



Sistemas Operacionais

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

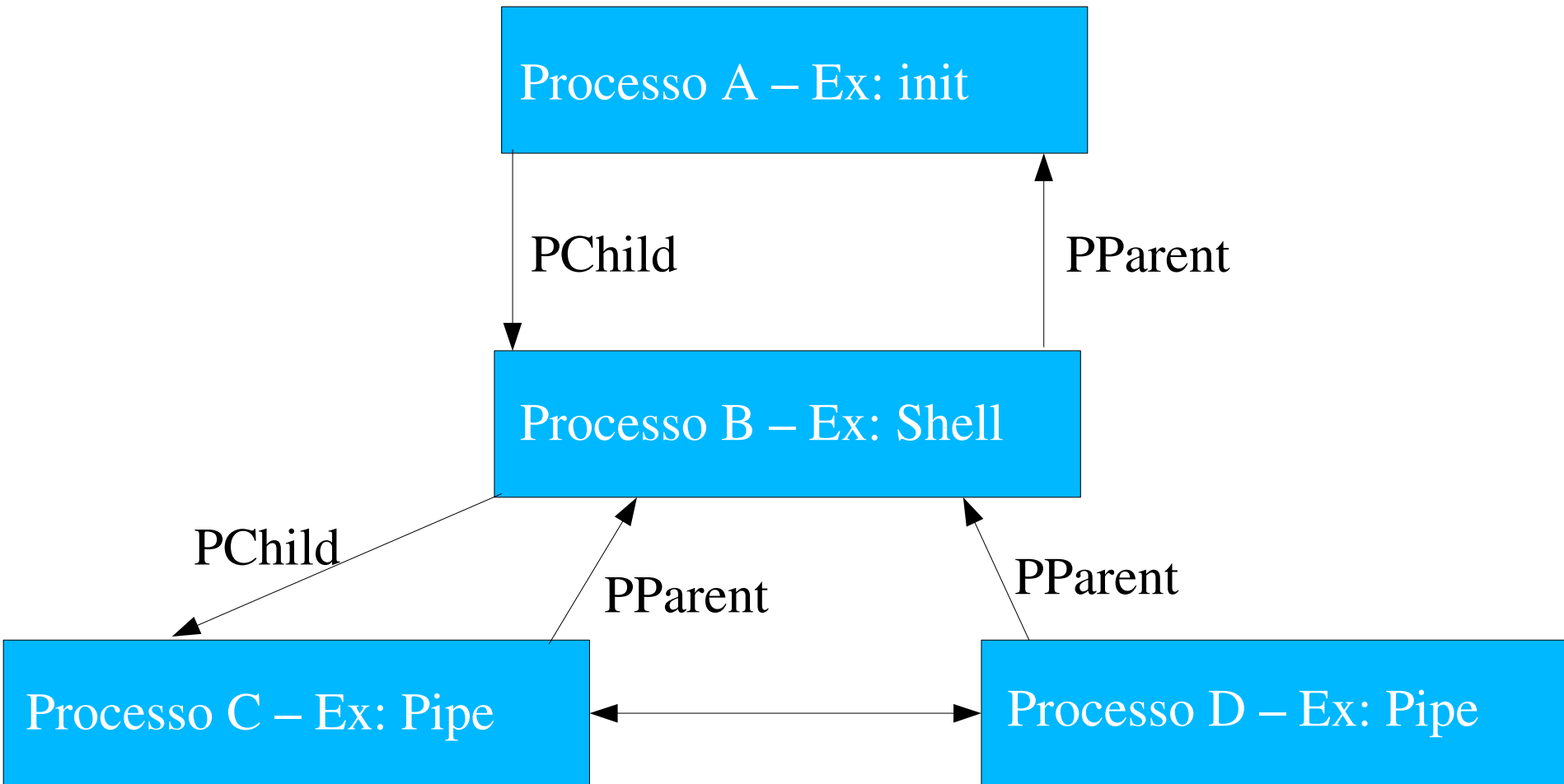
Hierarquia de Processos

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

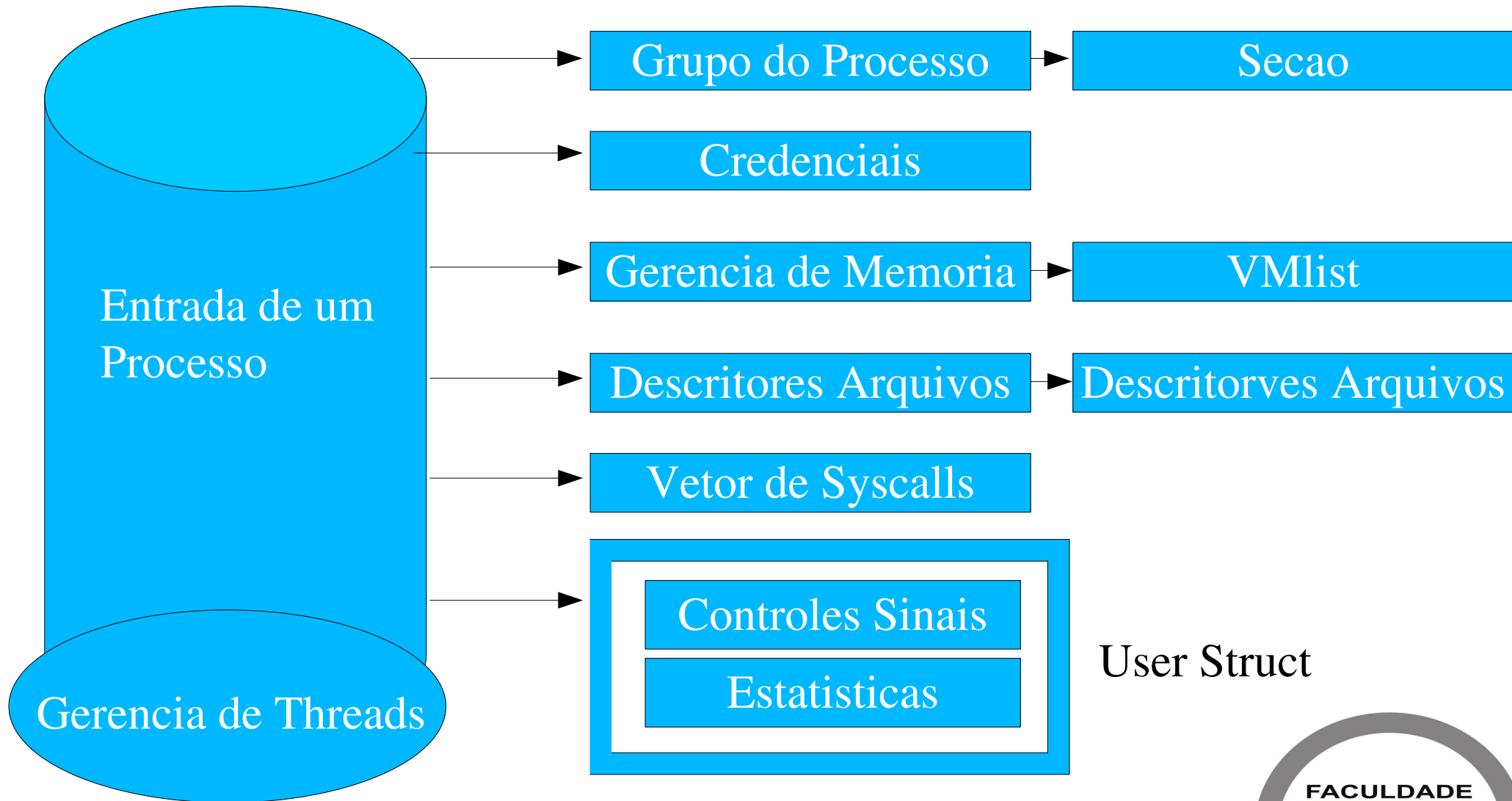
Cada processo possui uma estrutura de memória responsável pelo gerenciamento dos dados referentes a este processo, e conhecida como `task_struct`.



Hierarquia de Processos



Estrutura de Gerencia de Processos



Fork()

A criação de novos processos se dá por uma chamada a `fork()`, que gera um “clone” do processo atual, compartilhando inclusive o espaço de endereçamento (com recurso de COW – copy-on-write, ou seja, se modificado irá gerar uma cópia real) e o `timeslice` (tempo restante de processador).

Tal chamada (`fork()`) irá retornar para o processo pai, o `pid` do novo processo criado (processo filho), e ao processo filho, retornará 0, permitindo assim que no código, faça-se clara a diferenciação entre a execução 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 será então 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).

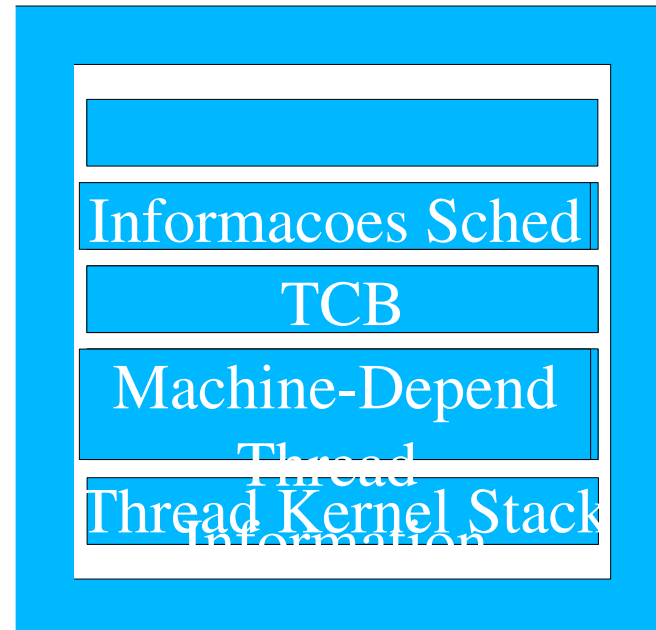
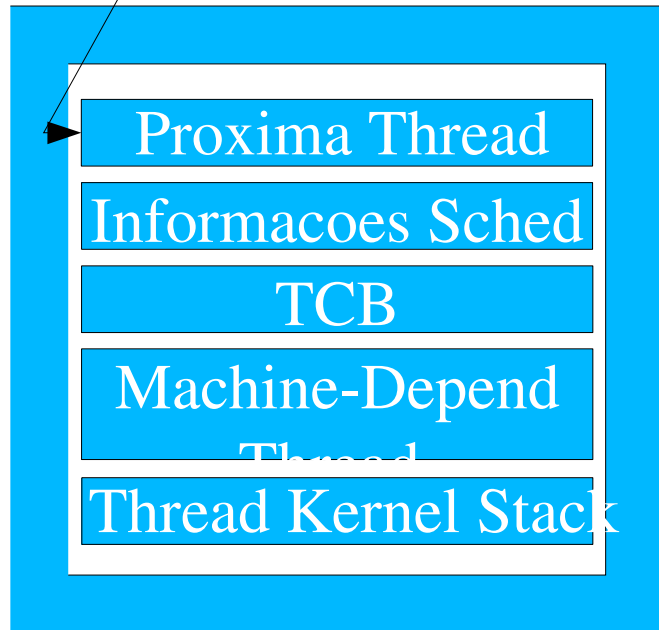
Podem ser gerenciadas de duas maneiras diferentes, via kernel, onde o sistema nativamente suporta threads, ou via bibliotecas, que “simulam” para o programador o ambiente das threads (as bibliotecas na verdade utilizam-se de chamadas ao sistema para gerar em kernel conceitos muito similares que os das 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



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 pendentos, enviados, mascara de sinais)
- Estado do Processo



Estado de Processos

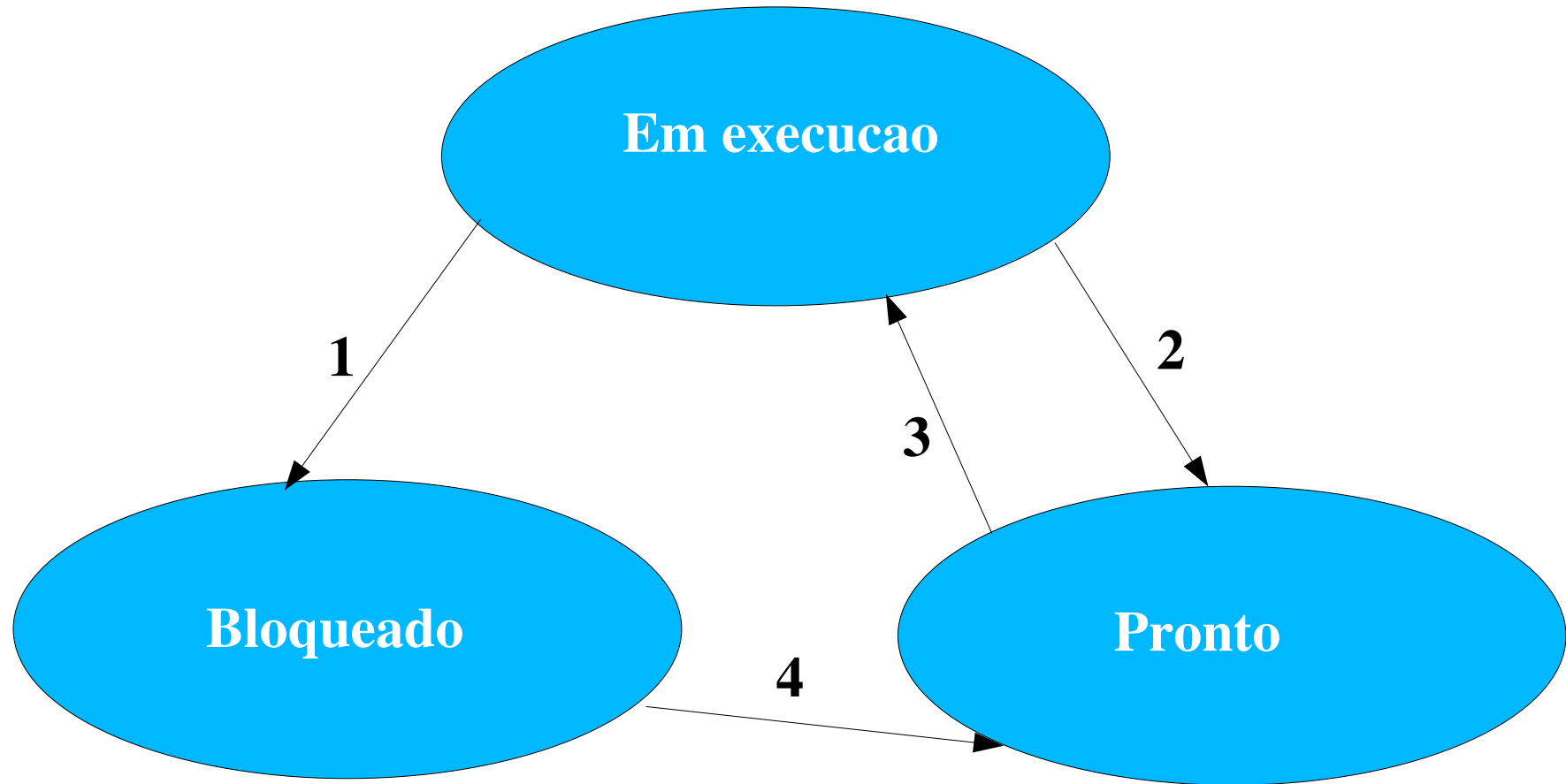
Um processo pode encontrar-se em um dos estados a seguir:

- NEW
- NORMAL
 - Runnable, Sleeping, Stopped
- ZOMBIE

Quando o processo vai ser criado pela syscall `fork()`, ele será marcado como NEW. Após a alocação dos recursos necessários para a execução, o processo será 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 será marcado como ZOMBIE após seu término, enquanto não os recursos alocados e não 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 disponível**



FIM! Será mesmo?

DÚVIDAS?!?

Rodrigo Rubira Branco
rodrigo@kernelhacking.com