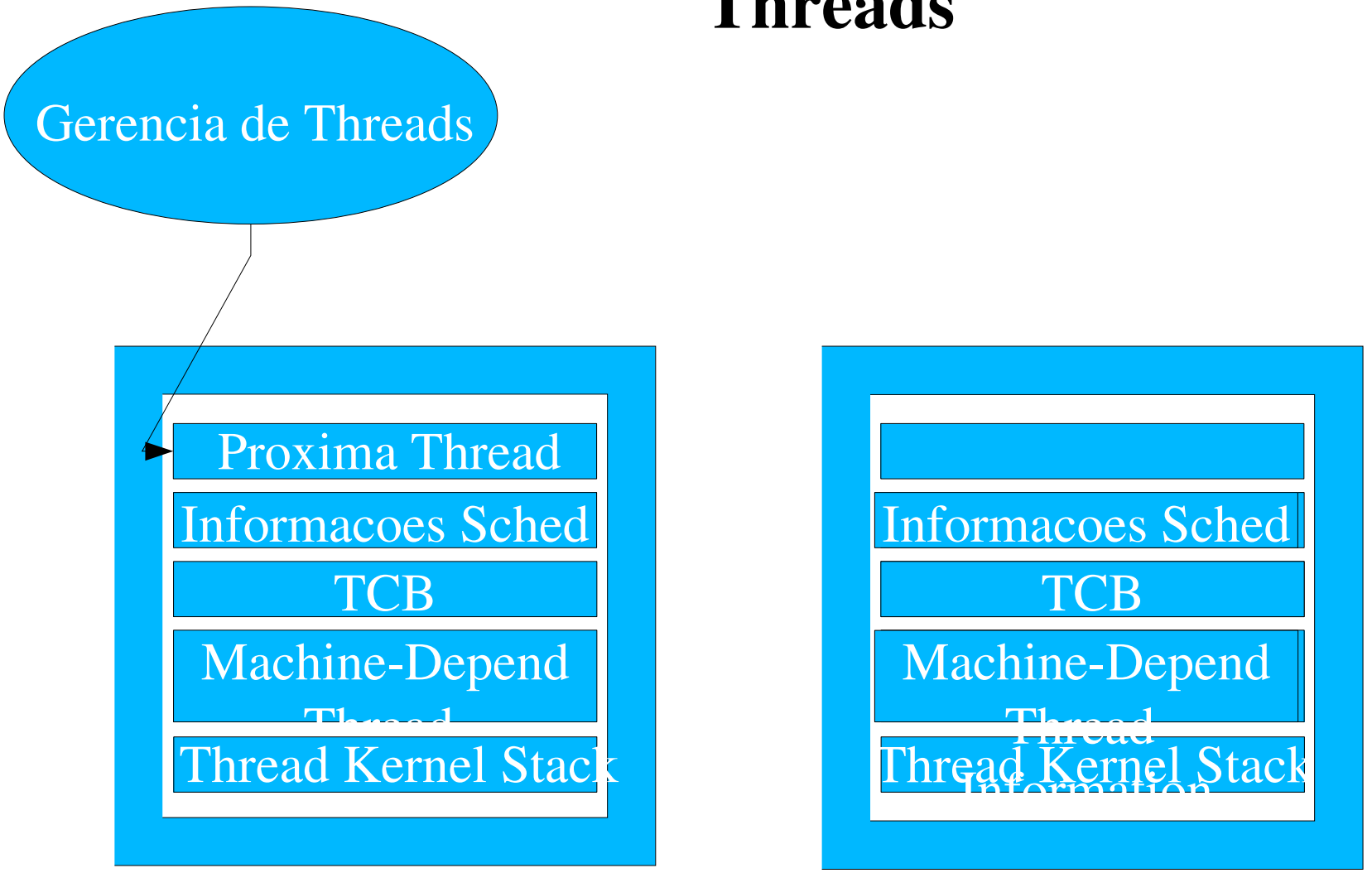
Threads

Forma extremamente poderosa de se aproveitar de recursos de sistemas multiprogramados, principalmente em ambientes distribuidos ou maquinas multiprocessadas (existem ganhos em diversas situacoes que nao sejam estas, como por exemplo, evitando-se desperdicio de tempo por bloqueios em chamadas ao sistema).

Podem ser gerenciadas de duas maneiras diferentes, via kernel, onde o sistema nativamente suporta threads, ou via bibliotecas, que "simulam" para o programador o ambiente das threads (as bibliotecas na verdade utilizam-se de chamadas ao sistema para gerar em kernel conceitos muito similares que os das threads, sendo no caso do linux por exemplo, cada thread com seu proprio PID, porem compartilhando enderecamentos de memoria).

Por este motivo a figura de processos usa o padrao BSD.

Threads


Gerencia de Threads
Proxima Thread
Informacoes Sched
TCB
Machine-Depend
Thread Kernel Stack
Informacoes Sched
TCB
Machine-Depend
Thread Kernel Stack

Process Control Block / Thread Control Block

Estrutura de dados que visa armazenar informacoes sobre os processos/threads.

Dentre as informacoes armazenadas no TCB, importante salientar:

- Registradores de proposito geral
- Ponteiros de Pilha
- Contador de Programa
- Status do Processador
- Registradores de Gerencia de Memoria (MMU por exemplo)

Process Control Block / Thread Control Block

Alem das estruturas (sub-estruturas) mostradas nos desenhos, a task_struct armazena outras informacoes importantes sobre os processos:

- PID (compartilhado pelas threads se o sistema tiver thread em kernel e unico para cada thread se for threads em bibliotecas)
- Estado dos Sinais (sinais pendentes, enviados, mascara de sinais)
- Estado do Processo

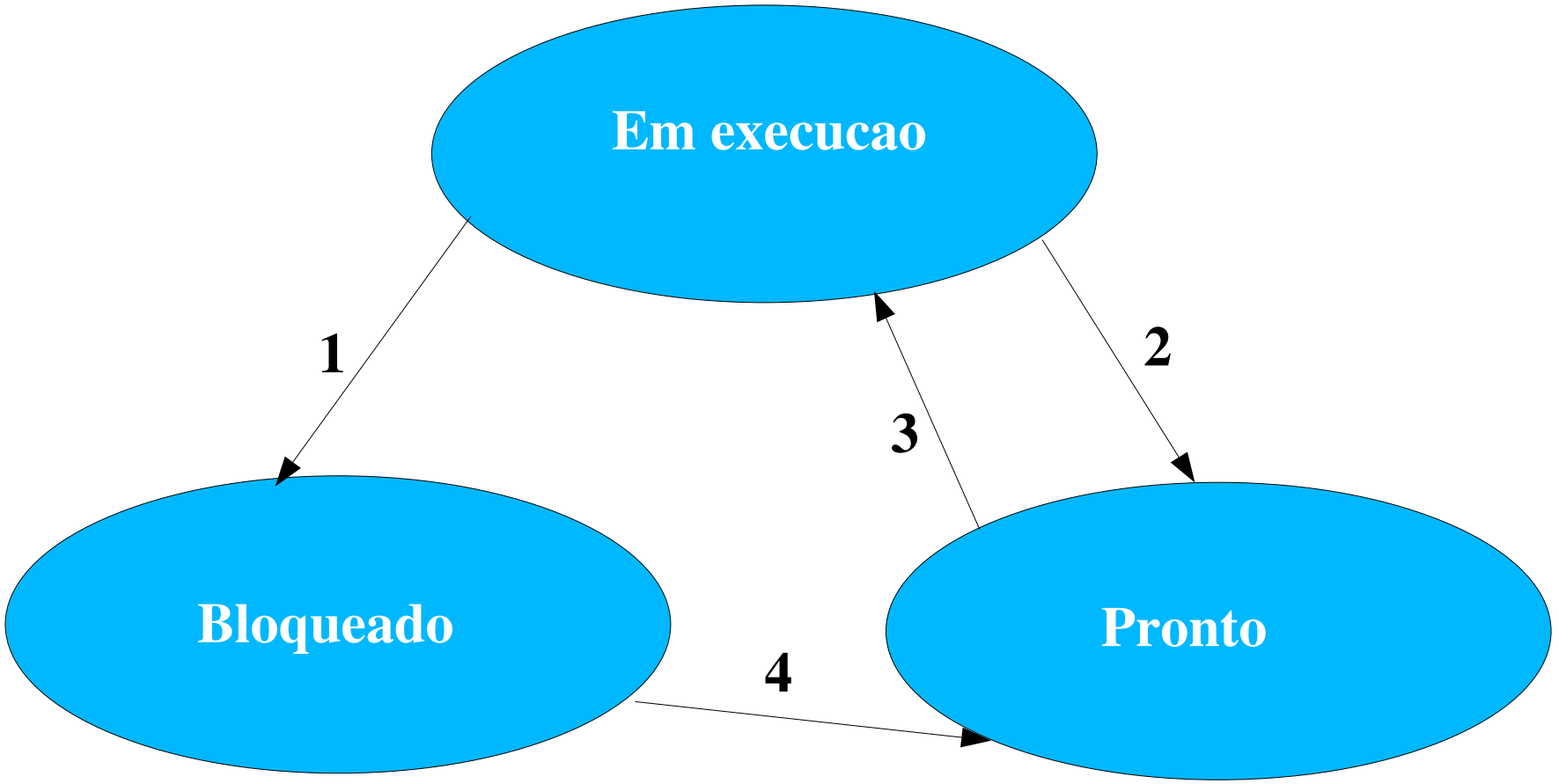
Estado de Processos

Um processo pode encontrar-se em um dos estados a seguir:

- NEW
- NORMAL
  Runnable, Sleeping, Stopped
- ZOMBIE

Quando o processo vai ser criado pela syscall fork(), ele sera marcado como NEW. Apos a alocacao dos recursos necessarios para a execucao, o processo sera marcado como NORMAL (runnable – pronto para executar ou executando, sleeping – esperando por algum evento, stopped – parado por algum sinal ou pelo processo pai). O processo sera marcado como ZOMBIE apos seu termino, enquanto nao os recursos alocados e nao comunicar seu processo pai.

Maquina de Estados (SO Modernos - Tanenbaum)



1. O processo bloqueia aguardando uma entrada
2. O escalonador (scheduler()) seleciona outro processo
3. O escalonador seleciona este processo
4. A entrada torna-se disponivel

FIM! Será mesmo?

DÚVIDAS?!?

Rodrigo Rubira Branco
rodrigo@kernelhacking.com