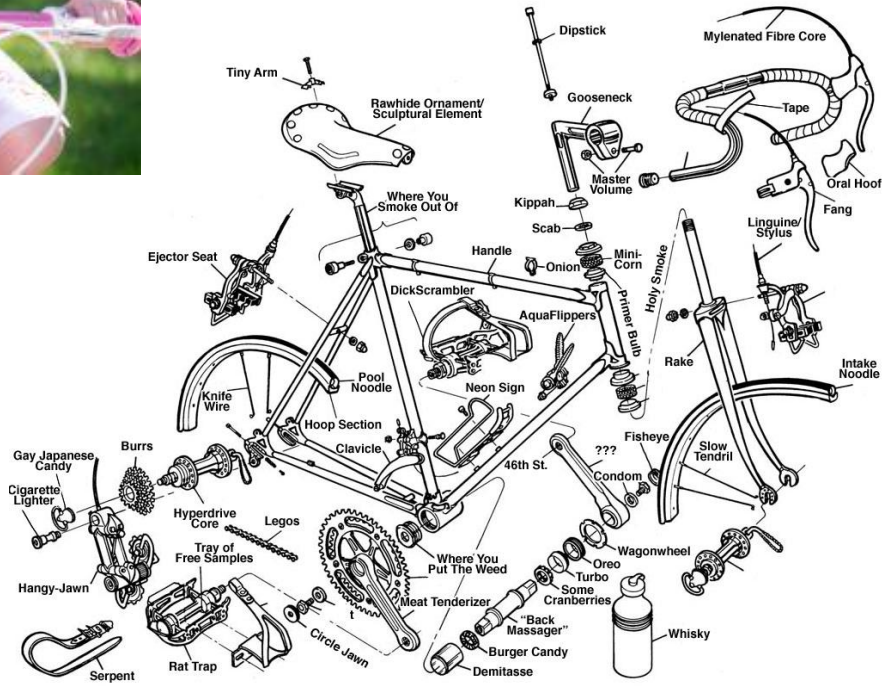
Estrutura software do Núcleo
Boot
Comutação de processos

Sistemas Operativos

O programa num slide

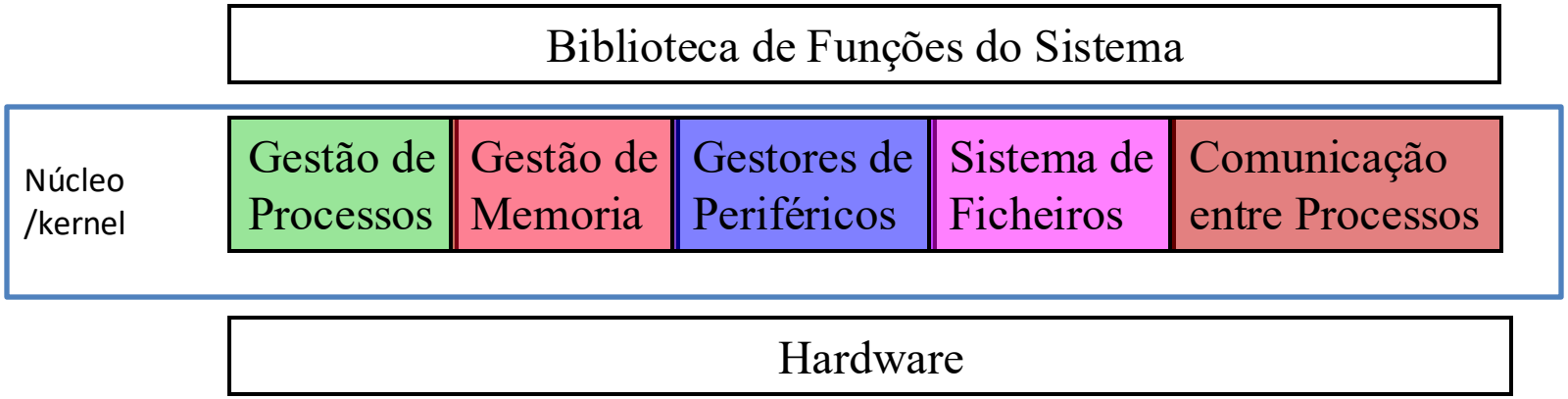


Organização
do
Sistema Operativo

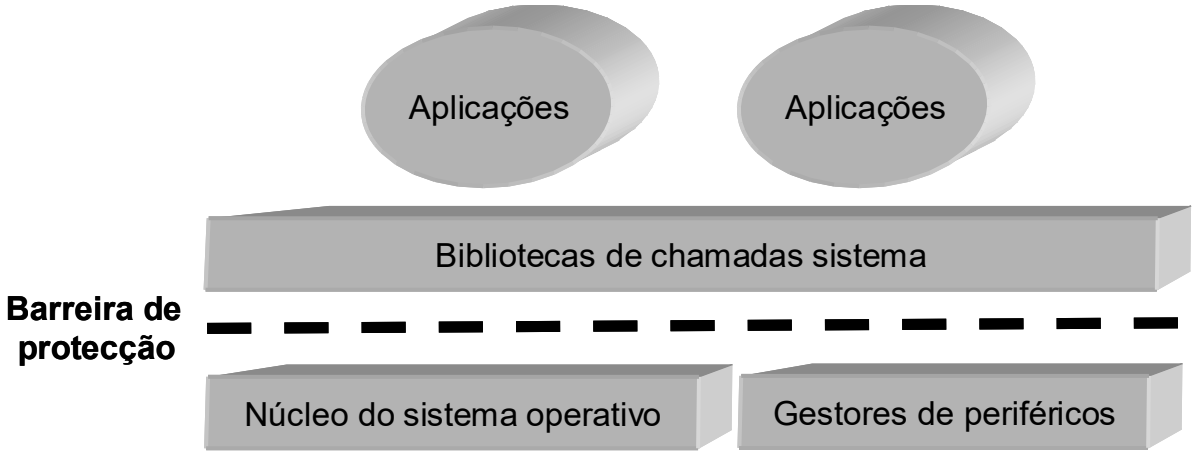
Organização do Sistema Operativo

- Como todos os grandes programas, o sistema operativo procura ter uma organização interna que facilite a sua manutenção e evolução

- Esta organização depende obviamente de cada sistema, mas se consideramos apenas a representação dos grandes módulos será semelhante a que apresentamos na figura

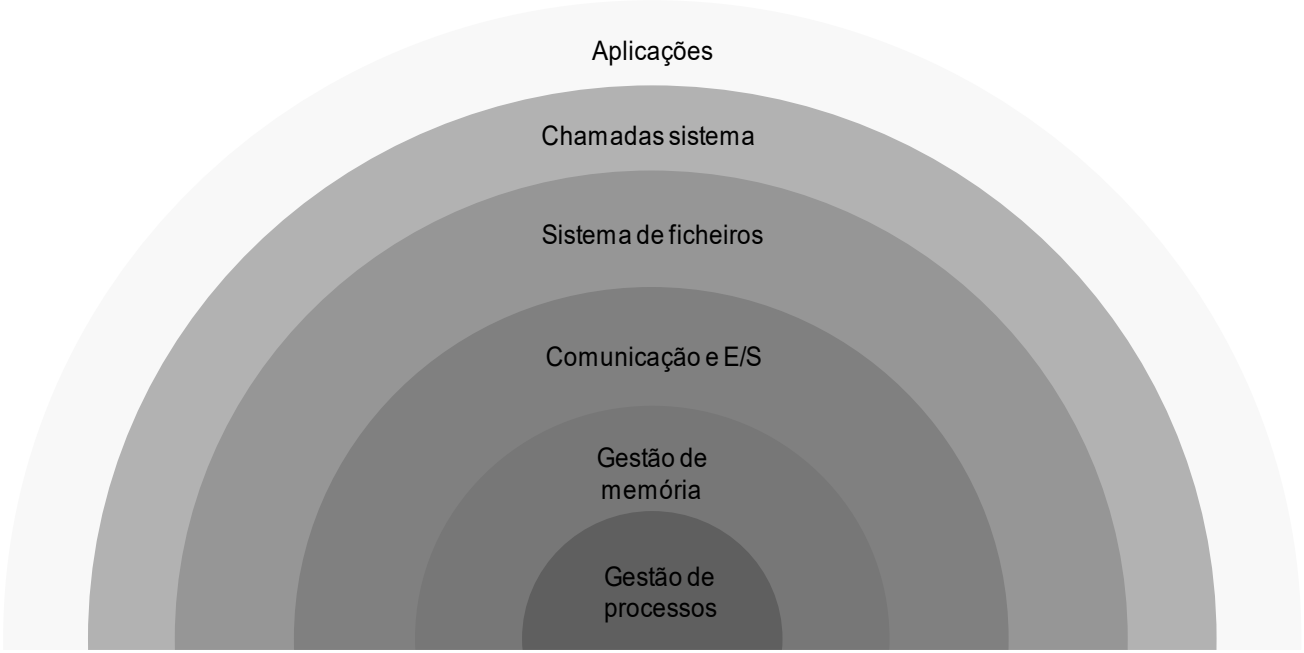


Estrutura Monolítica



- Um único sistema
- Internamente organizado em módulos
- Estruturas de dados globais
- Problema: como dar suporte à evolução
  - Em particular, novos periféricos
- Solução para este caso particular: gestores de dispositivos (*device drivers*)
- Problemas?

Sistemas em Camadas



- Cada camada usa os serviços da camada precedente
- Fácil modificar código de uma camada
- Mecanismos de protecção → maior segurança e robustez
- Influenciou arquitecturas como Intel
- Desvantagem principal?

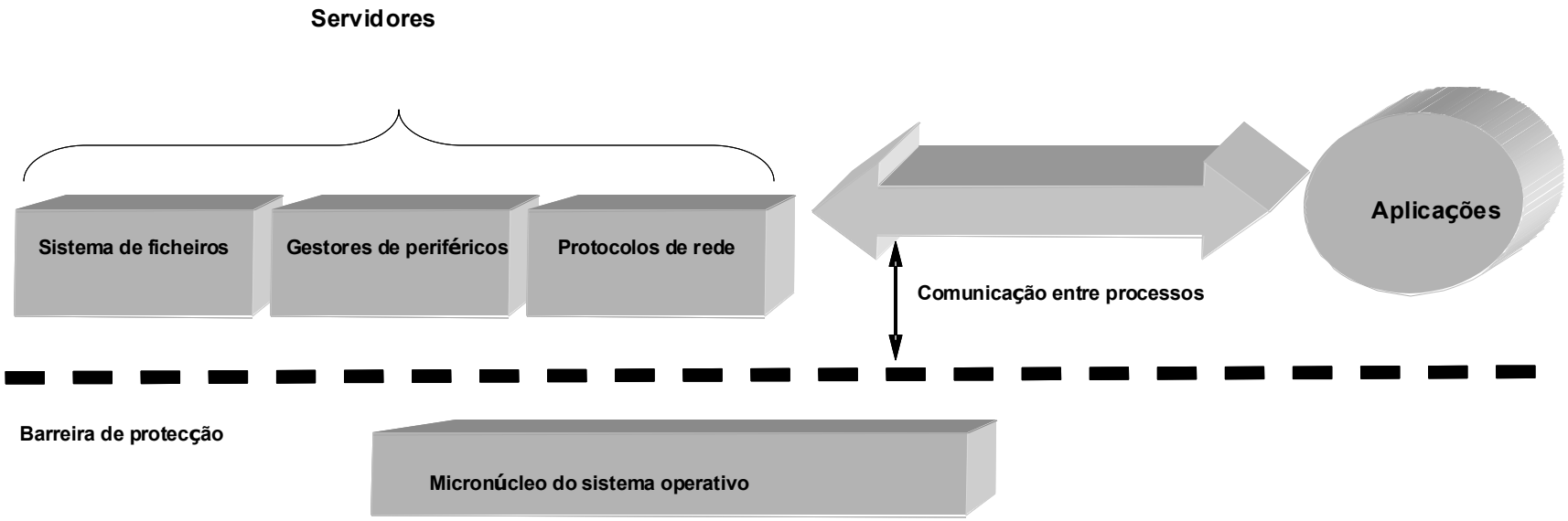
Micro-núcleo

- Propostas de investigação → separação entre:

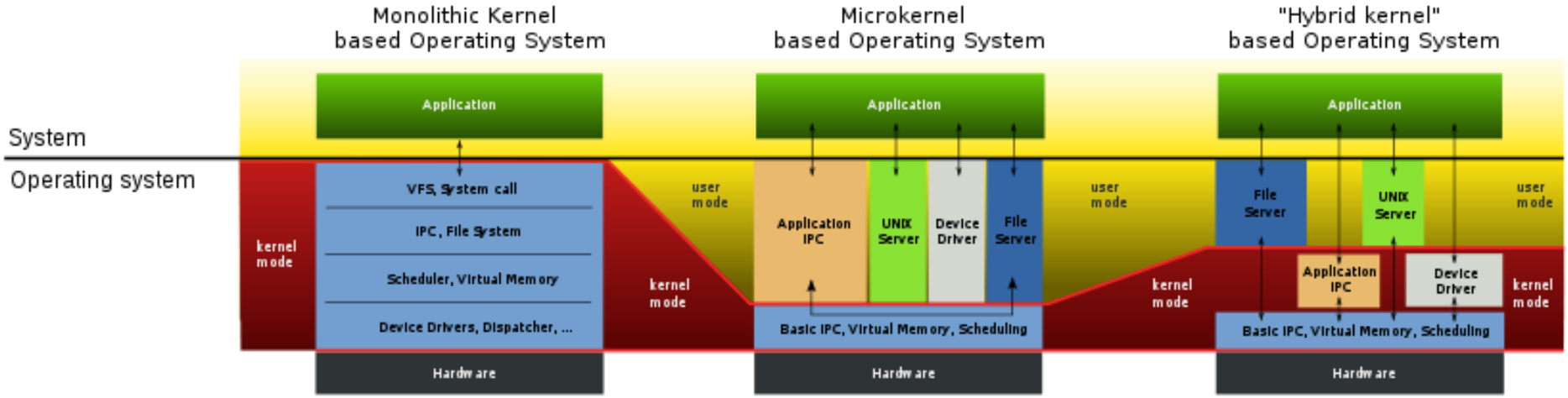
- Um micro-núcleo de reduzidas dimensões e que só continha o essencial do sistema operativo:
  - Gestão de fluxos de execução - threads
  - Gestão dos espaços de endereçamento
  - Comunicação entre processos
  - Gestão das interrupções

- Servidores sistema que executavam em processos independentes a restante funcionalidade:
  - Gestão de processos
  - Memória virtual
  - Device drivers
  - Sistema de ficheiro

Micro-Núcleo



Micro-Núcleo vs Monolítico



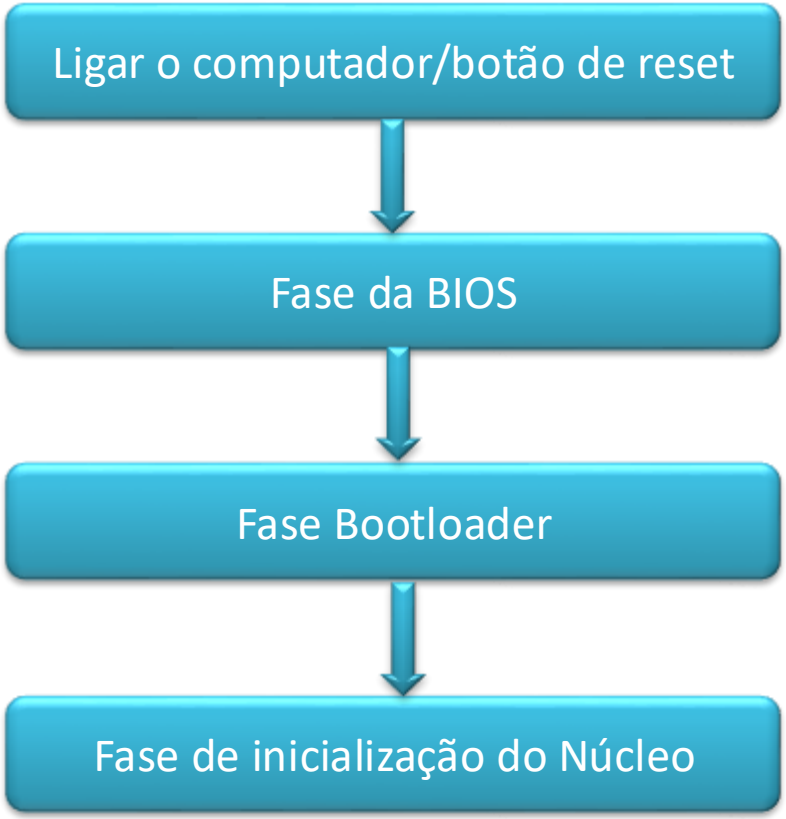
O que acontece desde que ligo a máquina até ao Sistema Operativo estar a funcionar?

Booting

- Ficou na tradição o nome *booting* que originalmente se relaciona com as presilhas (*bootstraps*) que ajudam a calçar as botas

- A expressão "*to pull oneself by one´s bootstraps*" era sinónimo de tarefa difícil efetuada sem ajuda externa

Ciclo de boot



Boot do núcleo de um sistema operativo

- Quando se inicia a máquina, *PC* aponta para um programa na *Boot ROM*
  - Nos computadores pessoais, o programa na Boot ROM chama-se BIOS (basic input/output system)
- O BIOS:
  - Copia bloco de código do disco (ou flash RAM, etc.) para RAM
  - Salta para a primeira instrução desse programa, chamado *bootloader*

Boot do núcleo de um sistema operativo (II)

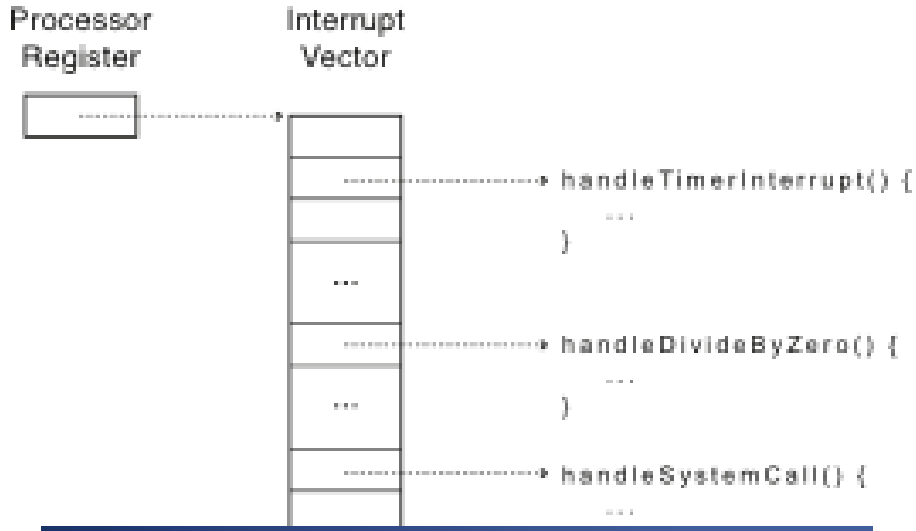
- *Bootloader*:
  - carrega o programa do núcleo em RAM
  - salta para rotina de inicialização do núcleo

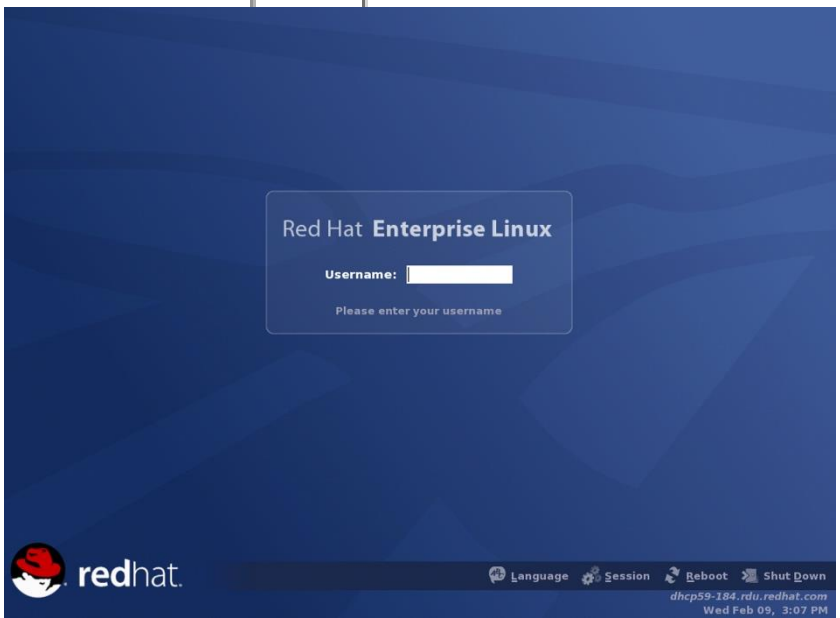
Boot do núcleo de um sistema operativo (III)

- Em Linux é conhecido o *Grand Unified Bootloader* – GRUB que foi desenvolvido para permitir carregar diversas versões de Linux

Boot do núcleo de um sistema operativo (IV)

- Inicialização do Núcleo:
  - Inicializa as suas estruturas de dados
  - Copia rotinas de tratamento de cada interrupção para RAM
  - Preenche tabela de interrupções (em RAM)
  - Lança os processos iniciais do sistema, incluindo o processo de login





Boot do núcleo de um sistema operativo (V)

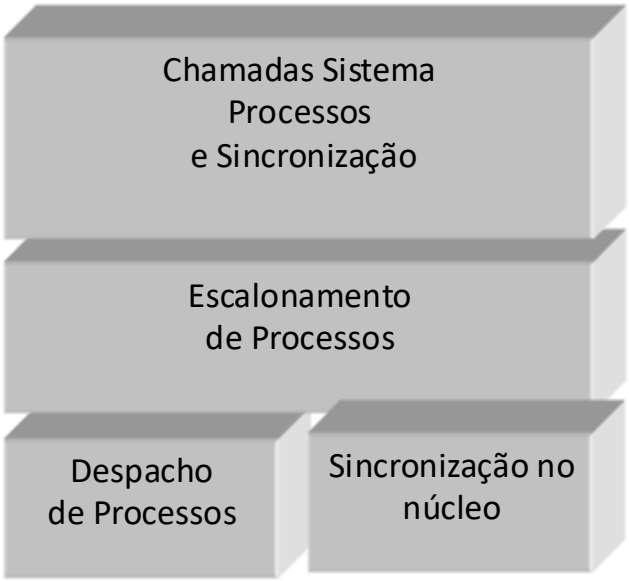
- Finalmente:
  - O núcleo consiste num conjunto de rotinas de tratamento de interrupções
  - Normalmente não estão em execução
  - São ativadas sempre que surjam interrupções

Gestão de Processos

Núcleo do Sistema Operativo

Gestor de Processos

- Entidade do núcleo responsável por suportar a execução dos processos
  - Multiplexagem do Processador
    - Despacho – efetua a transferência de controlo entre dois processos
    - Escalonamento – otimiza a gestão do processador
  - Sincronização no núcleo
  - Implementação das funções sistema relacionadas com os processos e sincronização: criação, sincronização, informação, eliminação



Representação dos Processos (e Tarefas)

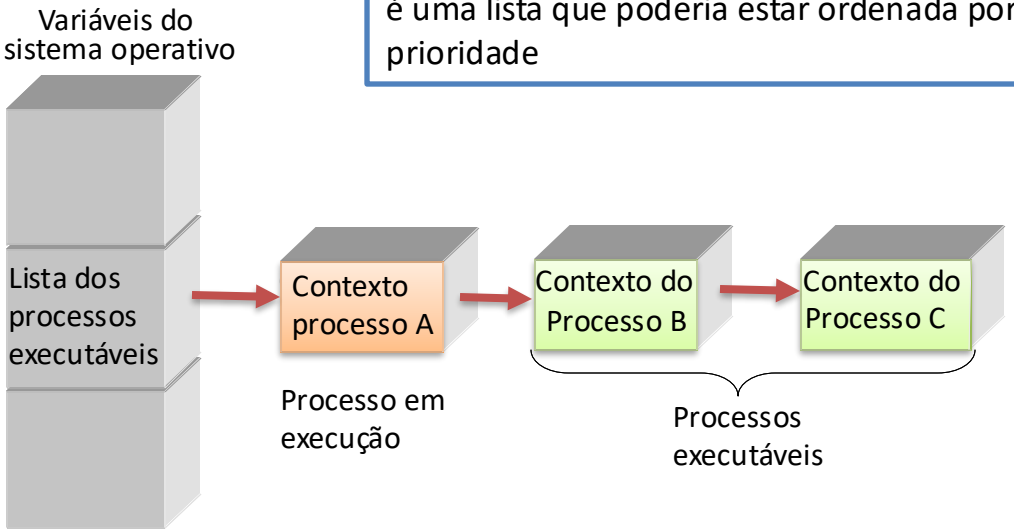
Contexto: representação de um processo no núcleo

- Contexto de hardware
  - Registos do processador (acumulador, uso geral, contador de programa, stack pointer, flags de estado do CPU)
  - Registos da unidade de gestão de memória

- Contexto de software
  - Identificação (processo, utilizador, grupo)
  - Prioridade
  - Estado do processo
  - Outras informações (periféricos em uso, ficheiros abertos, directório por omissão, programa em execução, contabilização de recursos, signals pendentes, etc.)

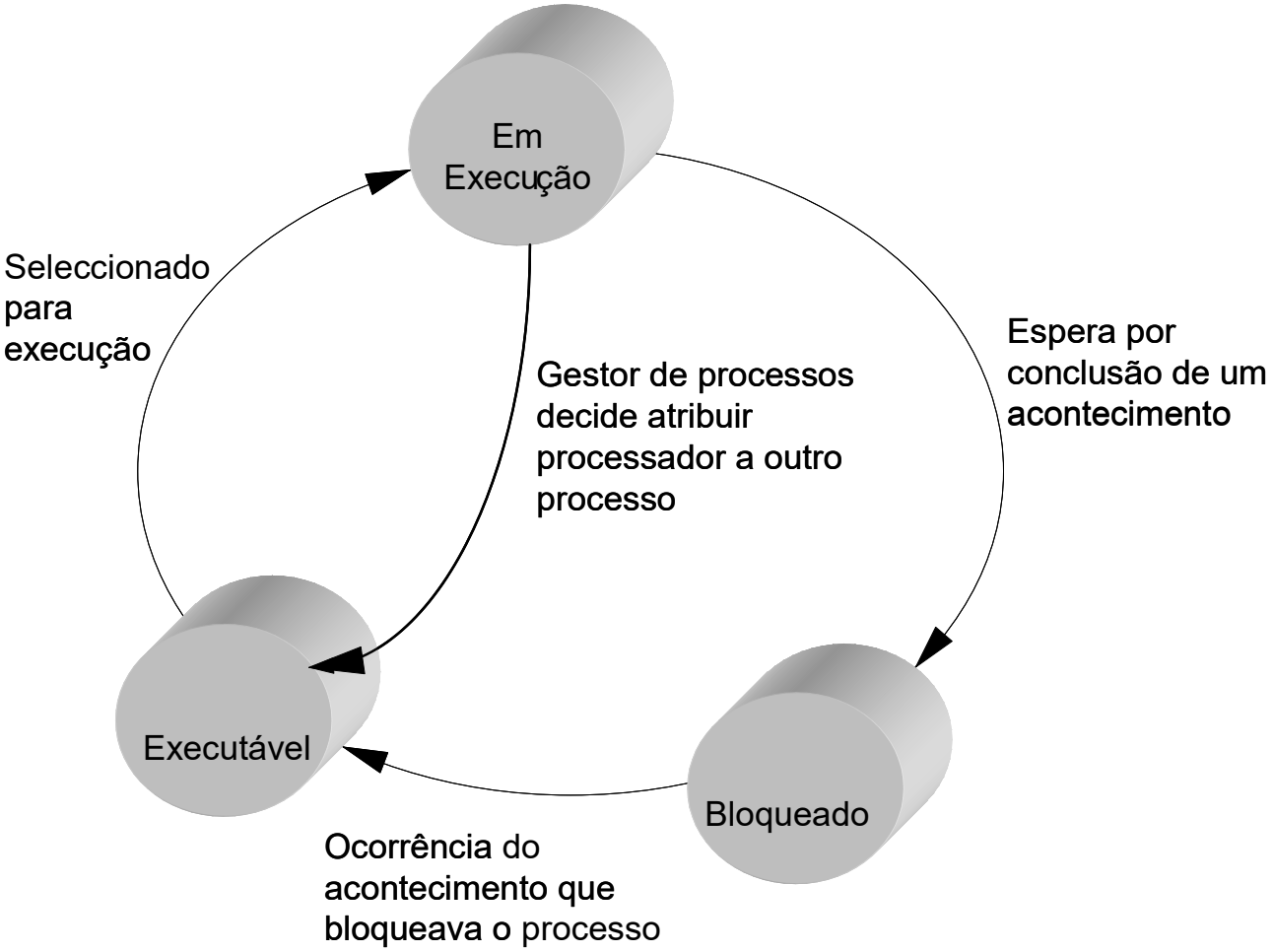
Lista dos Processos Executáveis (*)

A maneira mais simples de suportar o contexto dos processos é uma lista que poderia estar ordenada por um critério de prioridade



(*) Veremos que normalmente se usam estruturas mais complexas

Diagrama de Estado dos Processos



Comutação de Processos/Tarefas

Despacho ou *low-level scheduling*

Despacho

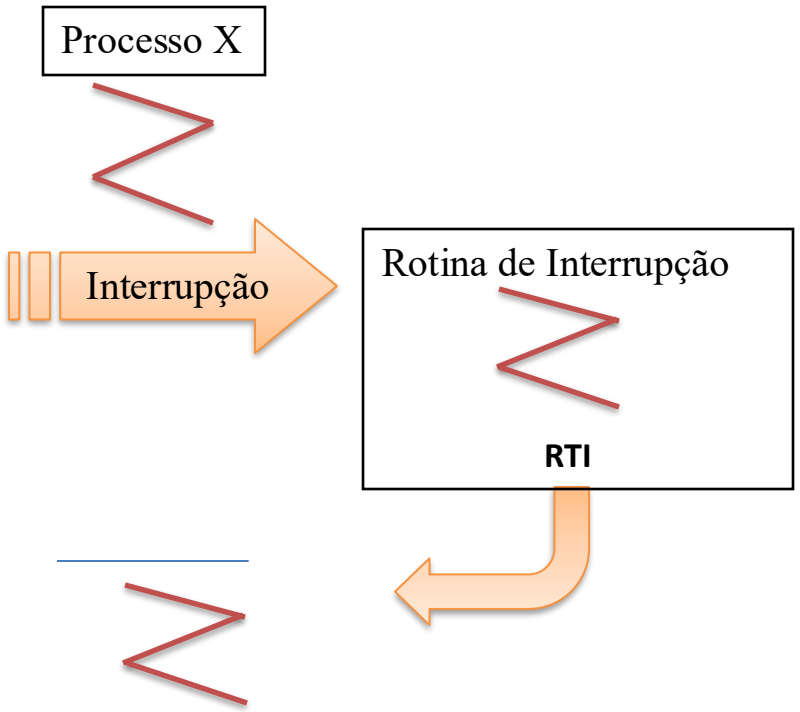
- A função do despacho é comutar o processador sempre que lhe seja indicado para o fazer.

- Funcionalidade:
  - copia o contexto hardware do processo em execução para o respetivo descritor (entrada na lista de processos)
  - escolhe o processo mais prioritário entre os executáveis
  - carrega o seu contexto hardware no processador
  - transfere o controlo para o novo processo
    - coloca *program counter* guardado no contexto do novo processo na pilha
    - *return from interrupt* (RTI) é "enganado"

Quando é que o Despacho é chamado?

- Para o Despacho ser chamado é necessário que algo mude no sistema.
- A forma como as mudanças são assinaladas são através das exceções ou das interrupções.
- Uma fonte importante de alteração do estado do sistema são as chamadas sistema que como vimos são desencadeadas por *software interrupts*.

Rotina de Interrupção



Quando se dá a aceitação da interrupção o *Program Counter* e as flags de estado do processador
são salvaguardado na pilha

A instrução de RTI repõe o *Program Counter* e o estado fazendo com que a execução continue na instrução seguinte à que se executava quando se deu a interrupção

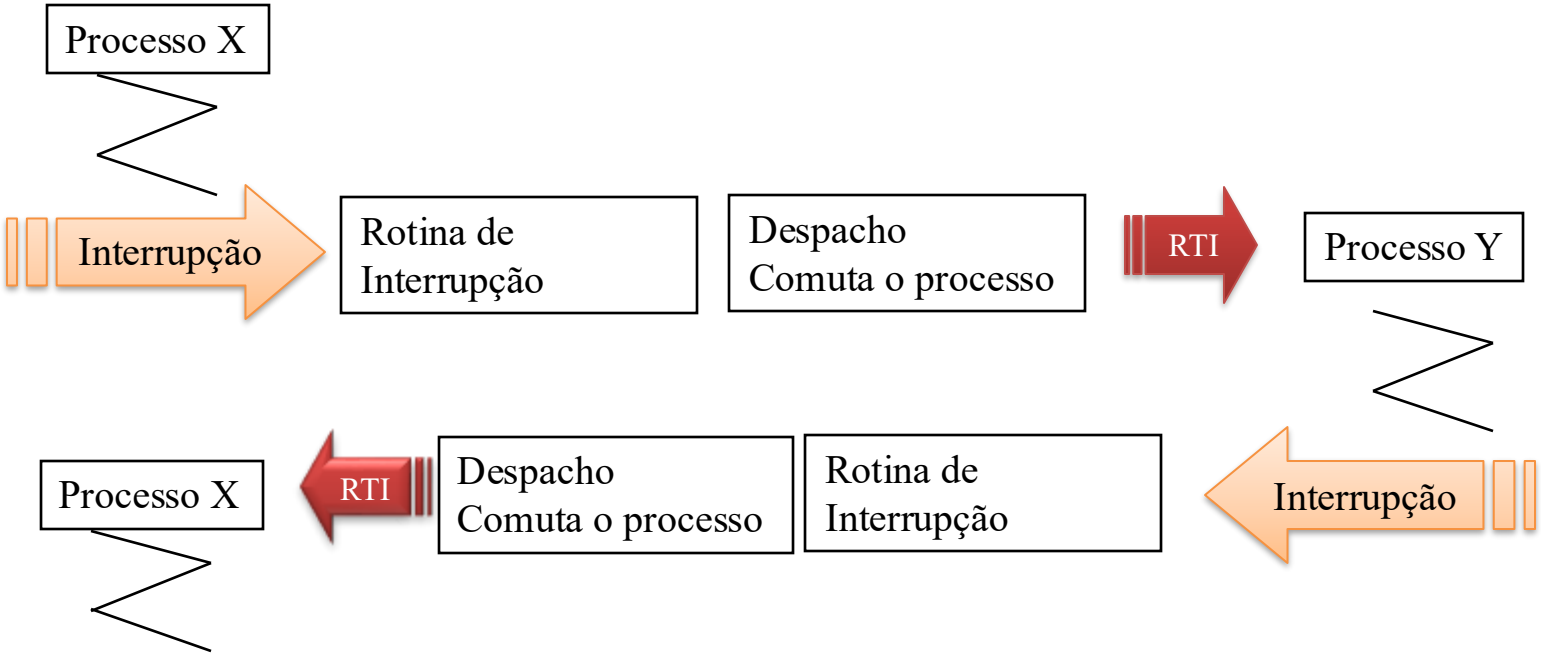
Rotina de Serviço da Interrupção

- Manter a estrutura normal da rotina de interrupção
  - Corre o código específico à interrupção, possivelmente alterando o estado dos processos

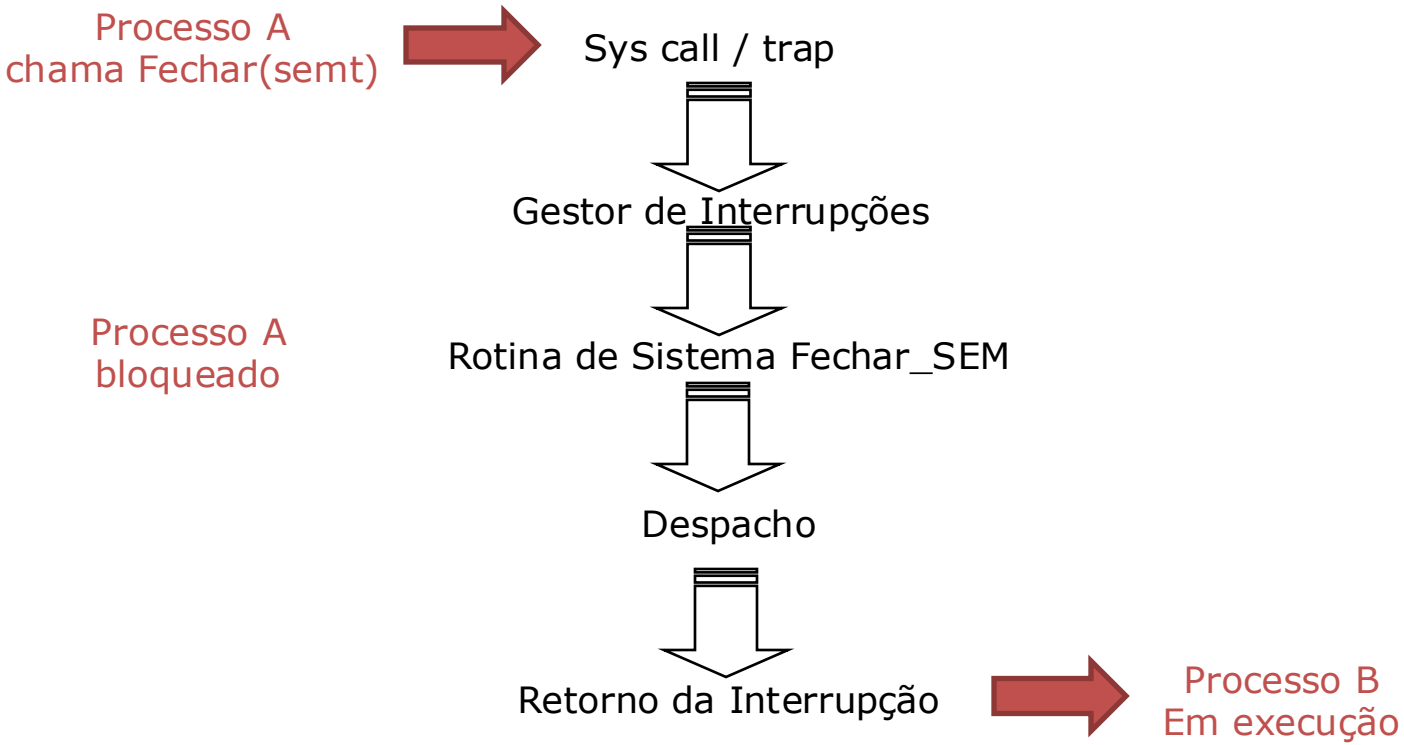
- ***Se alterar o estado dos processos invoca o Despacho*** para eventualmente escolher outro processo para execução

- O Despacho coloca na pilha o *program counter* (PC) e as *flags* de estado do processo que se irá executar e executa *return from interrupt* (RTI)

Despacho


Processo X
Interrupção
Rotina de Interrupção
Despacho Comuta o processo
RTI
Processo Y
Interrupção
Rotina de Interrupção
Despacho Comuta o processo
RTI
Processo X

Despacho: exemplo



E caso um processo não largue o processador?

- Interrupções de temporização
  - Permitem ao núcleo interromper um processo que esteja em execução há algum tempo
  - Oportunidade para o núcleo tirar esse processo de execução e executar outro processo
  - Conceito de uma curto período de tempo (*time-slice)* atribuído a um processo

Modo Núcleo e Rotina de Interrupção

- A rotina de interrupção tem de se executar em modo núcleo

- A mudança de modo corresponde a:
  - mudança para o modo de proteção mais privilegiado do processador
  - mudança do espaço de endereçamento do processo utilizador para o espaço de endereçamento do núcleo

- Mas é também necessário mudar a pilha uma vez que não se deveria usar a pilha do processo em modo utilizador para executar esta função do núcleo

- A pilha núcleo:
  - É usada a partir do instante em que o processo muda de modo utilizador para modo núcleo

Pilha Utilizador/Pilha Núcleo

- Porque são necessárias duas pilhas?
  - A pilha em modo utilizador é a base da computação dos programas
  - A pilha em modo núcleo tem de ser diferente para garantir a estanquicidade de informação entre a atividade das funções do núcleo e do utilizador
  - A pilha em modo núcleo está vazia quando o processo passa para modo utilizador e contem o contexto de invocação das funções do núcleo quando está em modo núcleo.
  - Como o processo se pode bloquear no núcleo tem de ser uma por processo para permitir guardar de forma independente o contexto

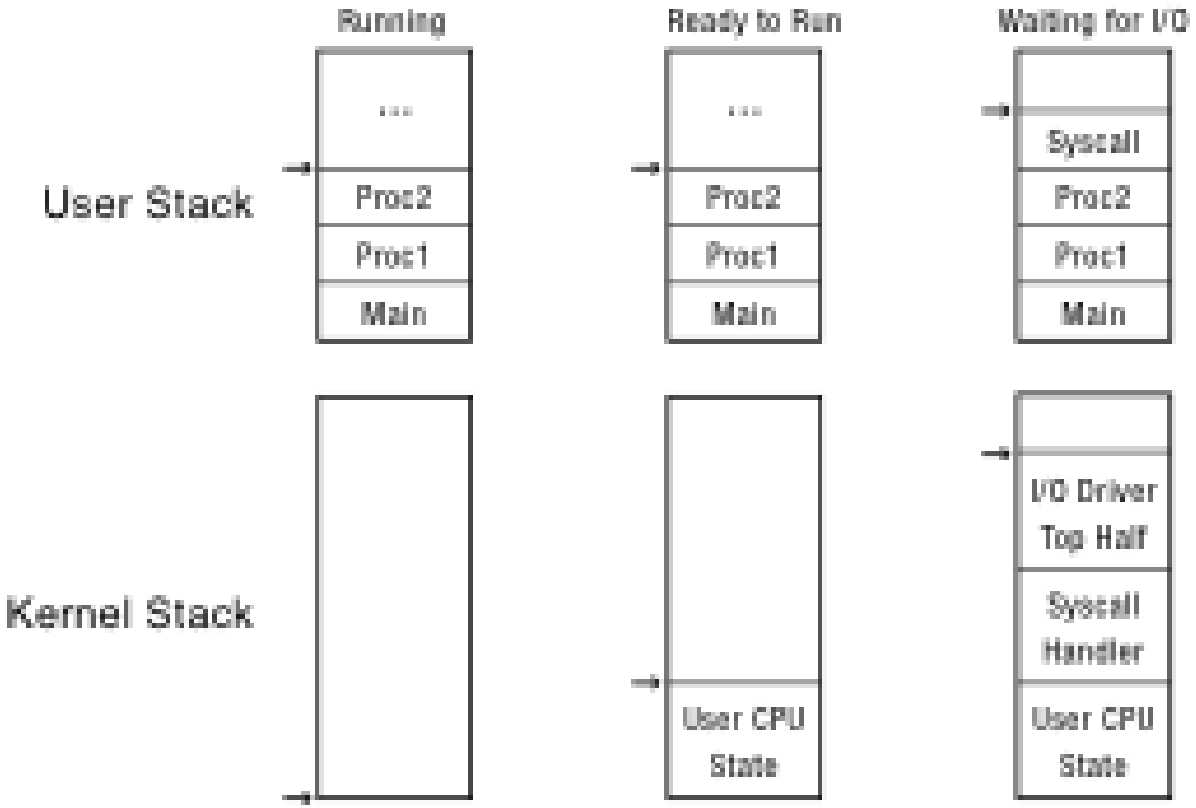
Porque duas pilhas? Um exemplo concreto

- Num processo multi-tarefa, uma tarefa fez chamada sistema
- Quando a rotina núcleo se executa, coloca variáveis locais das funções núcleo na pilha
- Problema: outras tarefas do mesmo processo podem estar a correr noutros processadores
  - Logo podem aceder e corromper a pilha da tarefa que fez chamada sistema
  - Corrompendo as variáveis locais usadas pelas rotinas do núcleo!
- Duas pilhas (utilizador e núcleo) por cada processo evitam estes problemas

Pilha Núcleo

- A rotina de tratamento de uma interrupção começa por mudar de pilha (alterando o *stack pointer*) para o da pilha núcleo
- Copia para esta pilha a informação que lhe permite retornar ao programa utilizador, pelo menos o *program counter* e as *flags* de estado do processador
- Enquanto estiver em modo núcleo passa a utilizar esta pilha
- Um especto interessante é se surgirem entretanto interrupções do *hardware* que também tem de salvaguardar registos na pilha
  - é um decisão de desenho ter uma pilha separada ou utilizar como "hospedeiro" a pilha núcleo do processo corrente (solução do Unix/Linux)

Duas pilhas: utilizador e núcleo



Utilização da Pilha Núcleo