Sistemas Operacionais Embarcados

Revisão de Conceitos sobre Sistemas Operacionais

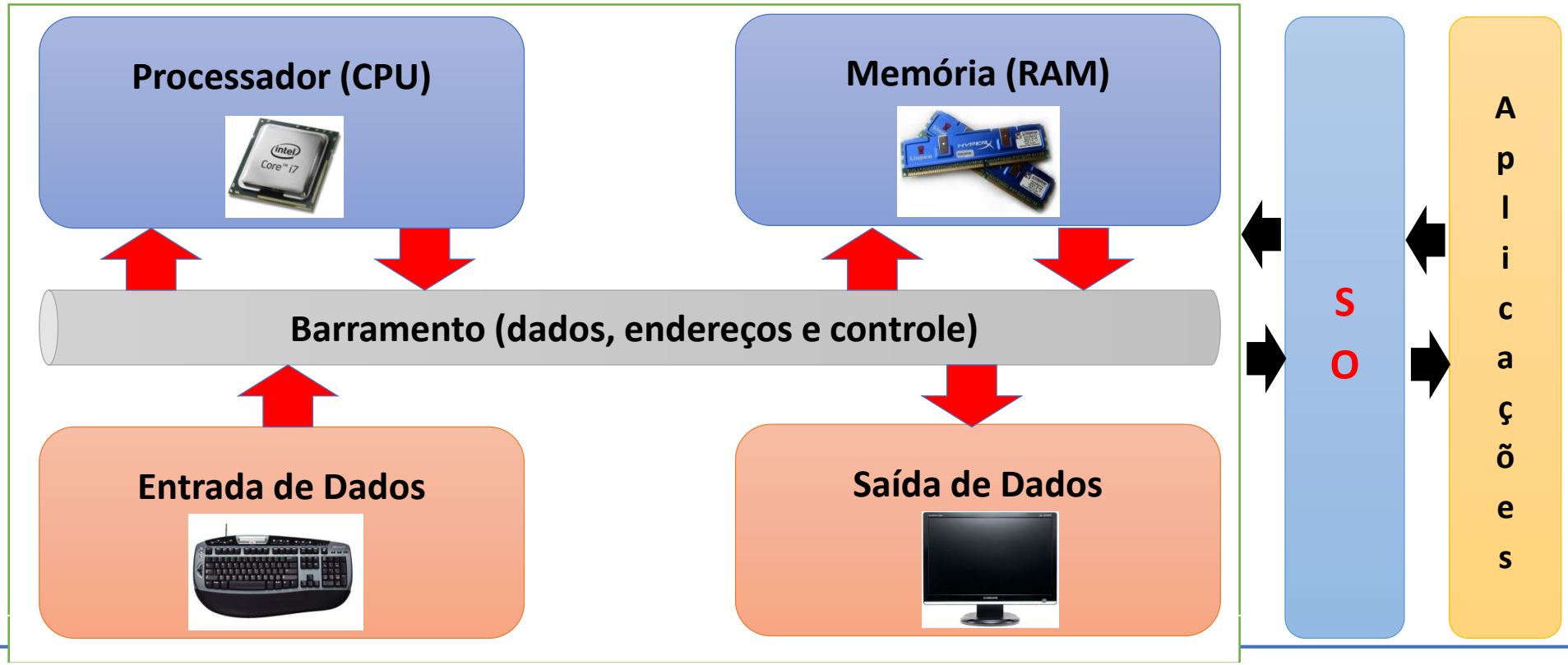
Prof. Anderson Luiz Fernandes Perez
Departamento de Computação
Centro de Ciências, Tecnologias e Saúde
Universidade Federal de Santa Catarina
Campus Araranguá

Conteúdo

- Definição de Sistema Operacional
- Responsabilidade de um Sistema Operacional
- Arquitetura Interna de um Sistema Operacional
- Chamadas de Sistema
- Definição de Processo
- Troca de Contexto
- Escalonamento de Processos
- Sincronização de Processos
- Comunicação entre Processos
- Gerenciamento de Memória
- Gerenciamento de Dispositivos de Entrada e Saída de Dados

Definição de Sistema Operacional

- Sistema Computacional



Definição de Sistema Operacional

- O que é um SO?
  - Um SO é um software que controla diretamente o hardware e fornece uma infraestrutura para outros softwares.
  - É o software de base da qual a maioria das aplicações depende.
  - Gerencia o compartilhamento de recursos entre entidades concorrentes.
  - Fornece vários serviços comuns que tornam as aplicações mais fáceis de escrever.

Definição de Sistema Operacional

- O que é um SO?

Um Sistema Operacional é composto **de um ou mais programas** que **fornece um conjunto de serviços**, o qual cria uma **interface entre aplicações e o hardware** do computador e que **aloca e gerencia recursos** compartilhados **entre múltiplos processos**.

Responsabilidade de um Sistema Operacional

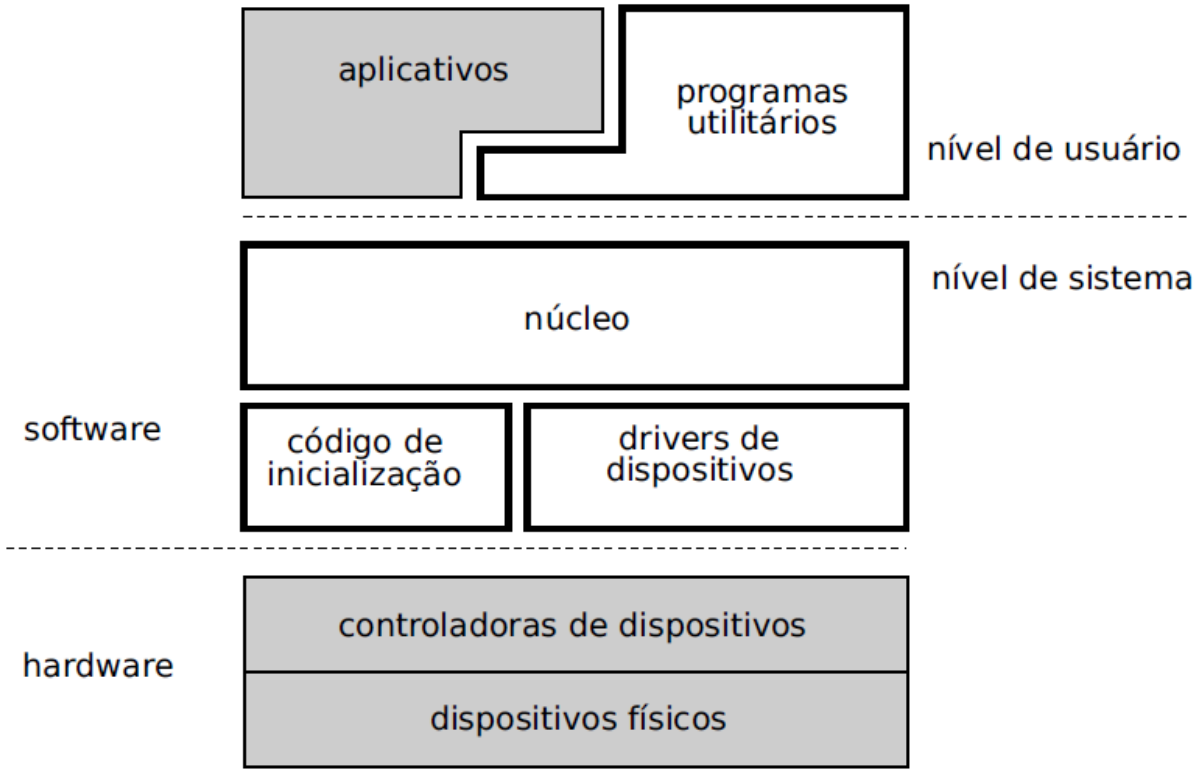
- Um sistema operacional deve lidar com um série de recursos e para cada um aplicar técnicas de gerenciamento específicas.
- São responsabilidade de um SO:
  - Gerenciar processos;
  - Gerenciar a memória;
  - Gerenciar os dispositivos de E/S;
  - Prover um sistema de arquivos;
  - Controlar a segurança do sistema computacional;
  - Prover um mecanismo de comunicação (rede);
  - Prover um interface com o usuário.

Arquitetura Interna de um Sistema Operacional

- Um SO é composto por diversos componentes com objetivos e funcionalidades complementares.
- Os componentes mais comuns presentes em um SO são:
  - Kernel (núcleo)
  - Drivers
  - Código de inicialização
  - Programas utilitários

Arquitetura Interna de um Sistema Operacional

- Integração entre os Componentes de um SO



Fonte: Maziero, 2010.

Chamadas de Sistema

- Chamadas de sistema ou chamadas ao sistema são rotinas implementadas no SO com para a execução de funções específicas.
- Essas rotinas são executadas pelas aplicações de usuário quando estas necessitam de algum serviço do SO.
- Todo sistema operacional possui uma API (*Application Programmer Interface*) que implementa um conjunto de chamadas de sistemas.

Definição de Processo

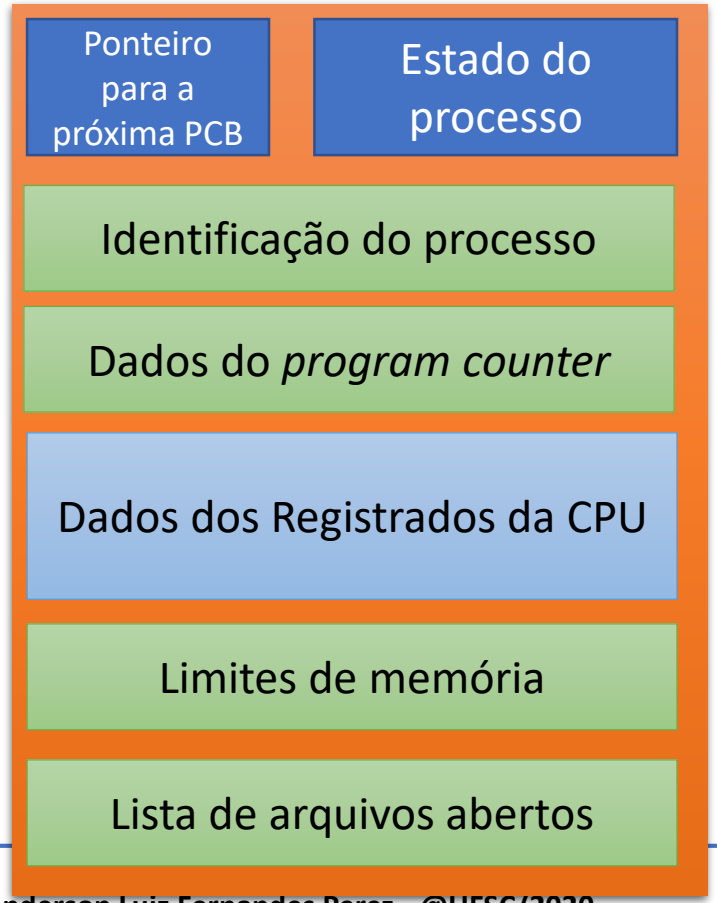
- Um processo é uma entidade ativa no sistema e representa um programa em execução.
- Um processo é representado no SO por uma estrutura de dados chamada PCB (*Process Control Block*/Bloco de Controle do Processo).
- A PCB possui informações sobre o contexto de execução do processo.

Definição de Processo

- Informações Contidas na PCB (*Process Control Block*)
  - Identificação do processo;
  - Estado de execução;
  - Dados do contador de programas (*Program Counter*);
  - Dados dos registrados da CPU;
  - Informações para o escalonamento de CPU, por exemplo prioridade;
  - Informações sobre memória;
  - Informações de contabilidade;
  - Informações sobre as operações de E/S (status das informações de E/S).

Definição de Processo

- Informações Contidas na PCB (*Process Control Block*)

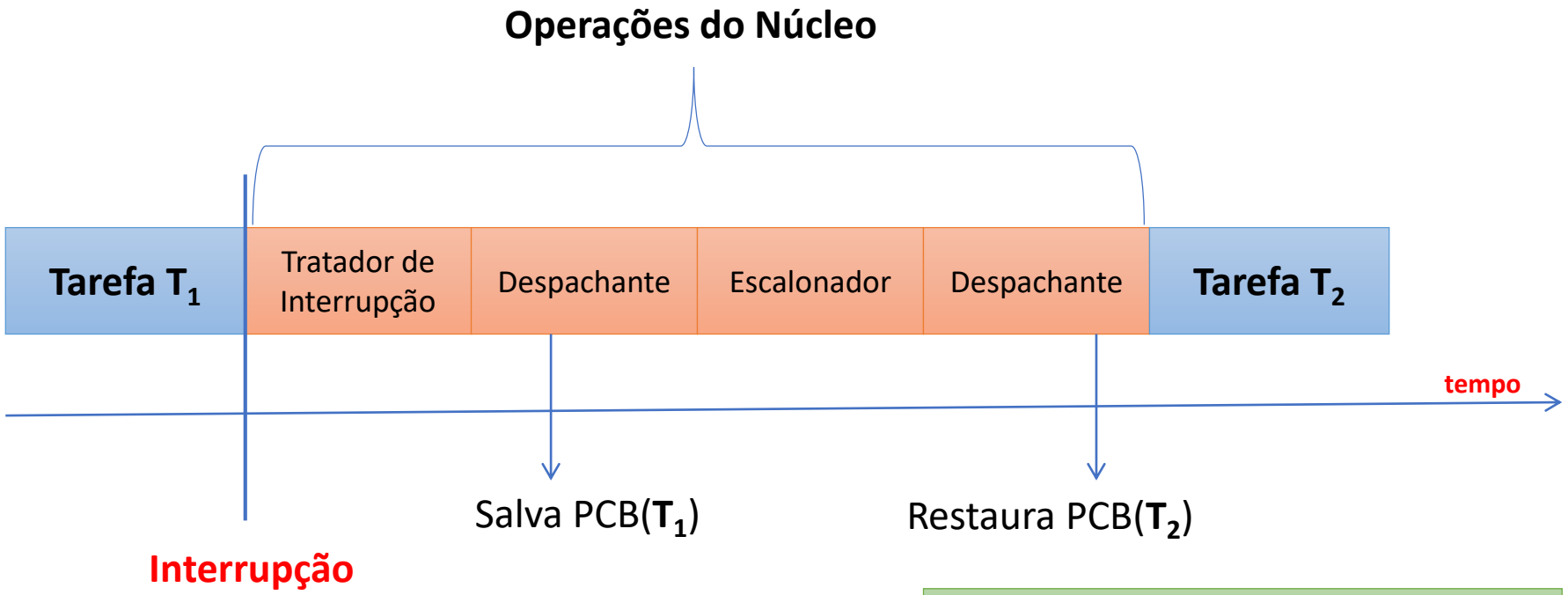


Troca de Contexto

- A troca de uma tarefa (processo ou thread) em execução, ou seja que está com a posse do processador, por outra, é definida como troca de contexto.
- Os dados de contexto de uma tarefa são armazenados em sua PCB (*Process Control Block*).
- A troca de contexto é realizada pelo despachante (do Inglês *dispatcher*) e a escolha de qual tarefa deverá assumir o processador é realizada pelo algoritmo de escalonamento de tarefas.

Troca de Contexto

- Passos de uma Troca de Contexto



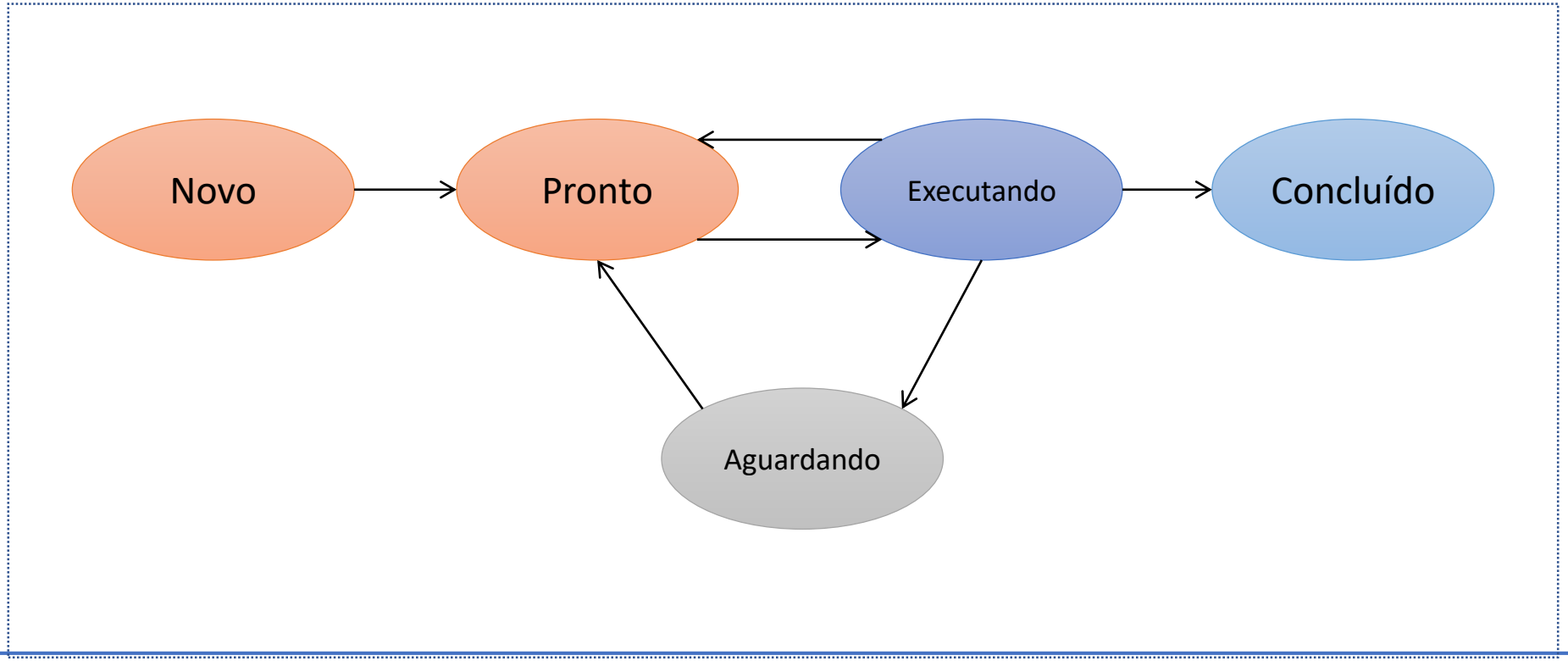
Alguns autores descrevem a PCB como TCB (*Task Control Block*).

Troca de Contexto

- Estados de uma Tarefa (processo ou thread)
  - Uma tarefa pode estar em um dos seguintes estados durante sua execução:
    - **Novo** – a tarefa acabou de ser criada;
    - **Pronto** – a tarefa está pronta para ser executada. Está na fila de tarefas aptas;
    - **Executando** – a tarefa está em execução;
    - **Aguardando** – a tarefa está aguardando um evento;
    - **Concluído** – a tarefa concluiu sua execução.

Troca de Contexto

- Diagrama de Troca de Contexto



Troca de Contexto

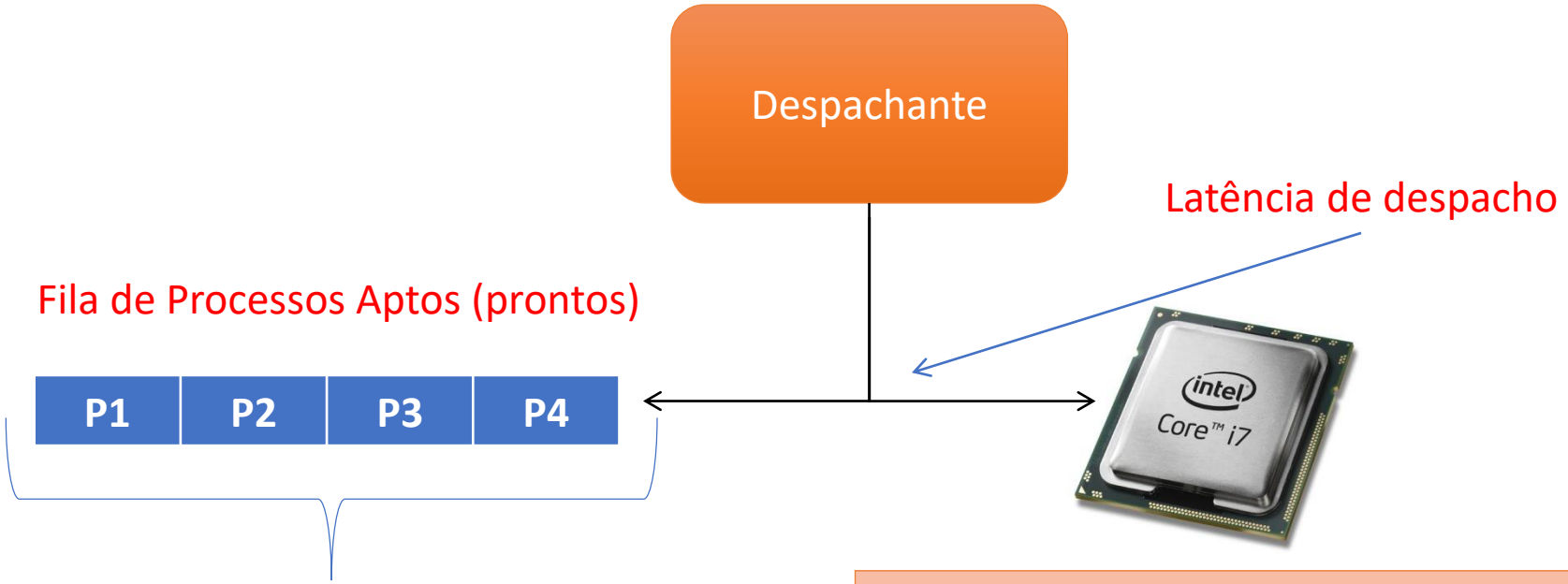
- A troca de contexto pode ocorrer quando:
  - O tempo de execução do processo atual se esgotou;
  - Ocorrer uma interrupção de hardware;
  - Ocorrer uma interrupção de software (execução de uma chamada de sistema – conhecida como *trap*);
  - Ocorrer um erro de execução.

Escalonamento de Processos

- A CPU é um dos principais recursos do computador, desta forma é necessário mantê-la ocupada o maior tempo possível.
- Em um sistema multiprogramado existe a necessidade de selecionar qual e por quanto tempo um determinado programa ocupará a CPU.
- Processos em execução alternam entre picos de CPU e picos de E/S.

Escalonamento de Processos

- Esquema do Escalonamento de Processos



Observação: a fila de processos aptos não é necessariamente uma estrutura de dados do tipo FILA, pode ser implementada como lista encadeada ou árvore.

Escalonamento de Processos

- Algoritmo Baseado em Prioridades
  - A cada processo é associada um número que indica a prioridade.
  - Processos com prioridade maior tem preferência de execução sobre os demais.
  - A prioridade pode ser definida da seguinte forma:
    - Número menor – prioridade maior
    - Número maior – prioridade menor
    - Ou
    - Número maior – prioridade maior
    - Número menor – prioridade menor
  - Geralmente se utiliza números baixos (menores) para indicar maior prioridade.
  - Esse algoritmo pode ser preemptivo ou não preemptivo.

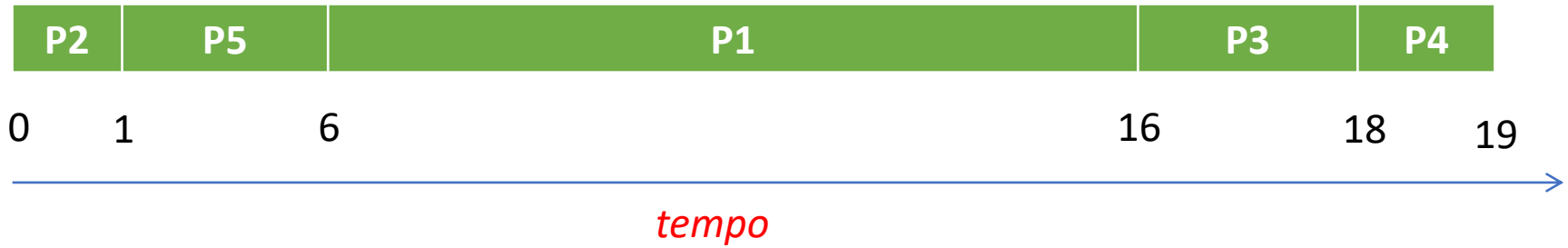
Escalonamento de Processos

- Algoritmo Baseado em Prioridades
  - Exemplo 1 (versão não preemptiva):
    - Processos aptos

ID	Tempo de Criação (chegada	Prioridade	Tempo de CPU
P1	0	3	10
P2	0	1	1
P3	0	4	2
P4	0	5	1
P5	0	2	5

Escalonamento de Processos

- Algoritmo Baseado em Prioridades
  - Exemplo 1 (versão não preemptiva):
    - Diagrama de Gantt



Escalonamento de Processos

- Algoritmo *Round-Robin* (revezamento)
  - Este algoritmo foi projetado para sistemas de tempo compartilhado.
  - É parecido com o algoritmo FCFS mas com preempção baseada em um tempo conhecido como *quantum*.
  - Geralmente um *quantum* de tempo tem duração de 10 a 100 milissegundos.
  - A fila de aptos é implementada como uma fila circular, sendo que novos processos são adicionados ao final da fila.

Escalonamento de Processos

- Algoritmo *Round-Robin* (revezamento)
  - Exemplo:
    - Processos aptos

ID	Tempo de Criação (chegada	Tempo de CPU
P1	0	24
P2	0	3
P3	0	3

Escalonamento de Processos

- Algoritmo *Round-Robin* (revezamento)
  - Exemplo:
    - Diagrama de Gantt – quantum de 4 milissegundos

Sincronização de Processos

- Processos cooperativos podem afetar outros processos em execução ou ser por eles afetados.
- Processos cooperativos podem:
  - compartilhar um espaço de endereçamento lógico;
  - compartilhar dados através de arquivos ou mensagens.

Sincronização de Processos

- Processos produtor e consumidor
  - Um recurso é compartilhado entre dois processos, o produtor que insere dados no buffer e o consumidor que retira dados do buffer.

Sincronização de Processos

- Exemplo:
  - Produtor e Consumidor
    - Recurso compartilhado entre dois processos.

Sincronização de Processos

- Produtor e Consumidor em C

```
#define BUFFER_SIZE 10

int buffer[BUFFER_SIZE], contador = 0;

// Produtor

Int in = 0, out = 0, item = 1;

while (1) {
  // Espera ocupada
  while (contador == BUFFER_SIZE);
  buffer[in] = item++;
  in = (in + 1) % BUFFER_SIZE;
  contador++;
}
```

```
// Consumidor

int itemConsumido, out = 0;

while (1) {
  // Espera ocupada
  while (contador == 0);
  itemConsumido = buffer[out] ;
  out = (out + 1) % BUFFER_SIZE;
  contador--;
}
```

Sincronização de Processos

- Processos produtor e consumidor
  - Quais os problemas inerentes ao compartilhamento do recurso (buffer) e da variável contador?

Condição de Corrida!

Sincronização de Processos

- Um seção crítica é um segmento de código em que o processo pode estar:
  - alterando variáveis comuns;
  - atualizando uma tabela;
  - gravando um arquivo;
  - ...

Sincronização de Processos

- Quando um processo estiver executando sua seção crítica, nenhum outro processo deve ter autorização para fazer o mesmo.
- Dois processos não podem estar executando suas seções críticas ao mesmo tempo.

Sincronização de Processos

- Cada processo deve solicitar permissão para entrar em sua seção crítica (protocolo da seção crítica).

```
do {
   Seção de entrada

   Seção crítica

   Seção de saída
   Seção remanescente

} while (1);
```

Sincronização de Processos

- Uma solução para o problema da seção crítica deve satisfazer aos quatro requisitos a seguir:
  1. Exclusão Mútua: enquanto um processo estiver executando sua seção crítica, outros processos não poderão executar suas seções críticas.
  2. Progresso: se um processo necessitar executar sua seção crítica e nenhum processo estiver executando sua seção crítica ele não deve ser impedido. A decisão sobre qual processo acessará a seção crítica deve ser resolvida entre os processos concorrentes e não deve afetar os demais processos do sistema.

Sincronização de Processos

- Uma solução para o problema da seção crítica deve satisfazer aos quatro requisitos a seguir:
  3. Espera Limitada: Um processo não deve aguardar indefinidamente o acesso a sua seção crítica.
  4. Eficiência: um processo deve acessar a seção crítica de maneira eficiente, ou seja sem operações adicionais.

Sincronização de Processos

- Para garantir a seção crítica existem três tipos de soluções possíveis que diferem consideravelmente quanto a recursos utilizados, desempenho e na facilidade de utilização:
  1. Soluções algorítmicas (algoritmo de Dekker e Peterson, algoritmo de Lamport);
  2. Soluções de hardware (inibição de interrupções, instruções especiais);
  3. Objetos do sistema operacional (semáforos, variáveis mutex).

Sincronização de Processos

- Semáforos
  - Criado por E. W. Dijkstra (1965)
  - Tipo abstrato de dado composto por um valor inteiro e uma fila de processos
  - Operações permitidas sobre o semáforo:
    - P(s) (testar / holandês - proberen)
      - Espera até que ***s*** seja maior que 0 e então subtrai 1 de ***s***.
    - V(s) (incrementar / holandês - verhogen)
      - Incrementa ***s*** em 1 unidade.

Sincronização de Processos

- Semáforos
  - Quando um processo executa a operação P sobre um semáforo, o seu valor inteiro é decrementado.
    - Se o valor for negativo, o processo é bloqueado e inserido no fim da fila desse semáforo
  - Quando um processo executa a operação V sobre um semáforo, o seu valor inteiro é incrementado.
    - Se existe algum processo bloqueado na fila desse semáforo, o primeiro processo da fila é liberado

Sincronização de Processos

- Semáforos
  - Definição de P()

```
P(S)
{
  S->valor--;
  if (S->valor < 0) {
    // adiona a lista de espera S-list
    bloqueia;
  }
}
```

  - Definição de V()

```
V(S)
{
  S->valor++;
  if (S->valor <= 0) {
    remove um processo de S->list
    desbloqueia;
  }
}
```

Comunicação entre Processos

- Os processos executando em um sistema computacional podem ser do tipo independentes ou cooperativos.
- Os processos independentes não compartilham dados com os demais processos.
- Os processos cooperativos compartilham algum tipo de dado com um ou mais processos.

Comunicação entre Processos

- A cooperação entre processos oferece as seguintes facilidades:
  - Compartilhamento de Informações: acesso concorrente a dados, por exemplo, mais de um processo acessando um arquivo em disco.
  - Velocidade do Processamento: a divisão de uma tarefa em subtarefas permite minimizar o tempo de processamento. O processamento paralelo acontece se houver mais de um processador disponível.

Comunicação entre Processos

- A cooperação entre processos oferece as seguintes facilidades:
  - Modularidade: um sistema com muitas funcionalidades pode ser dividido em várias threads.
  - Conveniência: um único usuário pode usufruir das vantagens do compartilhamento de informações entre processos.

Comunicação entre Processos

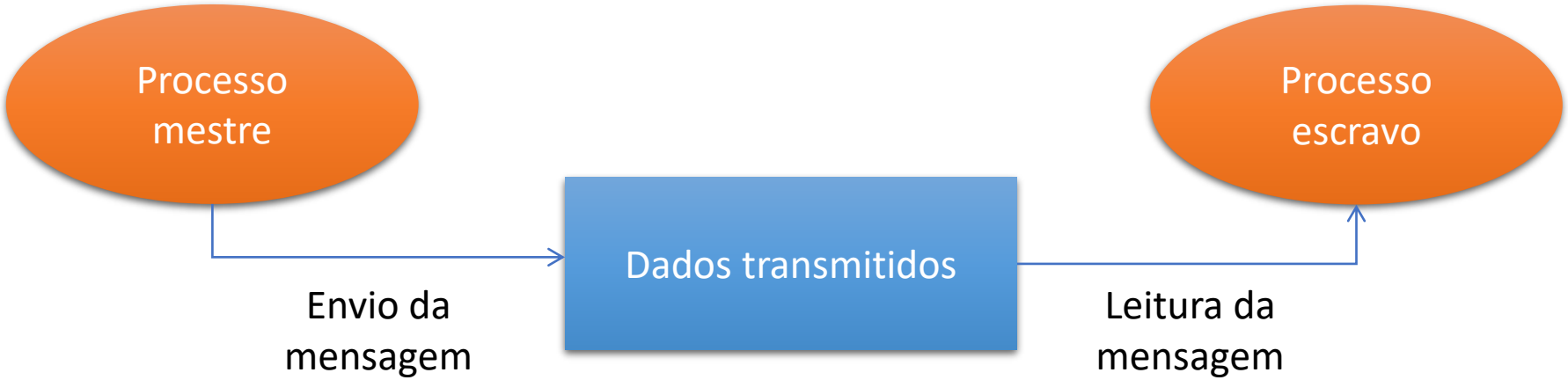
- A interação ou comunicação entre dois ou mais processos pode ser caracterizada por:
  - Um ou mais processos produtores: geram dados/informações que serem lidas/consumidas pelos processos consumidores;
  - Um ou mais processos consumidores: leem/consomem os dados gerados pelos produtores para executar alguma função específica.

Comunicação entre Processos

- Os mecanismos de comunicação entre processos (do Inglês IPC – *Inter-Process Communication*) implementados no SO podem ser:
  - Memória compartilhada;
  - Troca de Mensagens;
  - PIPE;
  - Comunicação Cliente-Servidor.

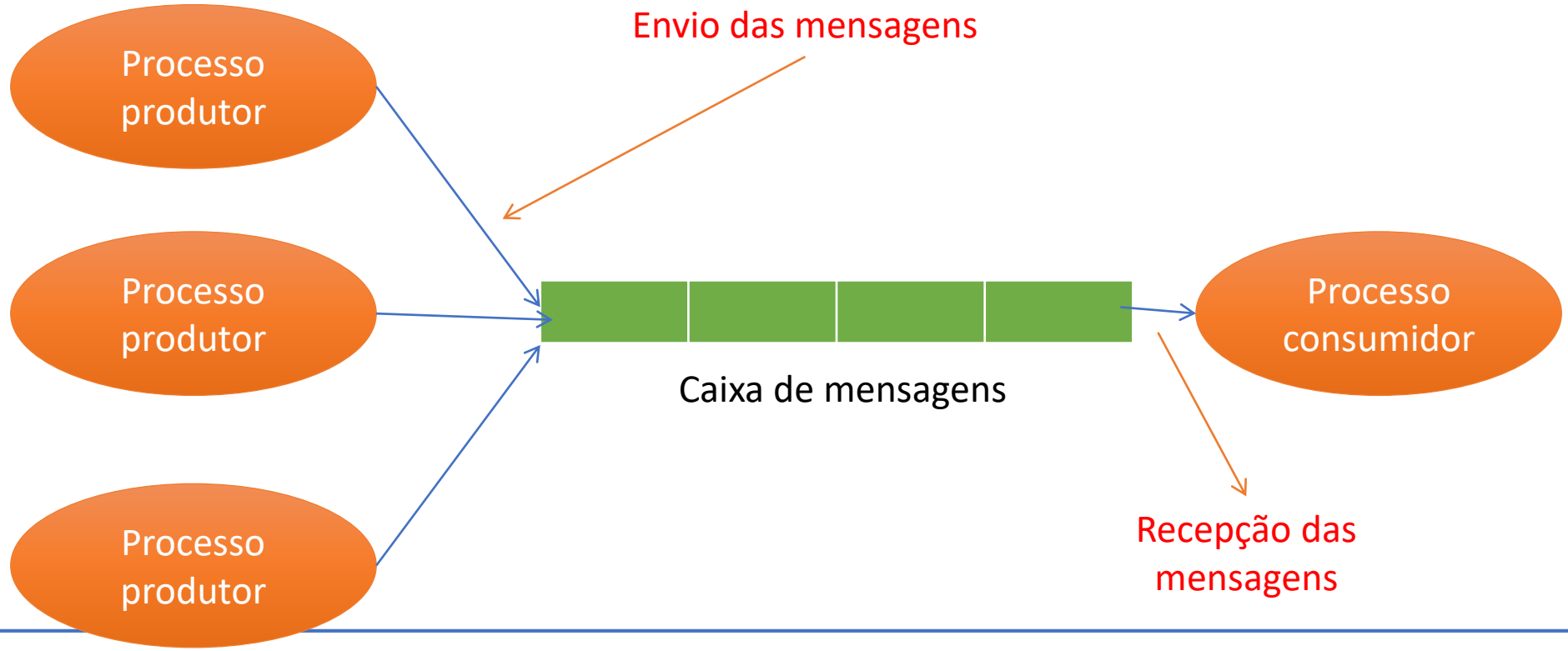
Comunicação entre Processos

- Modelo de Interação Um para Um (mestre-escravo)



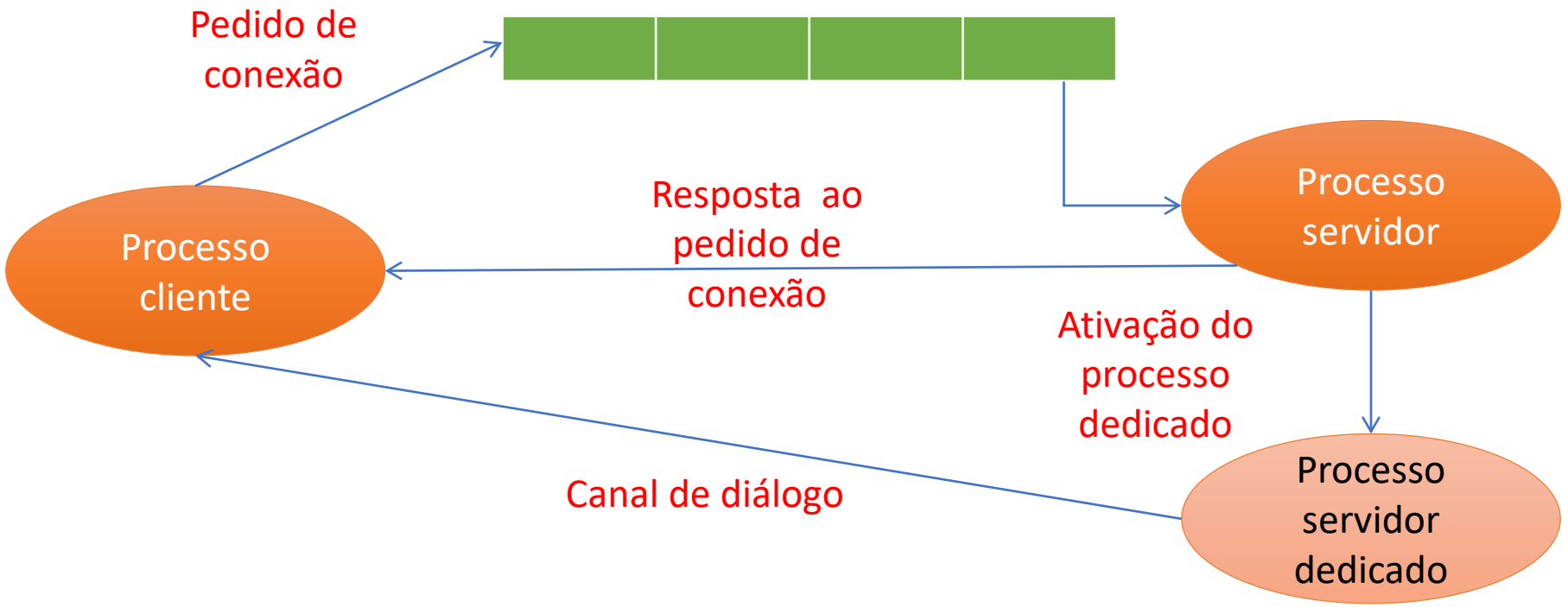
Comunicação entre Processos

- Modelo de Interação Muitos para Um



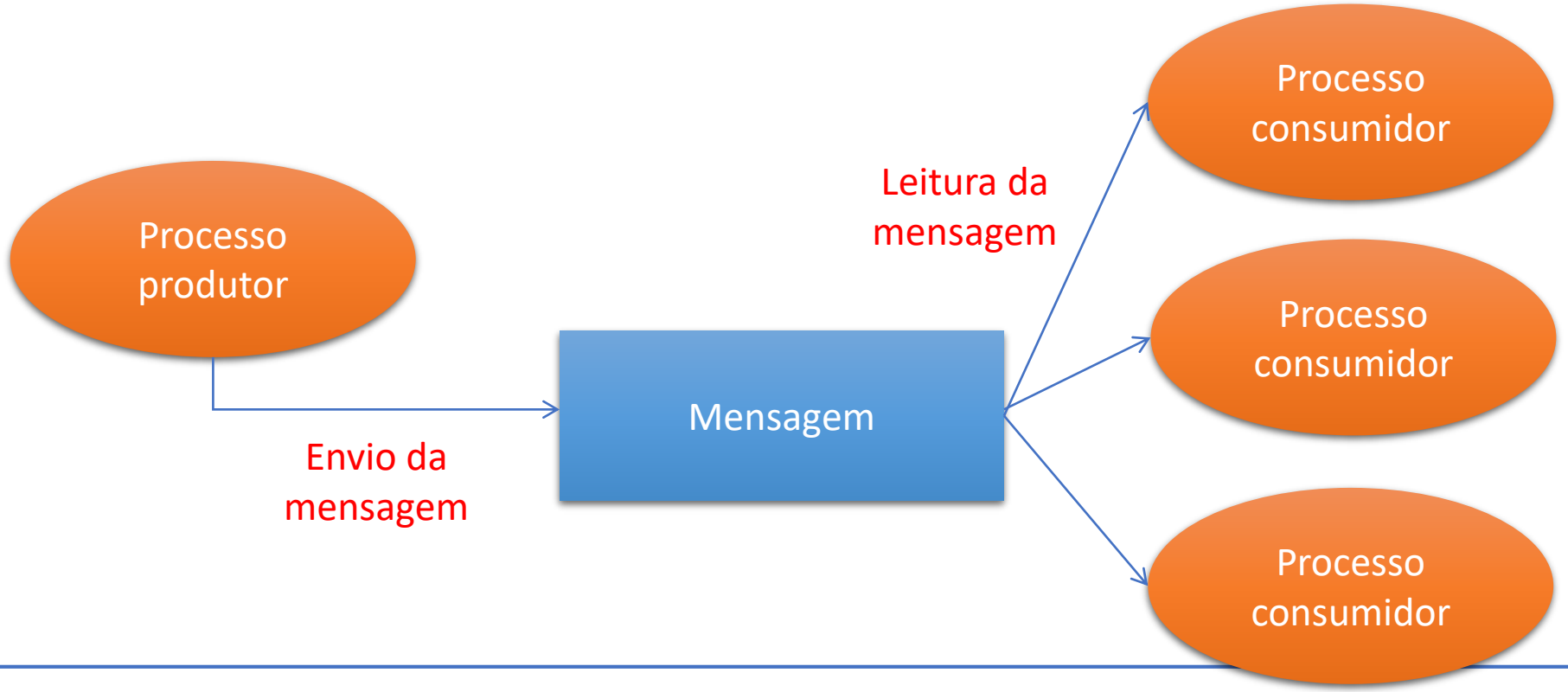
Comunicação entre Processos

- Modelo de Interação Um para Um de Vários



Comunicação entre Processos

- Modelo de Interação Um para Muitos (difusão)

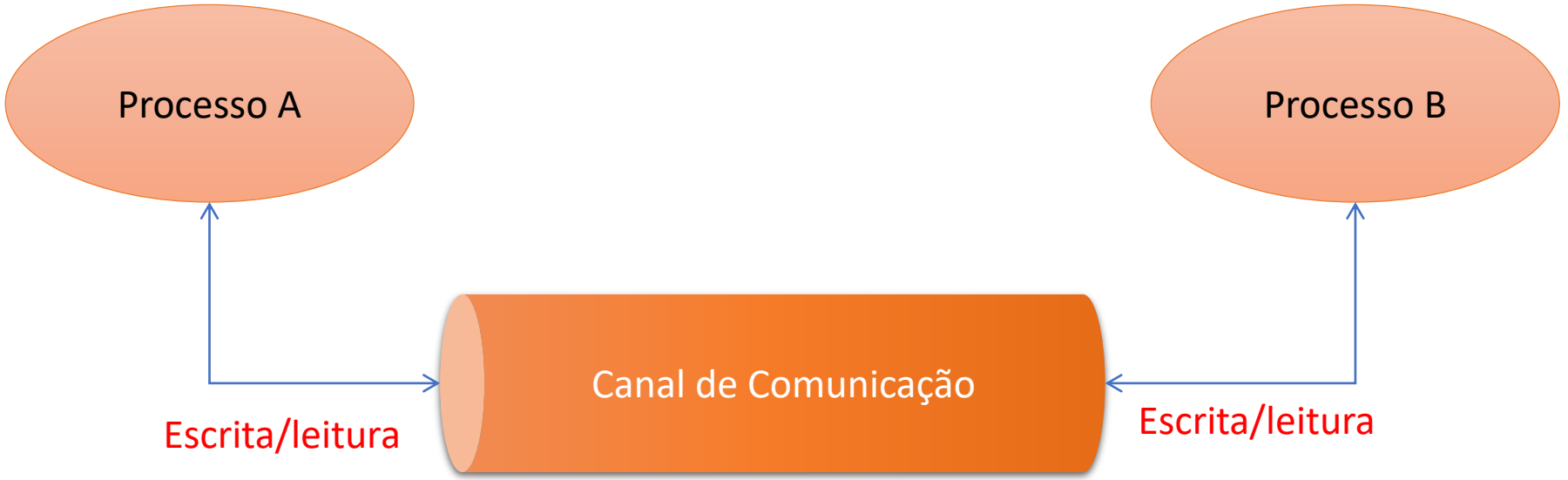


Comunicação entre Processos

- A comunicação no modelo computacional é realizada por mecanismos disponibilizados pelo sistema operacional ou disponibilizados nos ambientes de programação.
- A comunicação entre processos é suportada por um objeto do tipo *canal de comunicação*.
- A transferência de informações entre processos pode ser vista como resultado da invocação de operações sobre um objeto canal.

Comunicação entre Processos

- Objeto do tipo Canal de Comunicação



Comunicação entre Processos

- As operações realizadas em um canal de comunicação podem ser:
  - Criar: cria um canal de comunicação que será utilizado por dois ou mais processos.
  - Associar: associa o processo a um canal de comunicação.
  - Enviar: envia dados para o canal.
  - Receber: recebe dados do canal.
  - Terminar: fecha o canal de comunicação, sinalizando que a comunicação foi realizada.
  - Eliminar: elimina o canal de comunicação.

Comunicação entre Processos

- A comunicação entre processos pode se dar em diversos contextos, sendo:
  - Processos executando em um mesmo computador com relação hierárquica (processo pai e processo filho);
  - Processos executando em um mesmo computador sem relação hierárquica;
  - Processos executando em computadores diferentes com o mesmo sistema operacional;
  - Processos executando em computadores diferentes com sistemas operacionais diferentes.

Comunicação entre Processos

- A comunicação entre processos não somente determina um meio de troca de informações entre processos, mas também um mecanismo para sincronizar as ações dos processos comunicantes.
- A semântica na comunicação de processos determina o comportamento do processo ao receber e enviar um mensagem, e podem ser:
  - **Síncrona:** o produtor fica bloqueado até que o consumidor receba a mensagem e acesse o seu conteúdo.
  - **Assíncrona:** o produtor envia a mensagem e continua a execução, assim que esta tenha sido armazenada no canal de forma temporária até que seja recebida pelo consumidor.
  - **Cliente-Servidor:** o processo produtor (cliente) fica bloqueado até que o consumidor (servidor) tenha recebido a mensagem e enviado uma mensagem de resposta.

Comunicação entre Processos

- Um canal de comunição pode ser implementado de duas formas distintas:
  - Memória compartilhada: os processos comunicantes acessaram uma área de memória que faz parte do espaço de endereçamento de ambos.
  - Transferência através do núcleo do sistema operacional: os dados são sempre copiados para o núcleo antes de serem transferidos para o consumidor.

Comunicação entre Processos

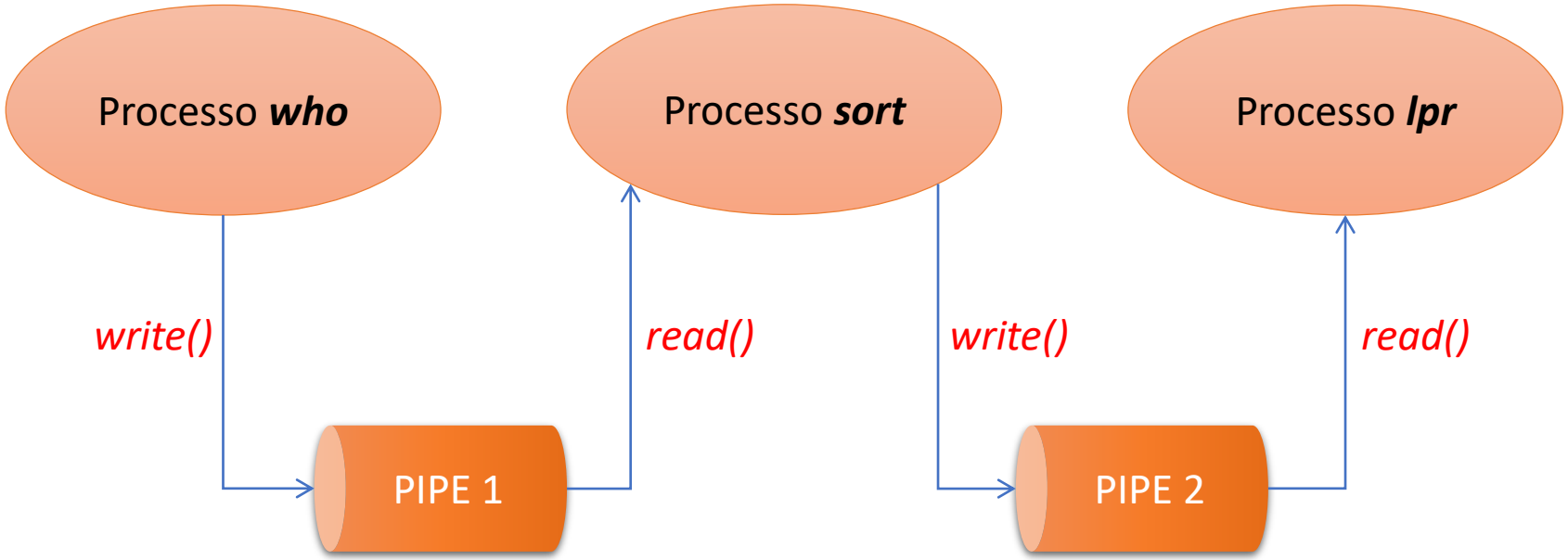
- Existem basicamente três classes de canais de comunicação:
  - Memória compartilhada;
  - Caixas de mensagens;
  - Conexões virtuais (stream).

Comunicação entre Processos

- Um PIPE pode ser enquadrado como uma versão limitada das classes de mecanismos das conexões virtuais.
- O PIPE liga dois processos o permite o fluxo de informações de forma unidirecional.
- São adequados para a implementação de mecanismos mestre-escravo (um-para-um).
- Os PIPEs podem ser anônimos ou possuírem um nome.
- Um PIPE anônimo tem que ser usado entre processos que possuam relação hierárquica (pai e filho).
- Cada PIPE é representando por um arquivo especial no sistema de arquivos.

Comunicação entre Processos

- Exemplo 1:
  - *who* | *sort* | *lpr*



Comunicação entre Processos

- Exemplo 2:
  - Programa em C para troca de mensagens entre dois processos via PIPE.
  - Cabe ressaltar que os processos possuem uma hierarquia, ou seja, processo pai e processo filho utilizando o PIPE.

Comunicação entre Processos

- Exemplo 2:

```
1.  #include <unistd.h>
2.  #include <stdio.h>
3.  #include <string.h>
4.  #include <stdlib.h>
5.  #define SIZE 100

6.  int main()
7.  {
8.    int fd[2], pid;
9.    char msg[SIZE];

10.  if (pipe(fd) < 0) {
11.    printf("Erro ao criar o pipe...\n");
12.    exit(0);
13.  }
14. pid = fork();

15.  if (pid > 0) {  // Processo pai
16.    close(fd[0]);
17.    write(fd[1], "Fala ai filho...", strlen("Fala ai
       filho...")+1);
18.    close(fd[1]);
19.  }
20.  else if (pid == 0) {  // Processo filho
21.    close(fd[1]);
22.    read(fd[0], msg, sizeof(msg));
23.    printf("Mensagem recebida do pai...:
       %s\n", msg);
24.    close(fd[0]);
25.  }
26.  return 0;
27. }
```

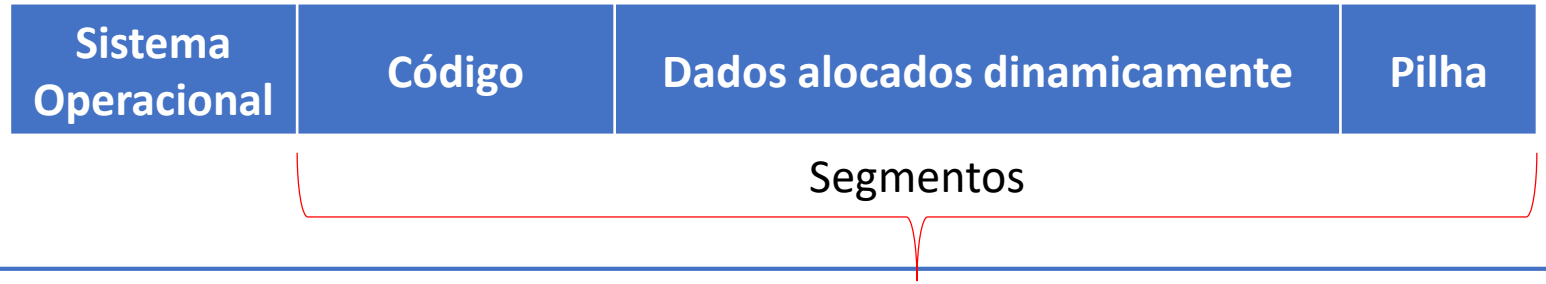
Gerência de Memória

- O Sistema Operacional é o responsável pelo gerenciamento da memória, ou seja, seu uso e otimização.
- Um processo ocupa uma porção de memória denominado *espaço de endereçamento do processo*.
- Um espaço de endereçamento de um processo é o conjunto de posições de memória que um programa executado por este processo pode referenciar.

Gerência de Memória

- Espaço de Endereçamento

- Espaço de Endereçamento Visto pelos Programadores



Gerência de Memória

- Existe uma clara noção de confinamento do processo ao seu espaço de endereçamento válido.
- O sistema operacional tem em cada instante um mapa preciso de quais posições de memória o programa pode acessar e de que forma.
- O confinamento garantido pelo SO é chamado de mecanismo de proteção de memória.

Gerência de Memória

- Um endereço de memória permite acessar um byte que conterá parte ou a totalidade do dado ou instrução que se quer acessar.

endereço -> valor

Ou

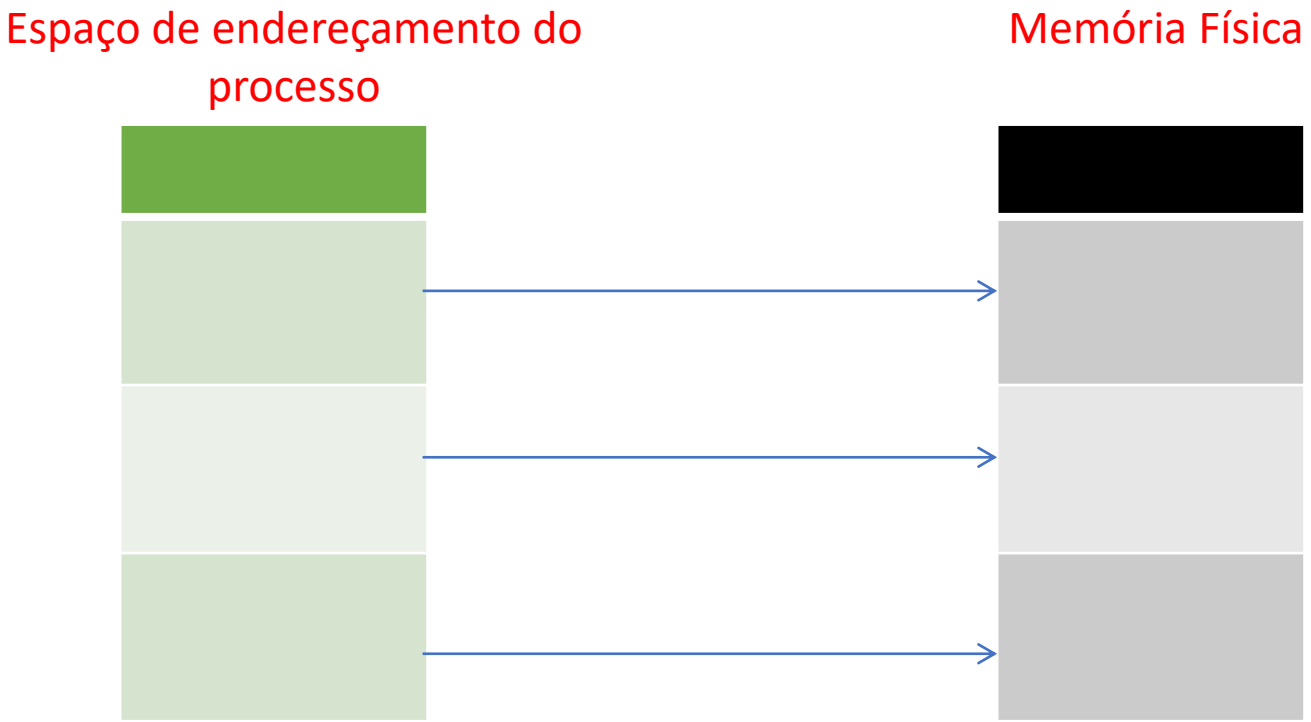
endereço virtual -> endereço real -> valor

Gerência de Memória

- Normalmente o sistema operacional disponibiliza um conjunto de chamadas de sistema para manipulação da memória.
- Chamadas de sistema para manipular memória:
  - Alocar;
  - Liberar;
  - Proteger;
  - Mapear;
  - Desmapear;
  - Associar;
  - Desassociar.

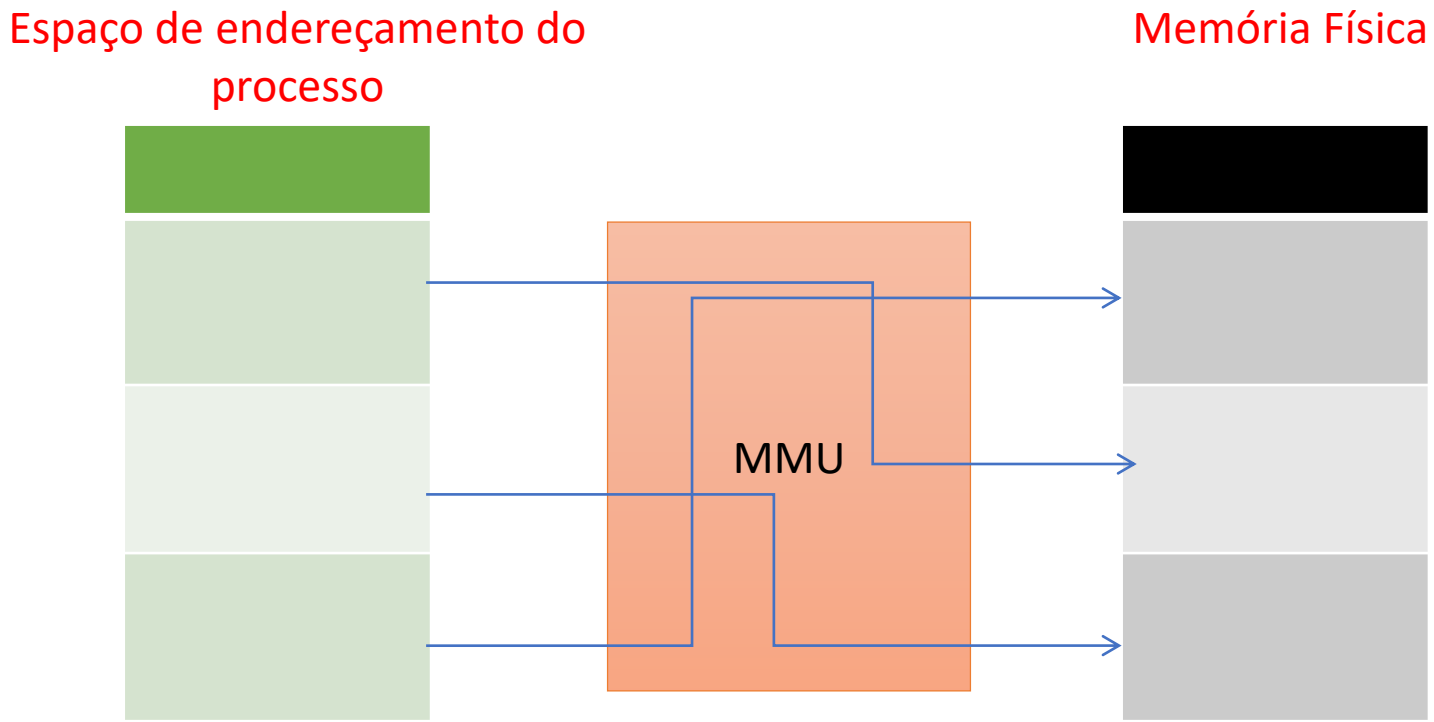
Gerência de Memória

- Endereçamento Real (exemplo)



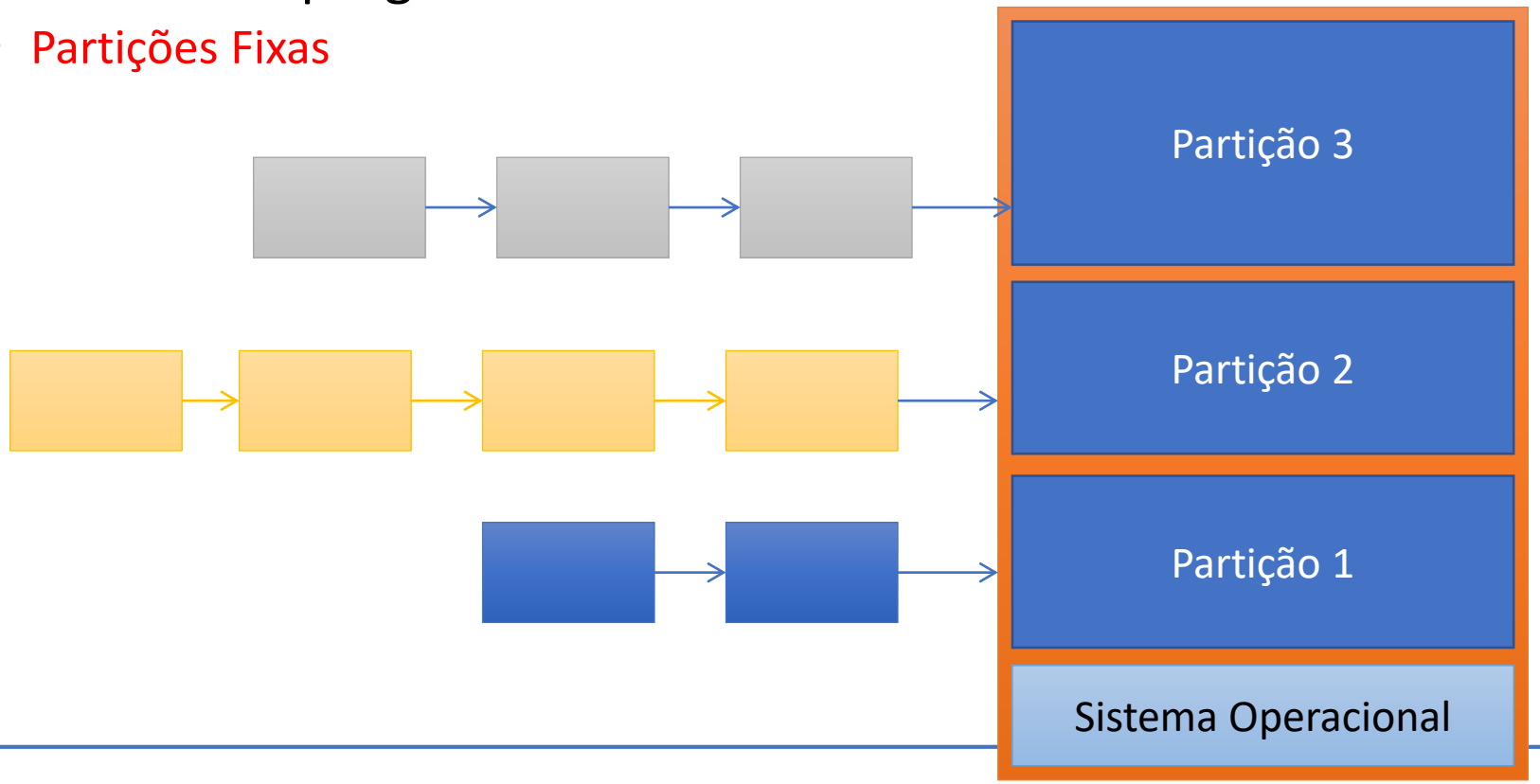
Gerência de Memória

- Endereçamento Virtual (exemplo)



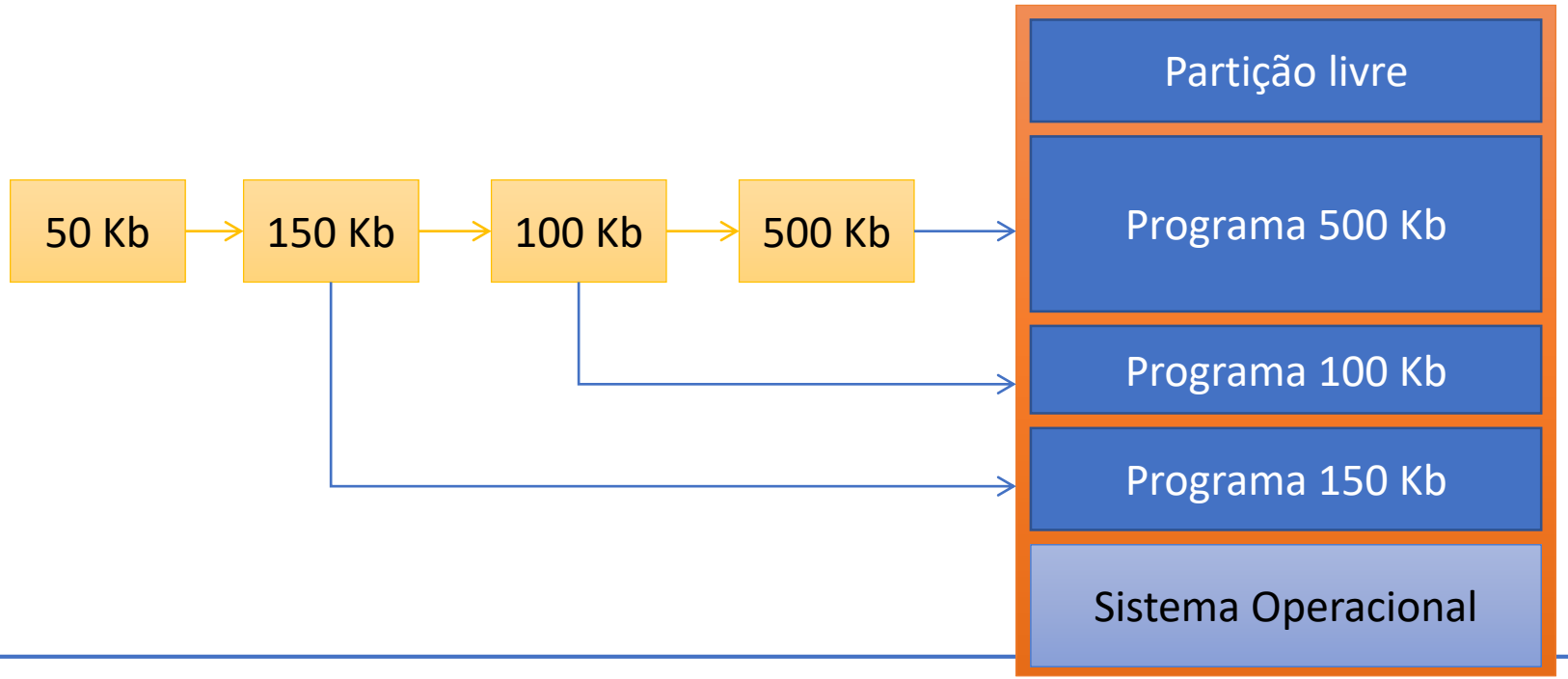
Gerência de Memória

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Fixas



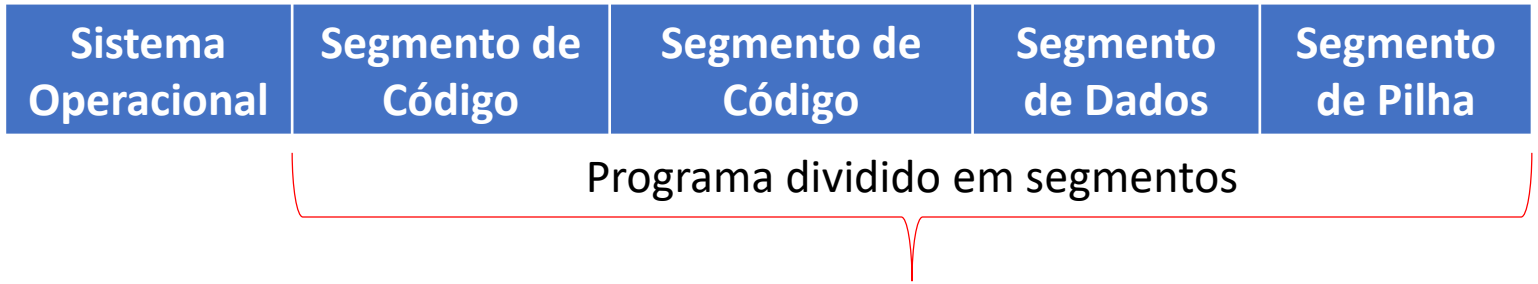
Gerência de Memória

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Variáveis



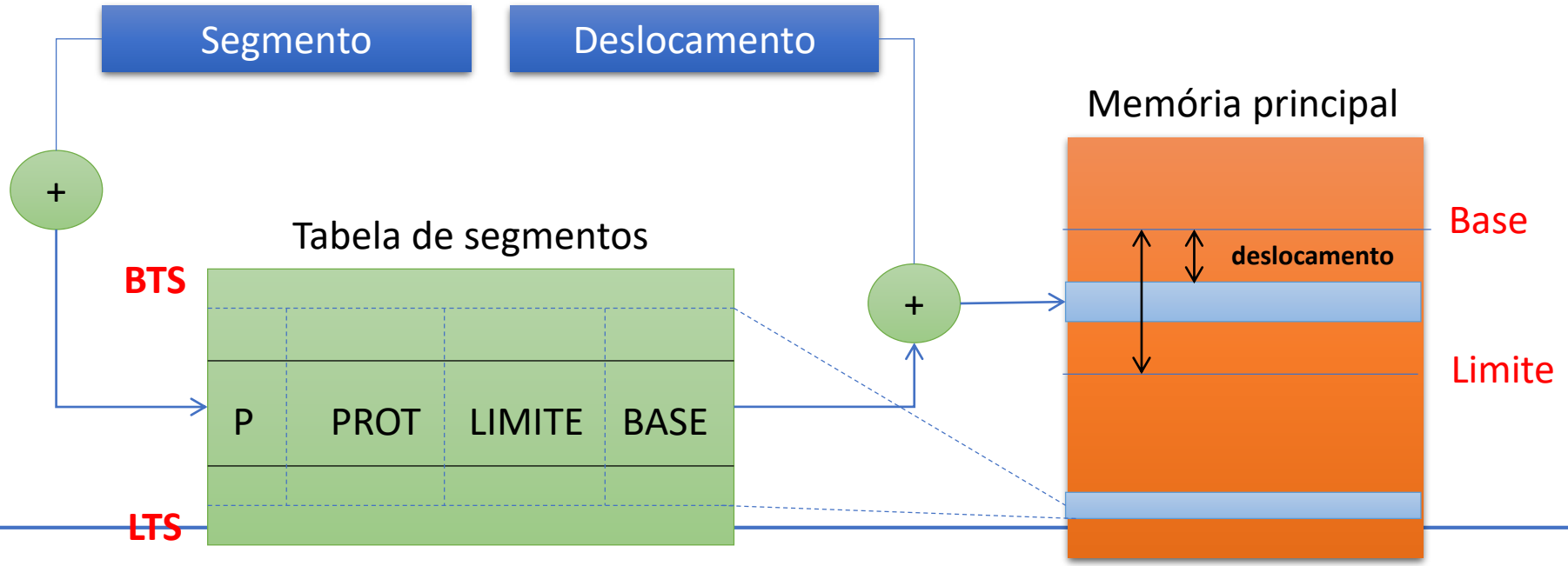
Gerência de Memória

- Endereçamento Virtual
  - Segmentação



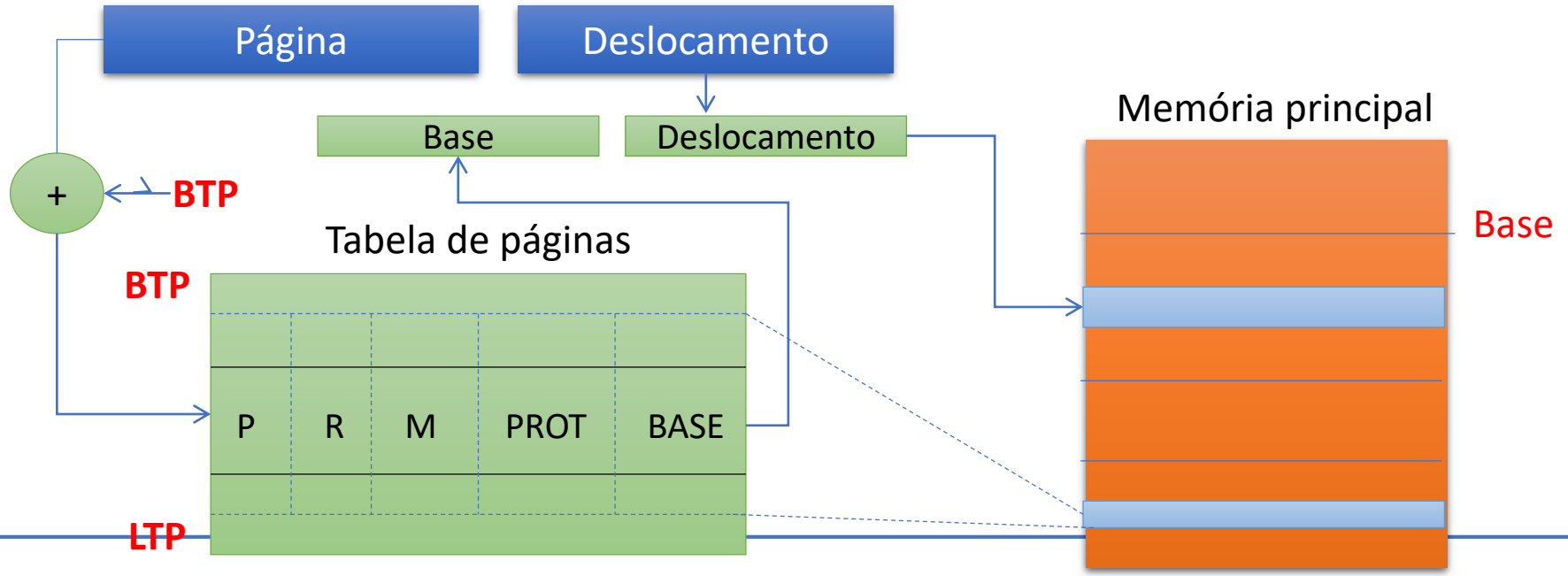
Gerência de Memória

- Endereçamento Virtual
  - Segmentação
    - Tabela de Segmentos



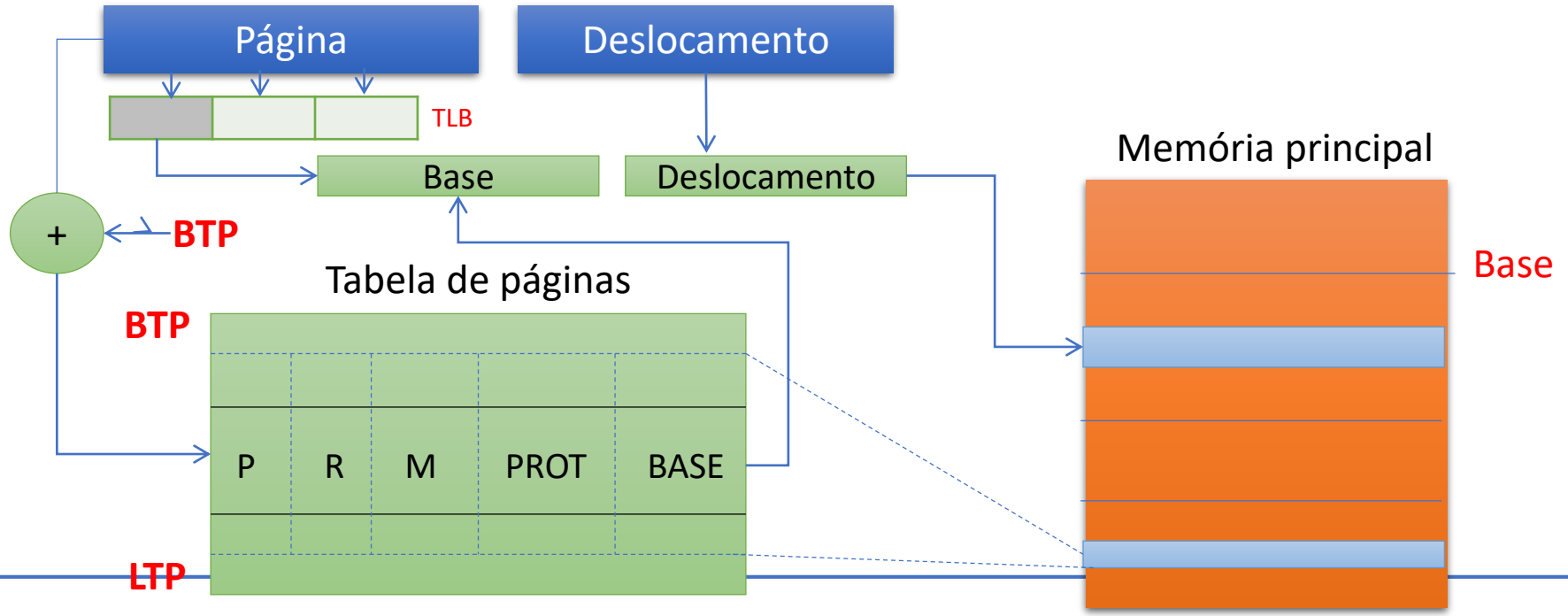
Gerência de Memória

- Endereçamento Virtual
  - Paginação
    - Tabela de Páginas



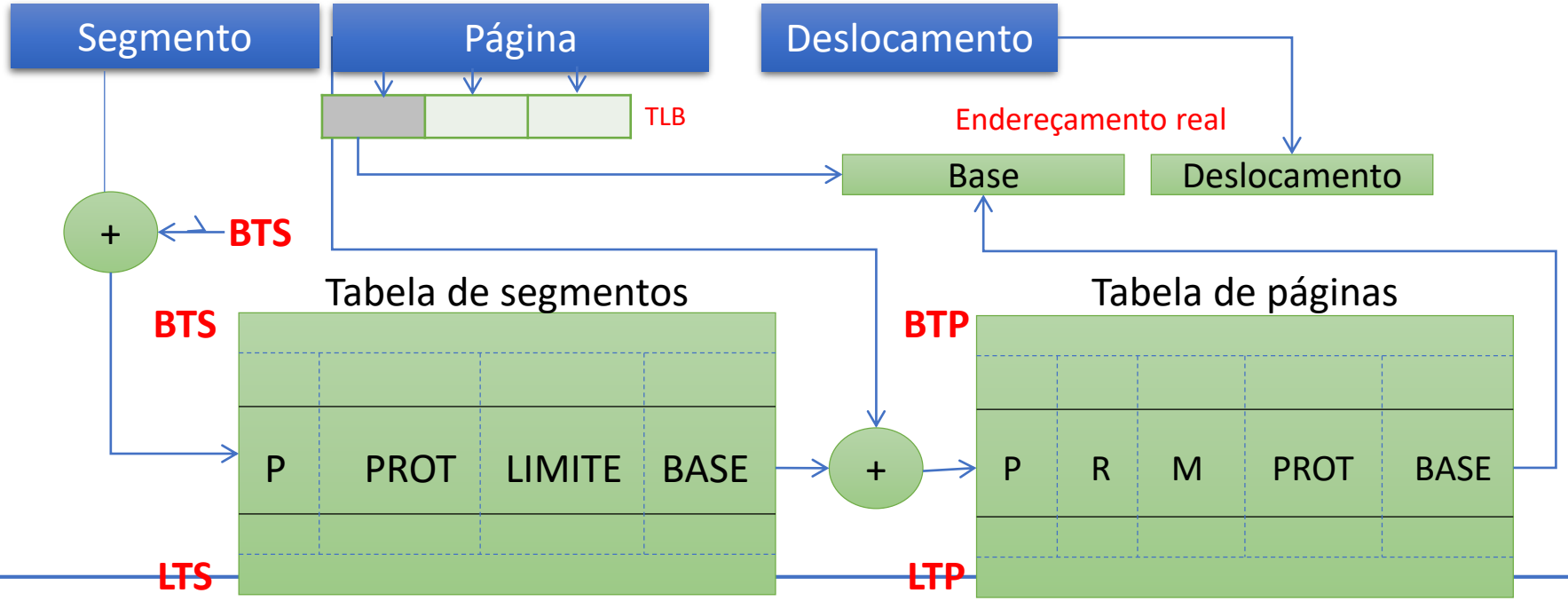
Gerência de Memória

- Endereçamento Virtual
  - Paginação
    - Tabela de Páginas com TLB



Gerência de Memória

- Endereçamento Virtual
  - Segmentação Paginada
    - Tabela de Segmentos/Páginas com TLB



Gerência de Entrada e Saída de Dados

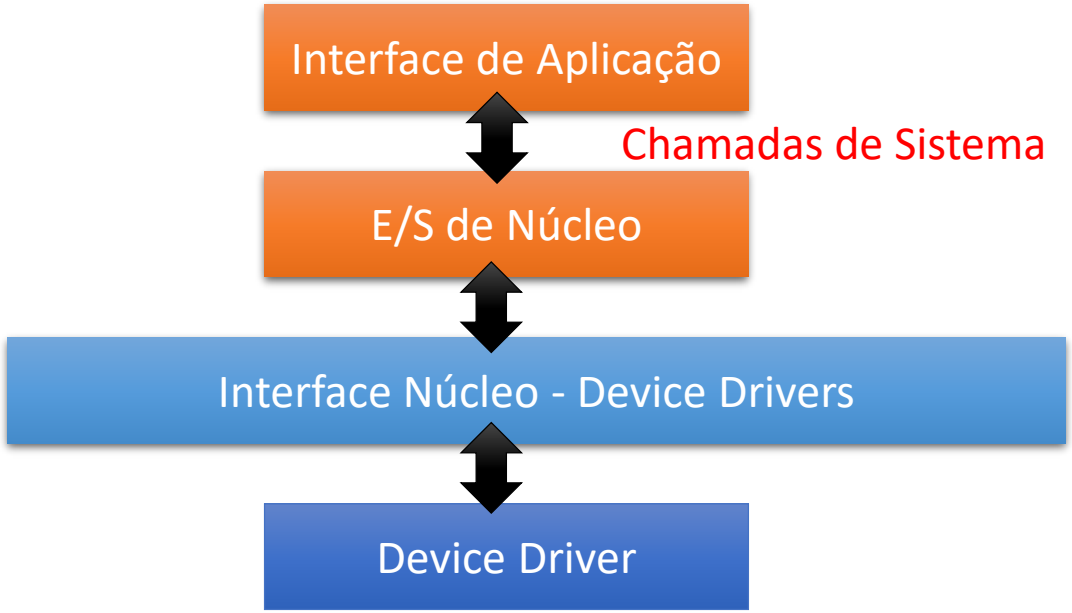
- A camada de E/S em um SO é responsável por prover um meio de comunicação entre os processos e os dispositivos de Entrada e Saída de dados.
- Os dispositivos de E/S, também conhecidos como periféricos, proveem uma gama de funcionalidades ao sistema computacional.
- A camada de E/S deve ser capaz de interfacear com diferentes dispositivos com diferentes configurações.

Gerência de Entrada e Saída de Dados

- Os objetivos da camada de E/S são:
  - Tornar a programação independente de periféricos e de todos os detalhes de E/S.
  - Permitir a fácil inclusão de novos periféricos, o que implica que é possível a diferentes fabricantes criarem o software de adaptação entre o núcleo do SO e as funções específicas de controle de periféricos.

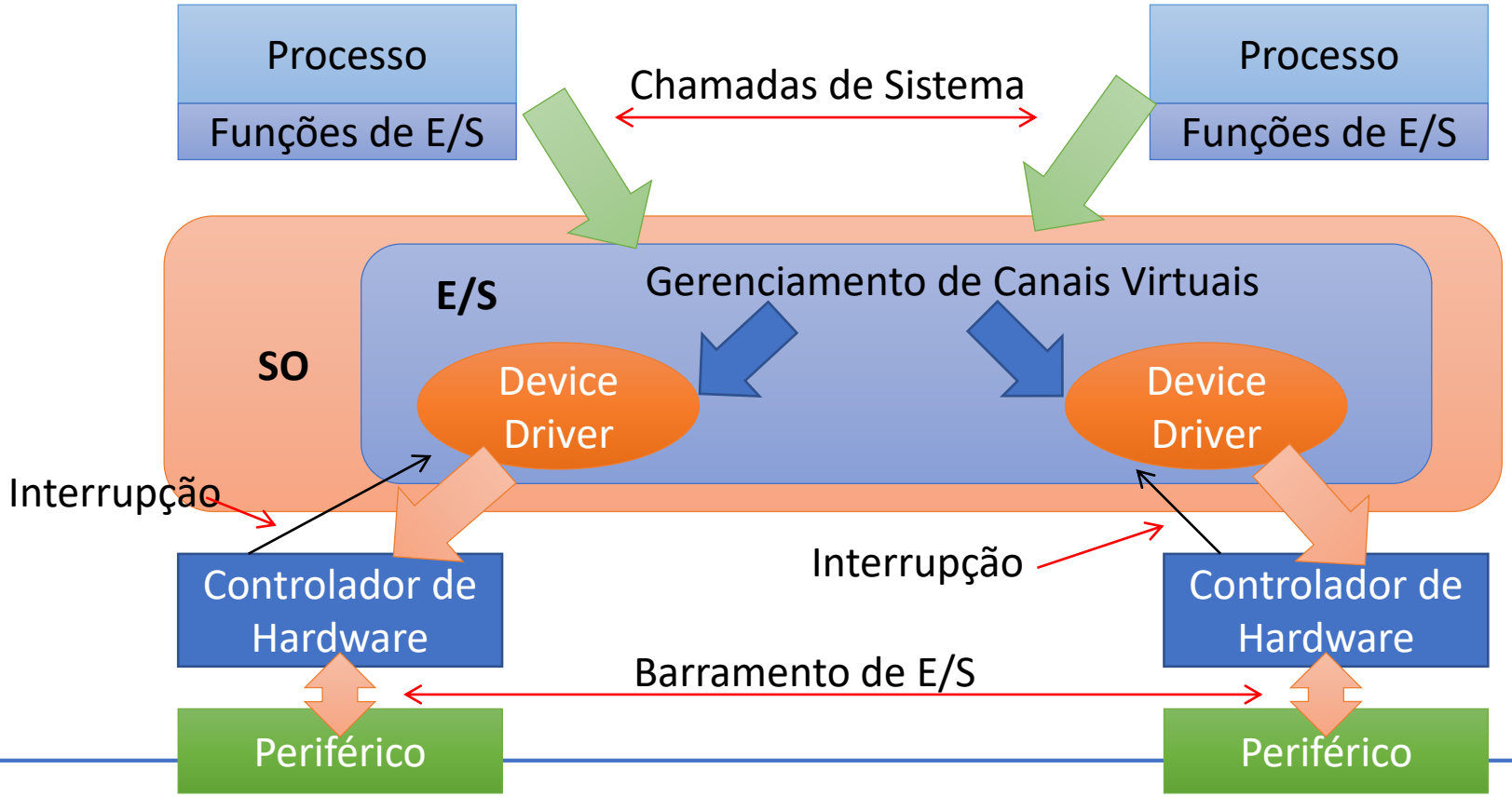
Gerência de Entrada e Saída de Dados

- Arquitetura de um Sistema de E/S



Gerência de Entrada e Saída de Dados

- Ações de um Operação de E/S



Gerência de Entrada e Saída de Dados

- Os *device drivers* podem ser de dois tipos com relação ao SO:
  1. Um processo independente: neste caso o device driver tem uma pilha de execução, cache, espaços de memória etc.
  2. Integrado ao núcleo: o *device driver* faz parte do núcleo do SO, evitando assim muitas troca de contexto na comunicação entre um processo e um dispositivo de E/S.

Gerência de Entrada e Saída de Dados

- A comunicação do *device driver* com o dispositivo ao qual gerencia, pode ser realizada da seguinte forma:
  - Acesso Direto a Memória (*DMA – Direct Memory Access*)
    - Quando existe um mapeamento direto entre a memória principal e o periférico, permitindo que este grave dados diretamente na memória.

Gerência de Entrada e Saída de Dados

- A comunicação do *device driver* com o dispositivo ao qual gerencia, pode ser realizada da seguinte forma:
  - E/S Programada
    - Leitura de dados diretamente do controlador. Por exemplo, atualização das informações sobre o posicionamento do mouse na interface gráfica.

Gerência de Entrada e Saída de Dados

- Quando um dispositivo de E/S conclui uma operação de escrita ou leitura de dados, esta deve ser sinalizada ao *device driver*.
- As formas de sinalização/verificação de fim de operação são:
  - Polling: o gerenciador testa periodicamente o estado do controlador.
  - Interrupção: o processador é interrompido quando a operação termina.