

Aula 2

Professores:

Felipe M. G. França
Valmir C. Barbosa

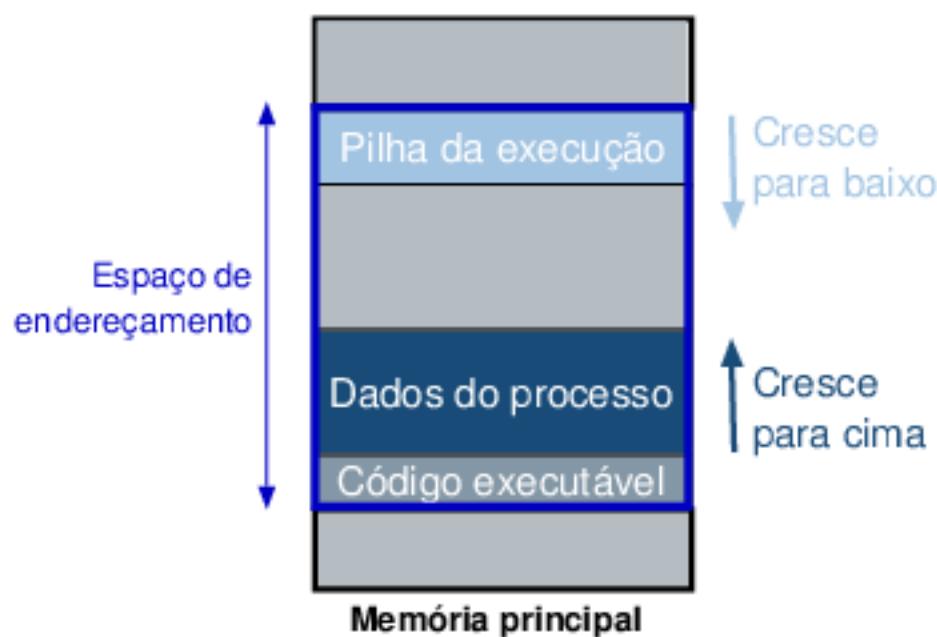
Conteúdo:

Elementos do sistema operacional

- Processos
- Arquivos e diretórios
- Interpretador de comandos (shell)
- Chamadas ao sistema operacional

Processo

- ➔ Um programa em execução na máquina.
- ➔ Possui um **espaço de endereçamento** na memória, com:
 - O código executável do processo.
 - Os dados do processo.
 - A pilha da execução do processo.



Processo

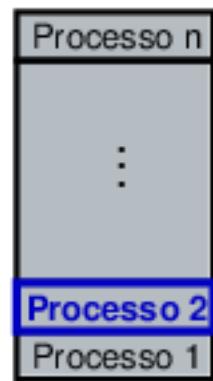
- ➡ Também associados ao processo estão os registradores da CPU.
- ➡ Existem outras informações associadas ao processo, como os identificadores de arquivos abertos.
- ➡ O sistema deve ser capaz de parar e reiniciar um processo, sem que isso afete a sua execução.
- ➡ O sistema deve permitir a criação e a destruição de processos.
- ➡ Cada processo está associado a um usuário, e possui:
 - Uma identificação para o processo no sistema (PID).
 - A identificação do usuário que o iniciou (UID).
 - A identificação do grupo do usuário que o iniciou (GID).

Processo

→ Tabela de processos do sistema:

- Possui uma entrada para cada processo do sistema.
- Armazena todas as informações do processo, com exceção das contidas no espaço de endereçamento.
- Permite que o sistema reinicie a execução de um processo suspenso.

Tabela de processos

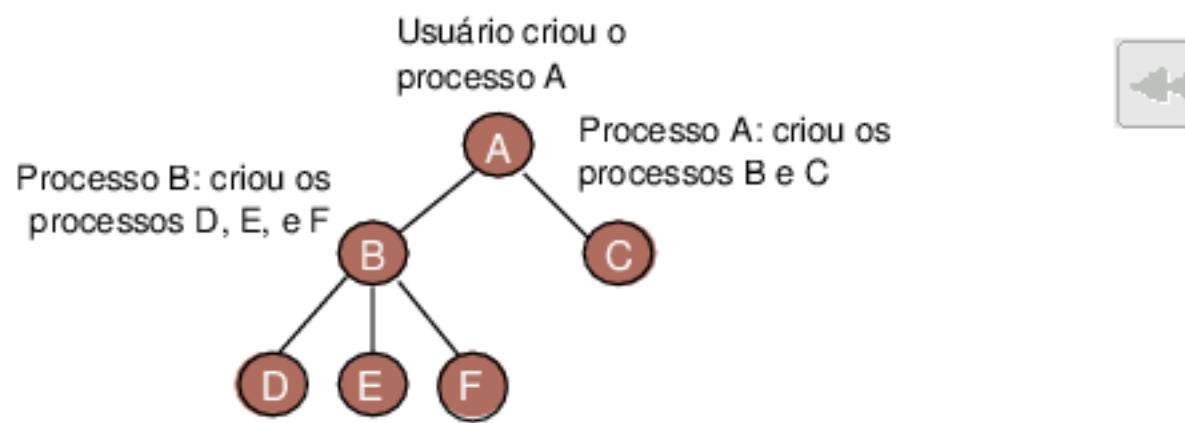


:
Sinais pendentes
Identificadores de arquivos abertos
Tempo de execução
GID
UID
PID

Exemplo do conteúdo que uma das entradas pode ter para o Processo 2 (as entradas são iguais para todos os processos). Em geral, a tabela é um vetor, e com isso, existirá um número máximo de processos que podem executar no computador.

Processo

- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

- ➡ Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- ➡ A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:

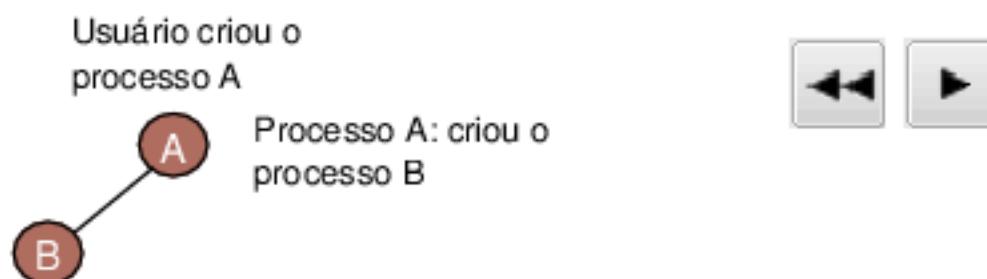
Usuário criou o
processo A



- ➡ Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

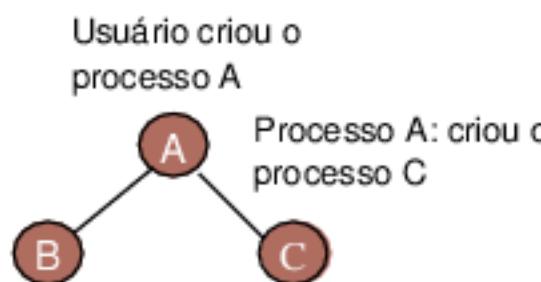
- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

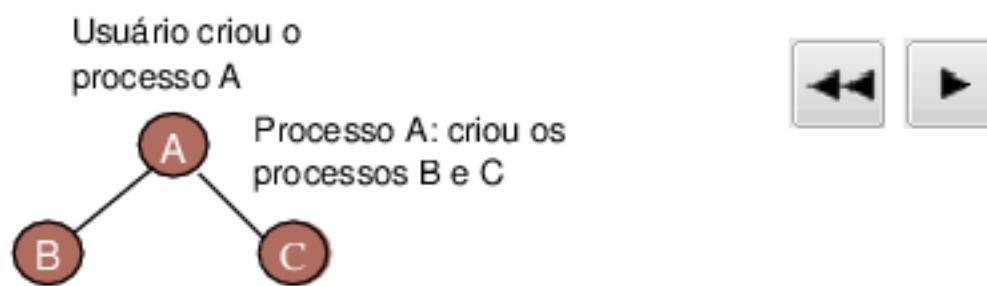
- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

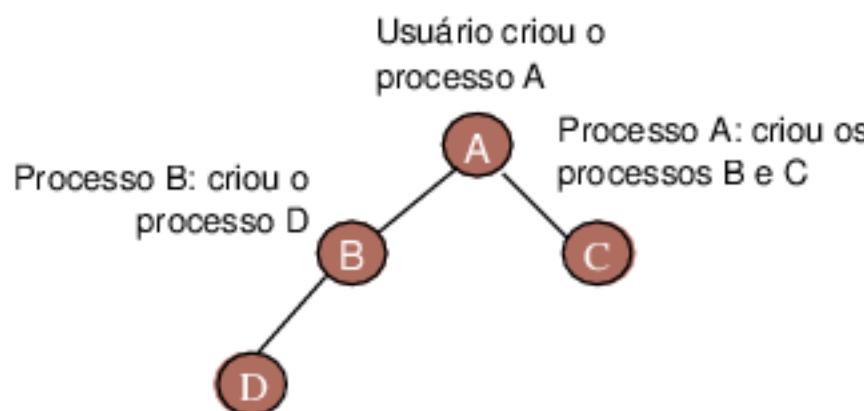
- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

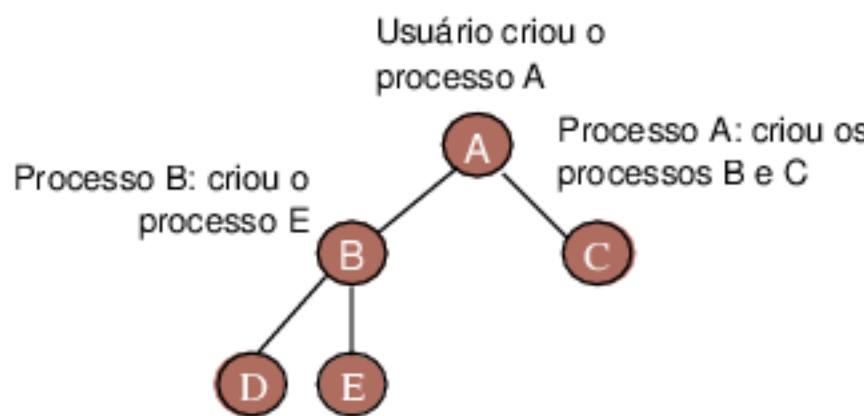
- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

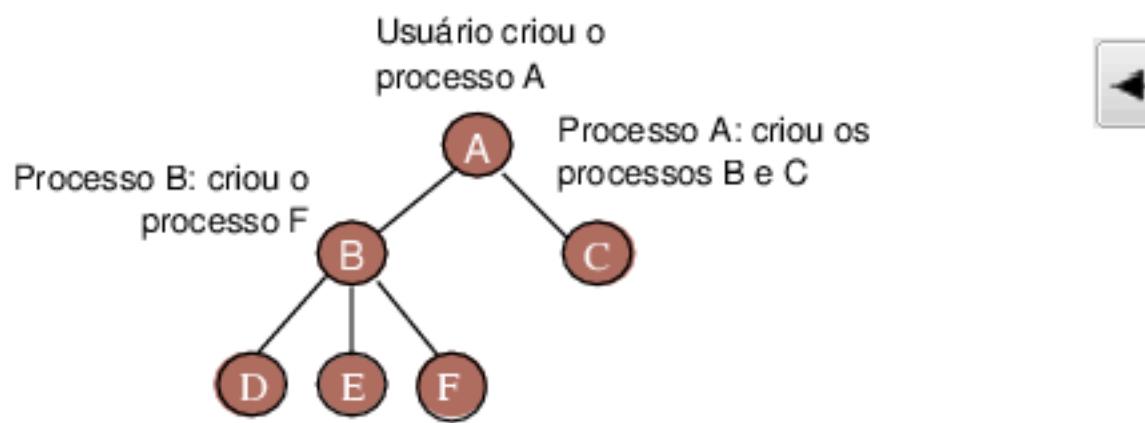
- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

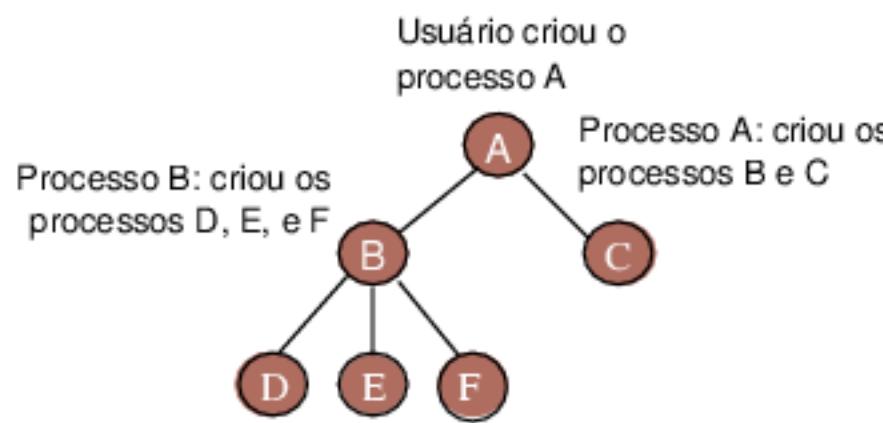
- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

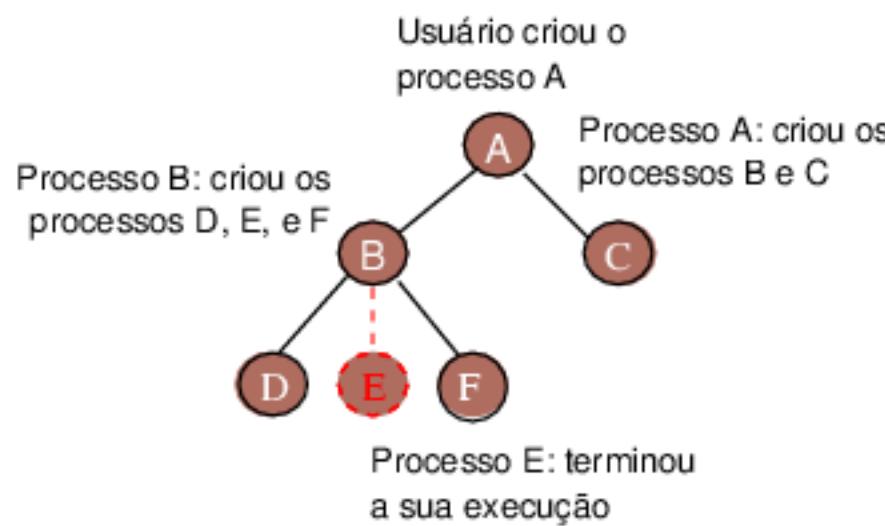
- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Processo

- Um processo pode criar um outro, que será chamado de o **filho** do processo. Este processo será o **pai** do processo criado.
- A criação de processos por outros gera uma hierarquia chamada de **árvore de processos**:



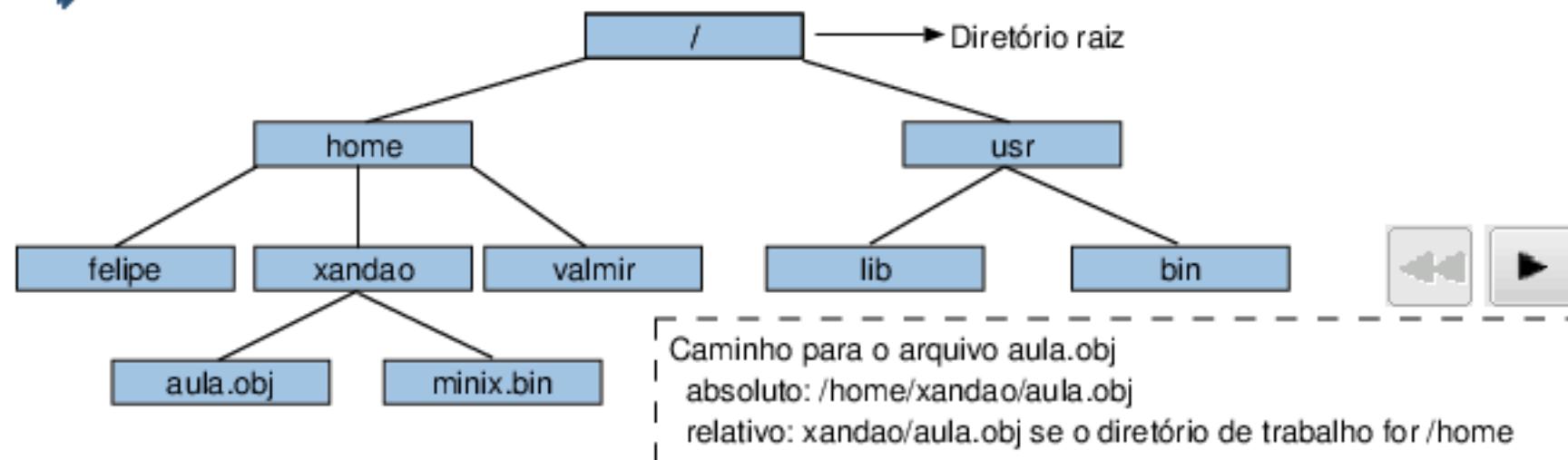
- Os processos podem responder a eventos através da definição e do tratamento de sinais.

Arquivos e diretórios

- ➡ Os **arquivos** organizam as informações armazenadas no disco.
- ➡ Um arquivo permite um acesso de mais alto nível e transparente a um dispositivo de E/S.
- ➡ Um **diretório** é um arquivo especial usado para agrupar um conjunto de arquivos relacionados.
- ➡ Um diretório pode também possuir outros diretórios.
- ➡ Cada processo do sistema possui um **diretório de trabalho**.
- ➡ O conjunto de arquivos da máquina é chamado de **o sistema de arquivos**.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

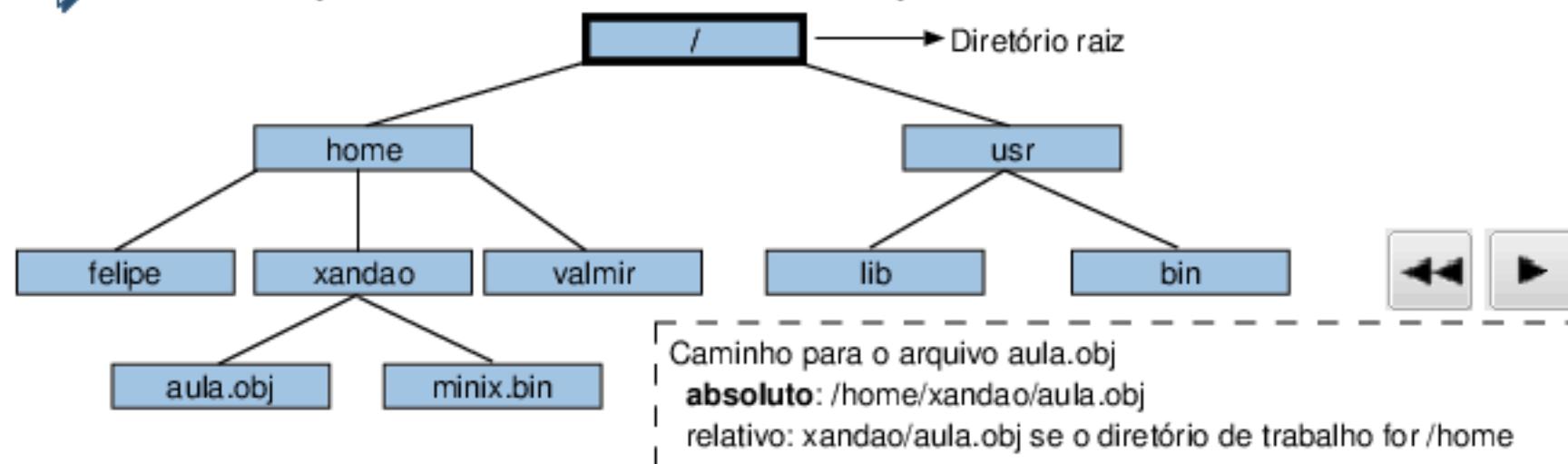


→ **Nome do caminho** de um arquivo:

- **Nome absoluto**: a partir do diretório raiz (inicia com /).
- **Nome relativo**: a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

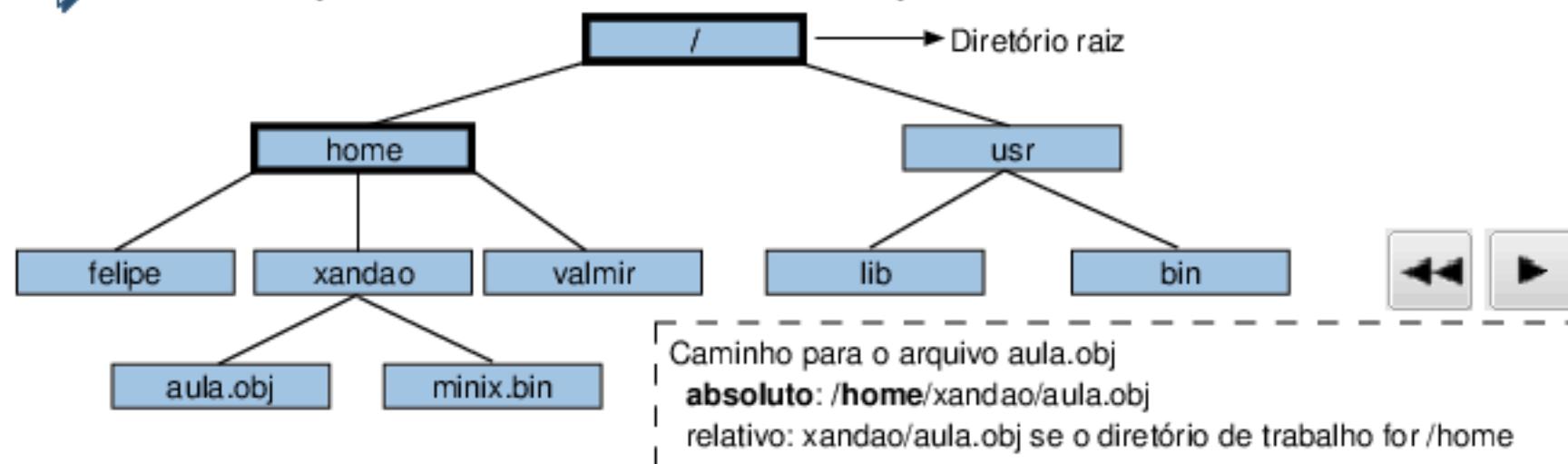


→ **Nome do caminho** de um arquivo:

- **Nome absoluto:** a partir do diretório raiz (inicia com /).
- **Nome relativo:** a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

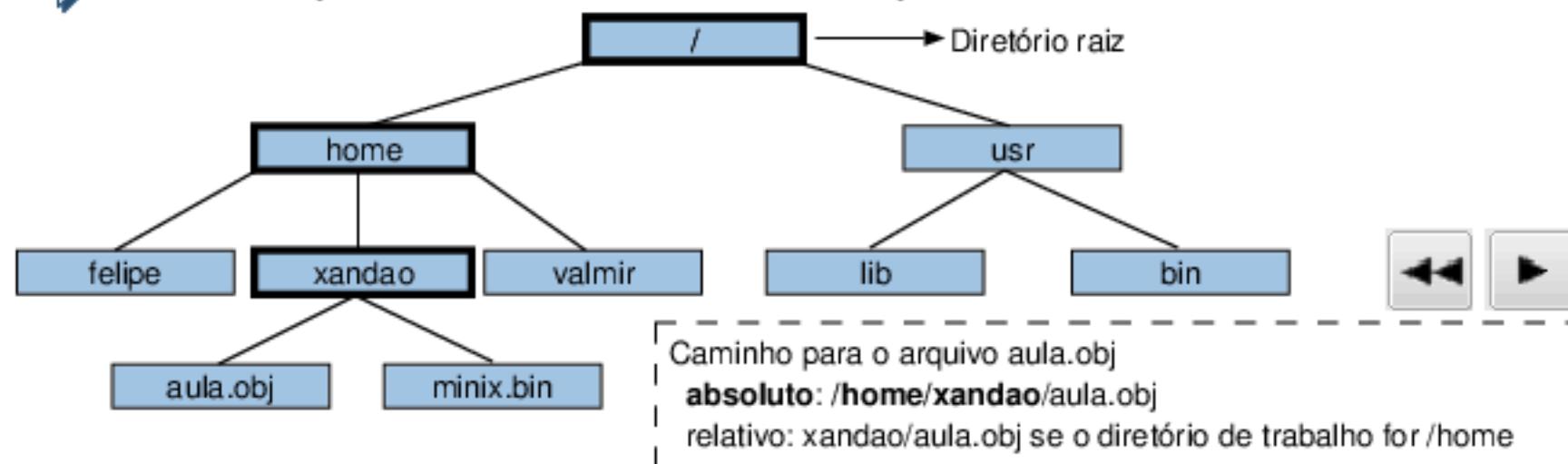


→ Nome do caminho de um arquivo:

- **Nome absoluto:** a partir do diretório raiz (inicia com /).
- **Nome relativo:** a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

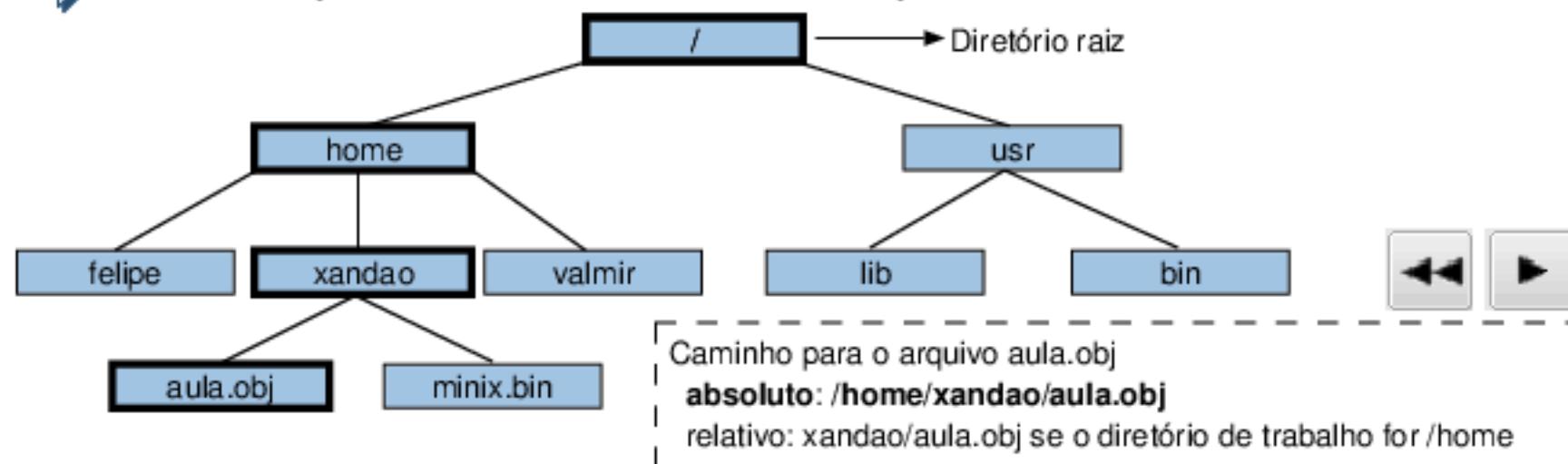


→ **Nome do caminho** de um arquivo:

- **Nome absoluto:** a partir do diretório raiz (inicia com /).
- **Nome relativo:** a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

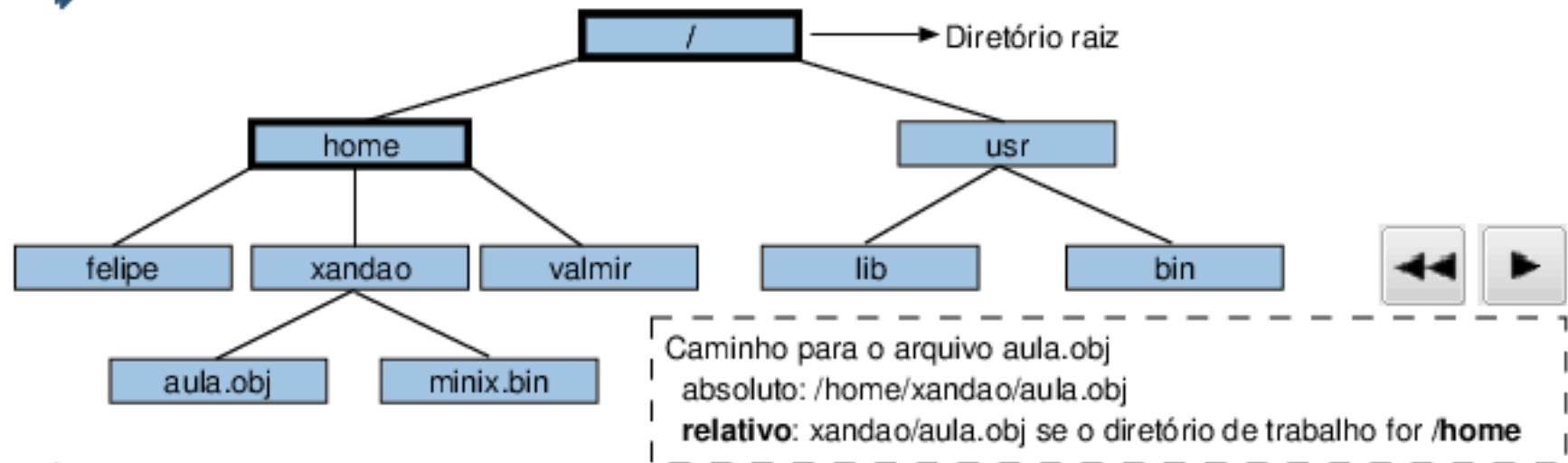


→ **Nome do caminho** de um arquivo:

- **Nome absoluto:** a partir do diretório raiz (inicia com /).
- **Nome relativo:** a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

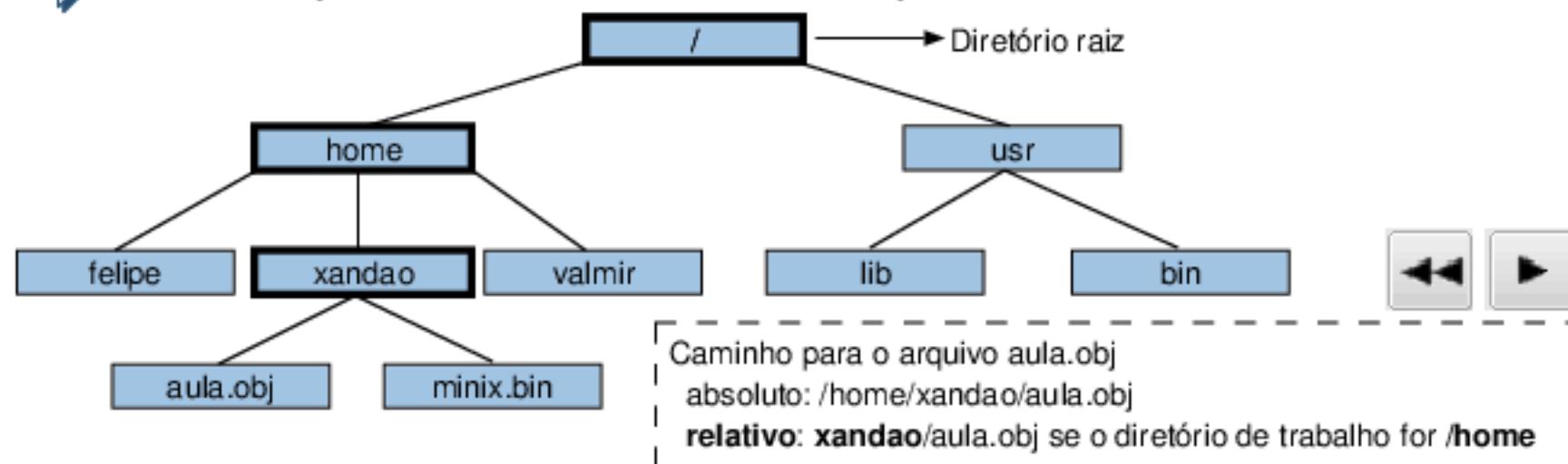


→ Nome do caminho de um arquivo:

- **Nome absoluto:** a partir do diretório raiz (inicia com /).
- **Nome relativo:** a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

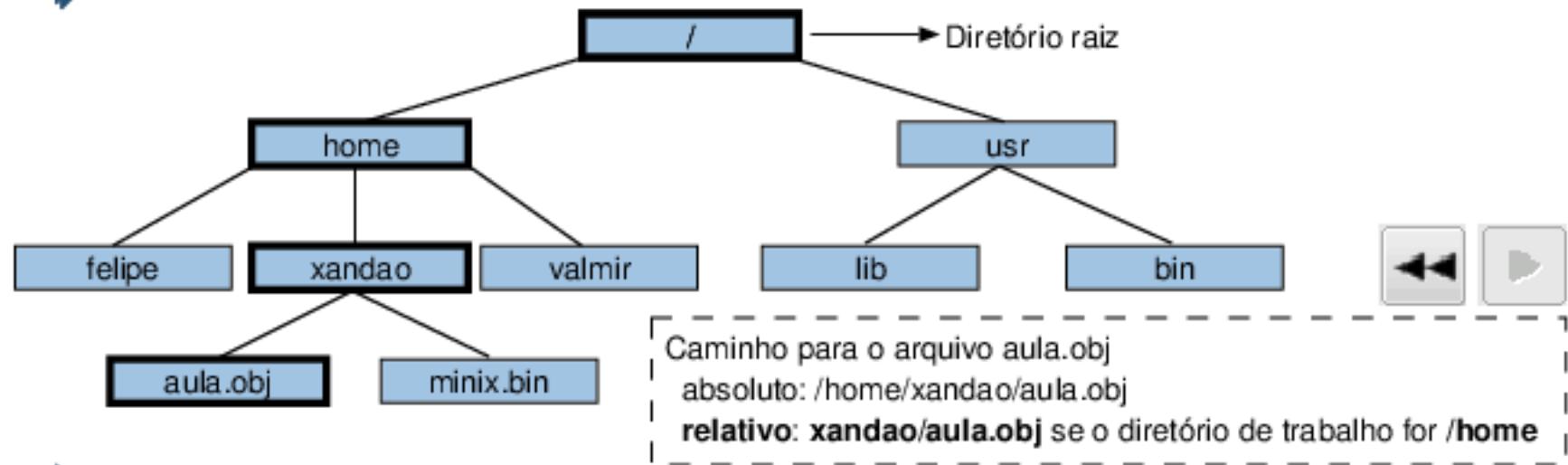


→ Nome do caminho de um arquivo:

- **Nome absoluto:** a partir do diretório raiz (inicia com /).
- **Nome relativo:** a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

→ A hierarquia de um sistema de arquivos também define uma árvore:

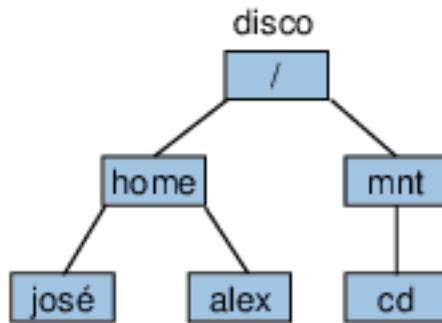


→ Nome do caminho de um arquivo:

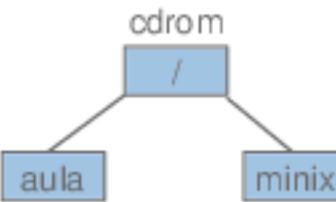
- **Nome absoluto:** a partir do diretório raiz (inicia com /).
- **Nome relativo:** a partir do diretório de trabalho.
- O arquivo é referenciado por um dos nomes de caminho.
- Os componentes do nome são separados por /.

Arquivos e diretórios

- ➡ Cada arquivo possui bits de proteção que definem a política de acesso ao arquivo.
- ➡ O sistema de arquivos que contém o sistema operacional é o único cujos arquivos são acessíveis via nome do caminho.
- ➡ Outros sistemas de arquivos não podem ser normalmente acessados:



Sistema de arquivos que contém o sistema operacional: os arquivos podem ser acessados.

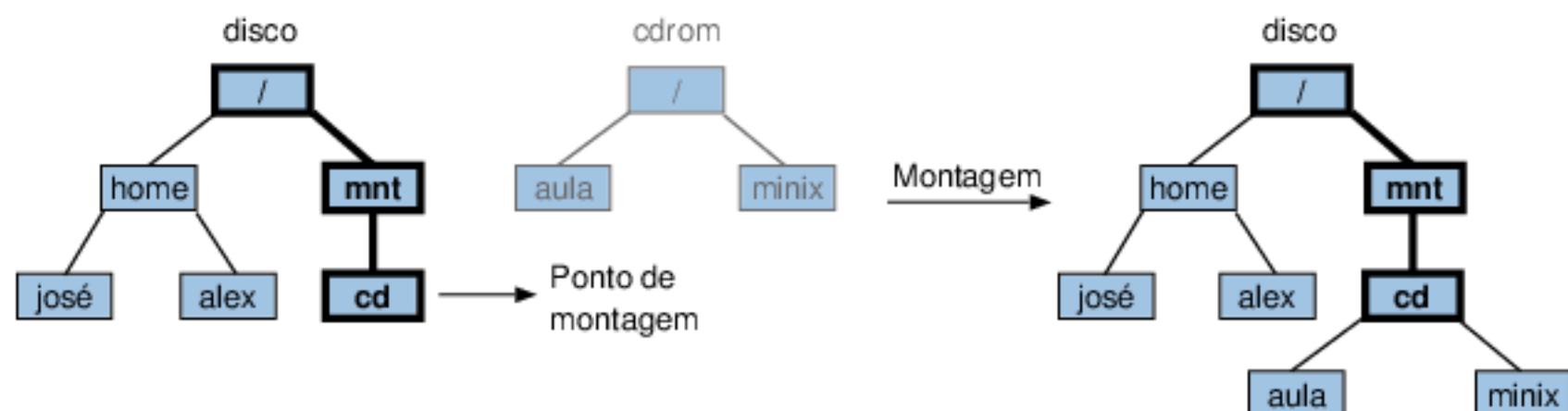


Sistema de arquivos de um CDROM que não contém o sistema operacional: os arquivos não podem ser acessados

Arquivos e diretórios

Montagem de um sistema de arquivos:

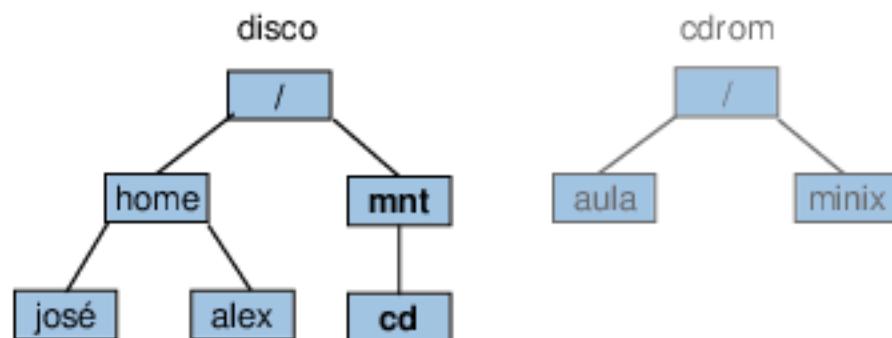
- Permite o acesso a outros sistemas de arquivos.
- O outro sistema é montado em um diretório, chamado de **o ponto de montagem**.
- O outro sistema é acessado a partir do ponto de montagem.



Arquivos e diretórios

► Montagem de um sistema de arquivos:

- Permite o acesso a outros sistemas de arquivos.
- O outro sistema é montado em um diretório, chamado de **o ponto de montagem**.
- O outro sistema é acessado a partir do ponto de montagem.



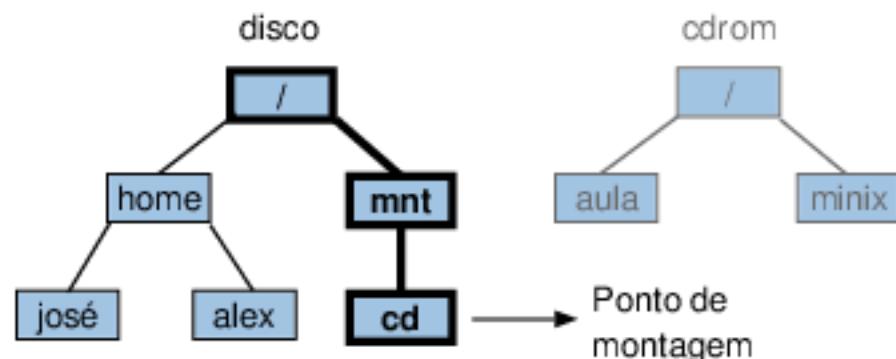
Inicialmente, somente os arquivos do sistema de arquivos que contém o sistema operacional (disco) podem ser acessados. Os arquivos do outro sistema de arquivos (cdrom) não podem ser acessados.



Arquivos e diretórios

Montagem de um sistema de arquivos:

- Permite o acesso a outros sistemas de arquivos.
- O outro sistema é montado em um diretório, chamado de o **ponto de montagem**.
- O outro sistema é acessado a partir do ponto de montagem.



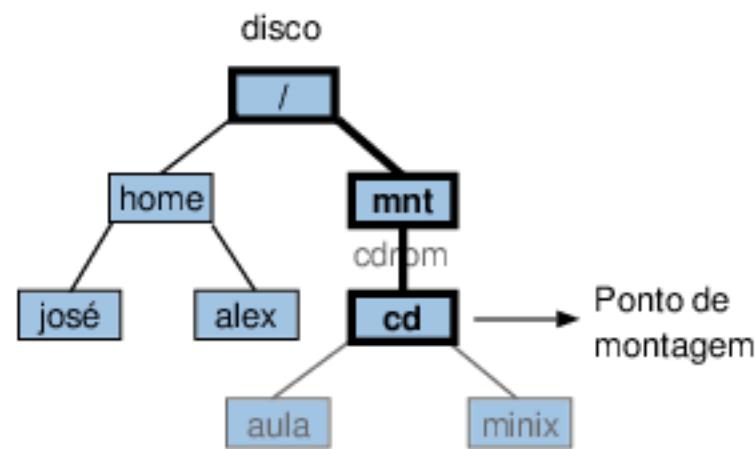
Para montar um sistema de arquivos num outro, primeiramente devemos definir um diretório, chamado de o **ponto de montagem**, a partir do qual os arquivos do sistema montado serão acessados. Neste exemplo, escolhemos o diretório `cd`, cujo nome de caminho é `/mnt/cd`.



Arquivos e diretórios

Montagem de um sistema de arquivos:

- Permite o acesso a outros sistemas de arquivos.
- O outro sistema é montado em um diretório, chamado de o **ponto de montagem**.
- O outro sistema é acessado a partir do ponto de montagem.



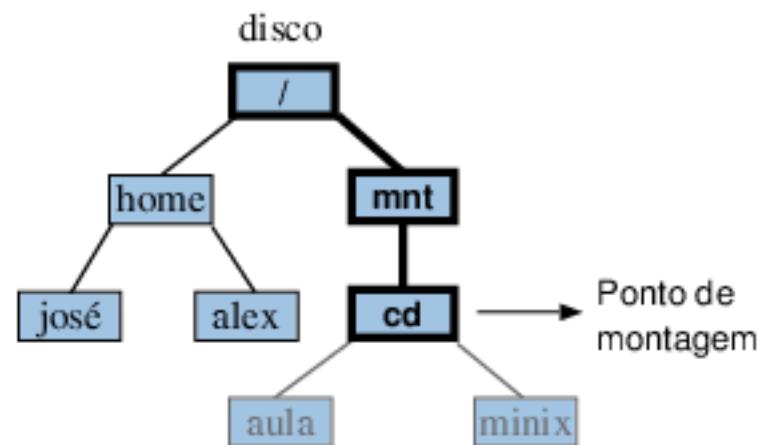
Depois de definido o ponto de montagem, o sistema de arquivos será montado neste diretório, no nosso exemplo, /mnt/cd, usando o comando do sistema chamado **mount**.



Arquivos e diretórios

Montagem de um sistema de arquivos:

- Permite o acesso a outros sistemas de arquivos.
- O outro sistema é montado em um diretório, chamado de **o ponto de montagem**.
- O outro sistema é acessado a partir do ponto de montagem.



Depois de montado o sistema de arquivos do CDROM, os arquivos serão acessados sempre a partir do diretório /mnt/cd, ao invés do diretório raiz /:

aula: acessado por /mnt/cd/aula;
minix: acessado por /mnt/cd/minix;



Arquivos e diretórios

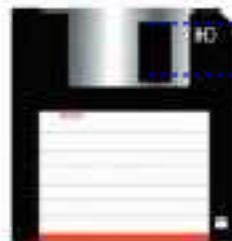


O sistema define um conjunto de **arquivos especiais**:

- O acesso aos dispositivos é similar ao de um arquivo comum.
- Os **arquivos especiais de bloco** são usados por dispositivos compostos por blocos acessados aleatoriamente.



O drive de disquete é identificado, nos sistemas UNIX, pelo arquivo especial cujo nome de caminho é **/dev/fd0**.



Um drive de disquete é um exemplo de um dispositivo composto por um conjunto de n blocos consecutivos. Estes blocos podem ser acessados aleatoriamente, isto é, você pode acessá-los em qualquer ordem, e não somente na ordem seqüencial 1, 2, ..., n, como, por exemplo, 1, 3, 4, 2, ao invés de 1, 2, 3, 4, quando n = 4.

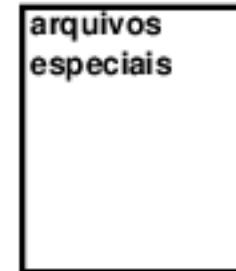
Arquivos e diretórios

→ O sistema define um conjunto de **arquivos especiais**:

- Os **arquivos especiais de caracteres** são usados por dispositivos modelados por um fluxo de caracteres.



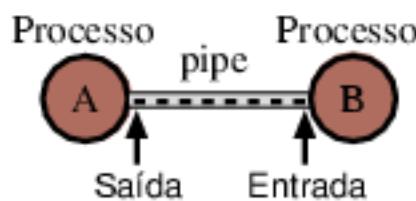
Fluxo de caracteres



Uma impressora é um exemplo de um dispositivo modelado por um fluxo de caracteres consecutivos, pois os caracteres são impressos na ordem em que eles foram gerados, e não é possível apagar de uma página impressa os caracteres já impressos. Por exemplo, após a impressão da palavra **arquivos**, não é possível retirar esta palavra da folha impressa.

Arquivos e diretórios

→ O sistema define um pseudoarquivo, chamado de **pipe**, que:



- Permite a troca de informações entre dois processos.
- Conecta a saída de um processo à entrada do outro.
- Os processos não sabem que estão usando um pipe.

Interpretador de comandos

- ➡ Principal interface entre o usuário em um terminal e o sistema.
- ➡ Um interpretador é iniciado sempre que um usuário usa o sistema.
- ➡ Ao iniciar, mostra um prompt (\$) que permite executar programas:

```
$date  
Qua Mar 22 16:02:25 UTC 2006  
$
```

- ➡ Para que o prompt retorne imediatamente, devemos adicionar & ao final do nome do programa:

\$mysort alunos.txt &	O usuário executou um programa de ordenação que demora para gerar os resultados.
[1] 2370	
\$date	Enquanto o programa mysort executa, o usuário
Qua Mar 22 16:02:25 UTC 2006	decidiu usar o programa date.
\$	
[1]+ Done	Depois de algum tempo, o programa mysort
\$	finalmente termina a sua execução.

Interpretador de comandos

- ➡ O Interpretador cria um processo filho para cada programa.
- ➡ O terminal do usuário é a entrada e a saída padrões do programa:
 - A entrada padrão dos programas é o teclado.
 - A saída padrão dos programas é o monitor.
- ➡ A entrada padrão pode ser lida de um arquivo usando <, e a saída padrão pode ser escrita em um arquivo usando >:

\$date > data.txt

Salva a data atual no arquivo data.txt.

\$sort < alunos.txt > salunos.txt

Lê o arquivo alunos.txt, e salva o arquivo ordenado em salunos.txt.

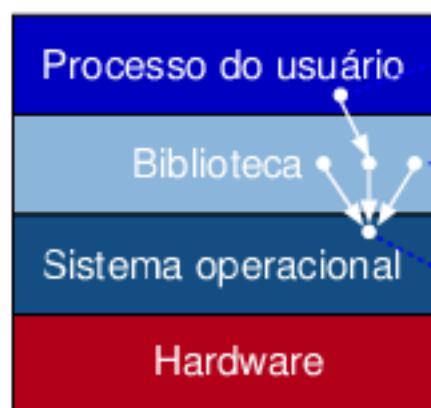
- ➡ Um pipe entre dois processos pode ser criado usando |:

\$cat arq1 arq2 arq3 | print

Concatena os arquivos arq1, arq2, e arq3, e envia o resultado para o programa print.

Chamadas ao sistema operacional

- ▶ Definem a interface entre o sistema operacional e os processos.
- ▶ Em geral, os processos usam chamadas a bibliotecas, pois:
 - Podemos definir uma interface padrão (POSIX).
 - Podemos definir várias funções para facilitar o uso de uma mesma chamada ao sistema.
 - O uso direto das chamadas é dependente da máquina, e de difícil implementação.



- O processo do usuário usa somente chamadas a uma biblioteca, pois estas são mais simples de usar do que as chamadas ao sistema operacional.
- A biblioteca lida com os detalhes da chamada, além de fornecer vários modos de usar uma mesma chamada.
- Não é desejável que o processo do usuário use diretamente uma chamada ao sistema operacional, pois a implementação desta chamada é complicada.

Chamadas ao sistema operacional

→ Para uma completa interface, são definidas chamadas para:

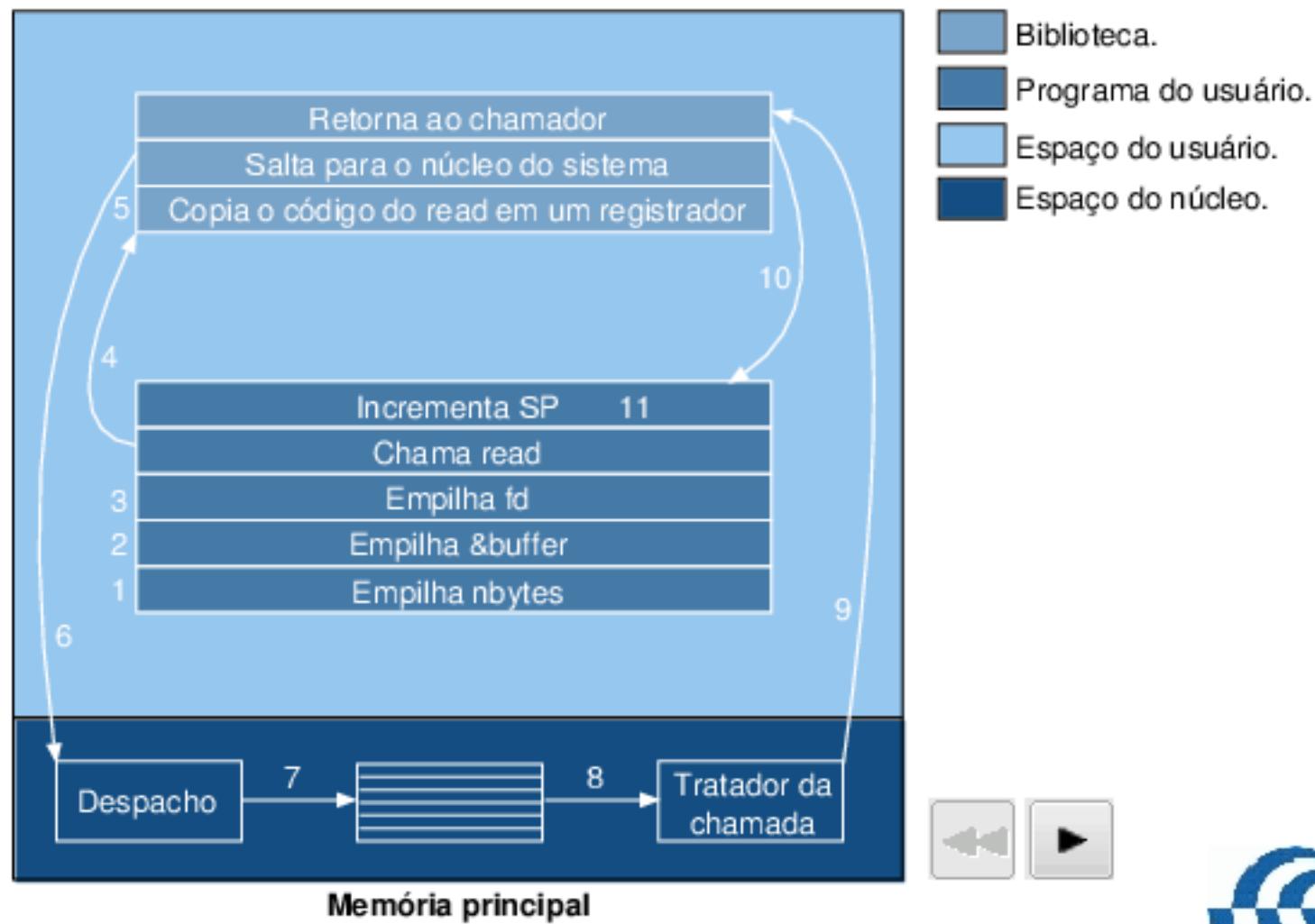
- Gerenciamento de processos.
- Gerenciamento do sistema de arquivos.
- Geração e tratamento de sinais.
- Gerenciamento do tempo do sistema.

→ Quando um processo usa uma chamada ao sistema:

- O procedimento da biblioteca que implementa a chamada ao sistema é executado.
- Uma instrução TRAP é usada para sair do modo usuário e entrar no modo núcleo.
- Um código especial no núcleo identifica qual é a chamada, e executa esta chamada.
- Depois da execução da chamada, o controle é retornado ao processo do usuário.

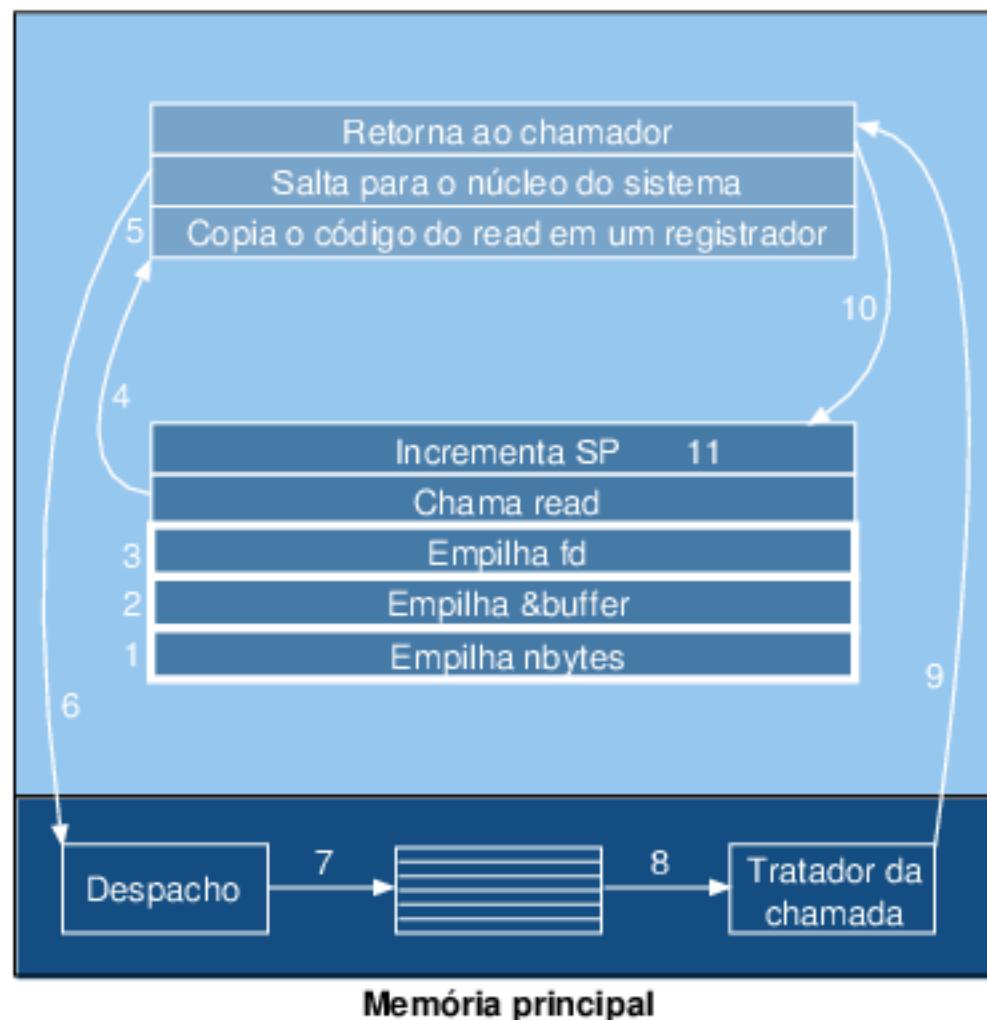
Chamadas ao sistema operacional

→ Exemplo de uma chamada: read(fd, &buffer, nbytes):



Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



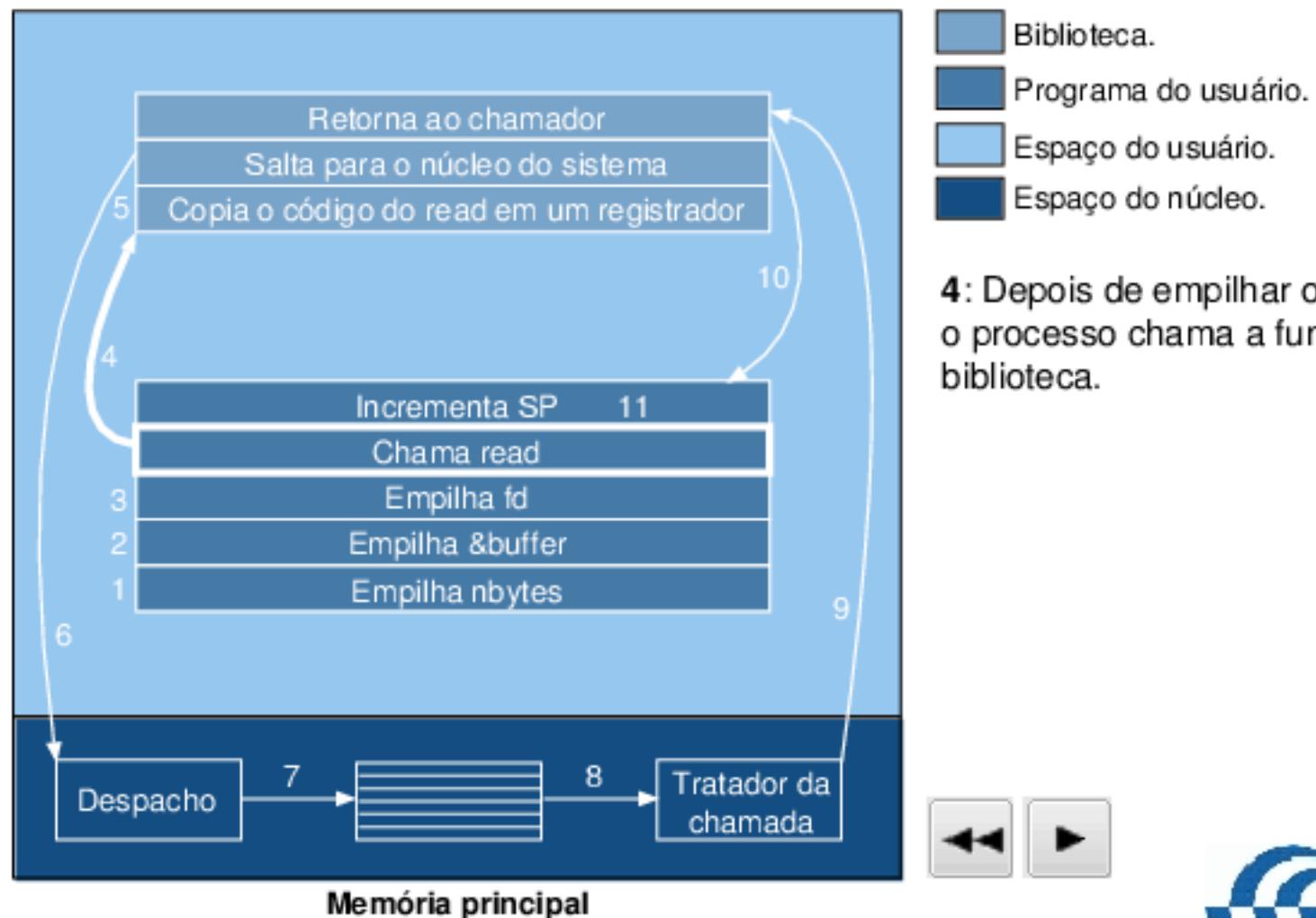
- Biblioteca.
- Programa do usuário.
- Espaço do usuário.
- Espaço do núcleo.

1-3: O processo do usuário empilha os parâmetros usados pela chamada ao procedimento `read` da biblioteca, que implementa uma interface para a chamada ao sistema `read`, para ler um conjunto de bytes do arquivo identificado por `fd`.



Chamadas ao sistema operacional

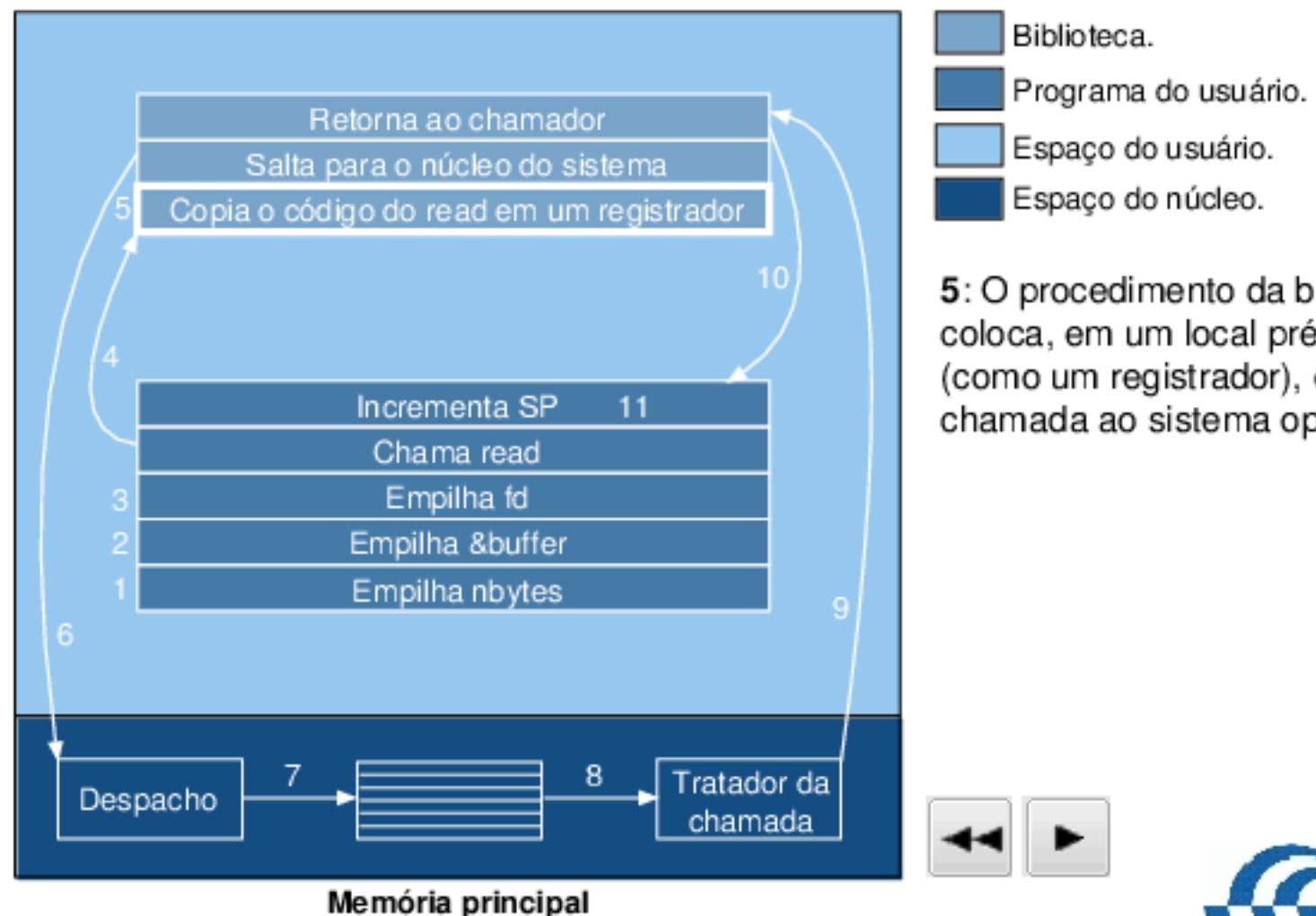
Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



4: Depois de empilhar os parâmetros, o processo chama a função read da biblioteca.

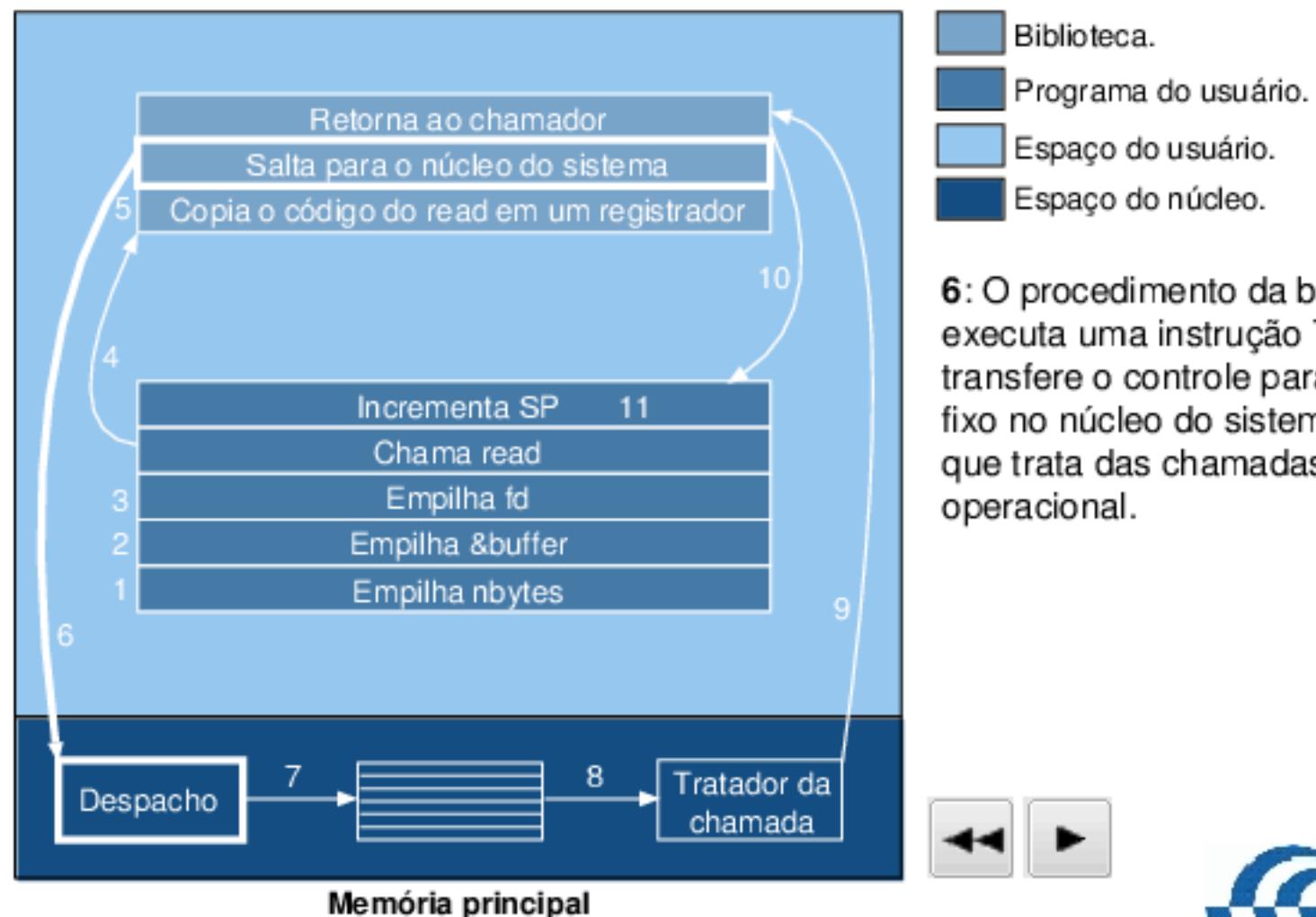
Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:

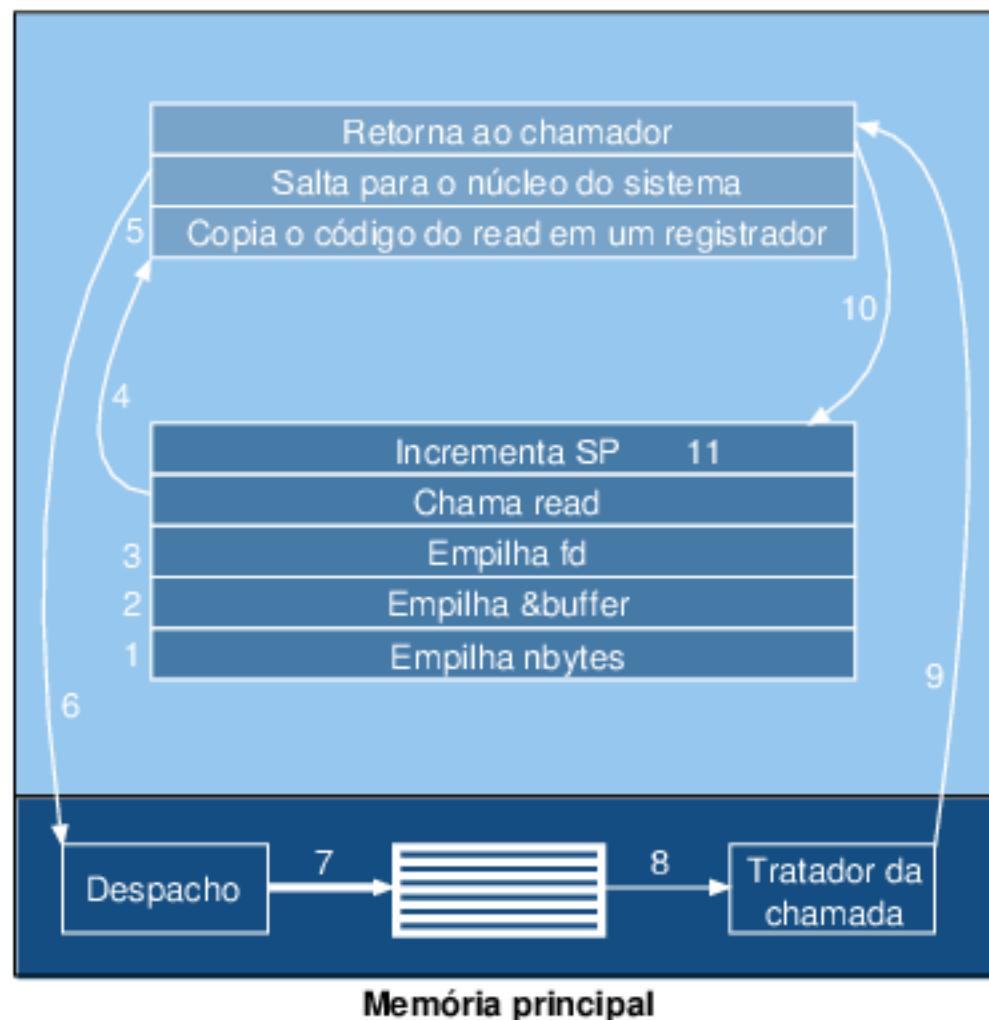


6: O procedimento da biblioteca agora executa uma instrução TRAP, que transfere o controle para um endereço fixo no núcleo do sistema operacional, que trata das chamadas ao sistema operacional.



Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



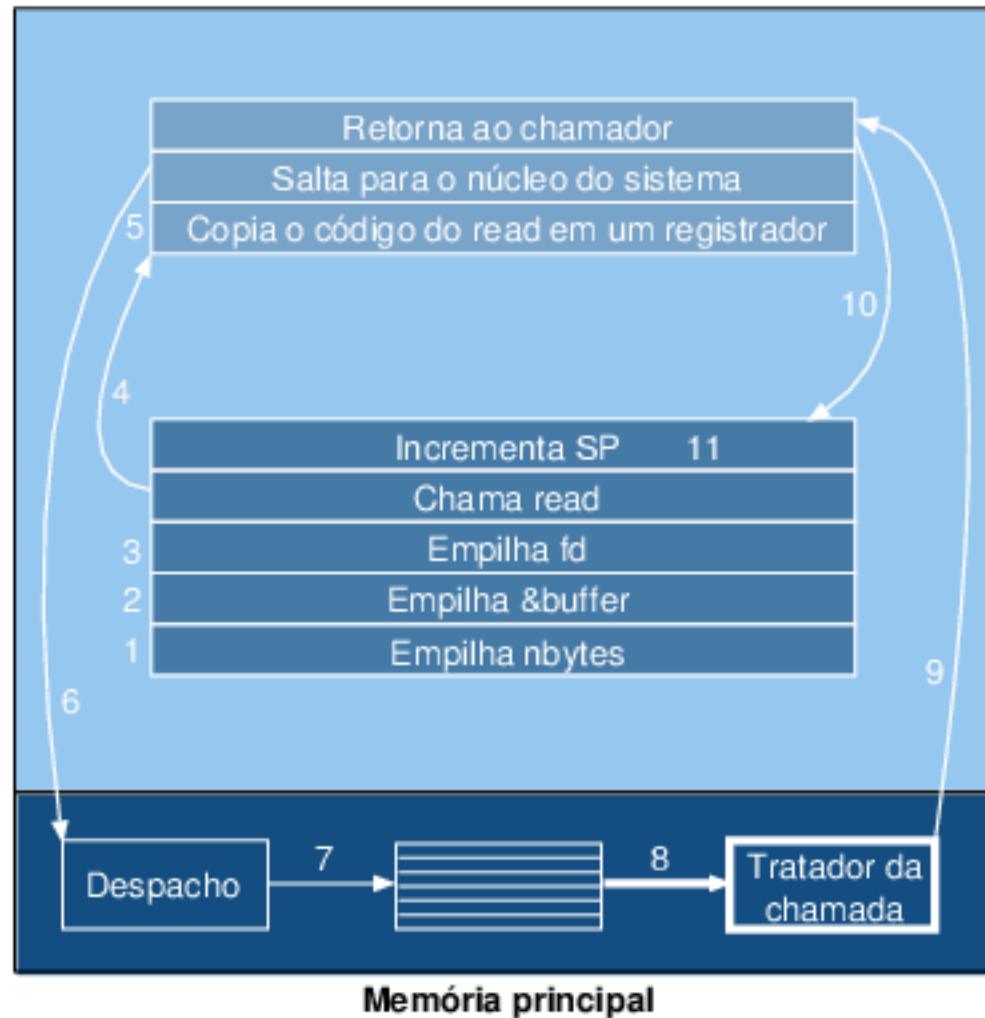
- Biblioteca.
- Programa do usuário.
- Espaço do usuário.
- Espaço do núcleo.

7: O código do núcleo então obtém o endereço do procedimento que trata da chamada ao sistema definida pelo código dado, usando este código como um índice numa tabela com os endereços dos procedimentos que implementam as chamadas.



Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



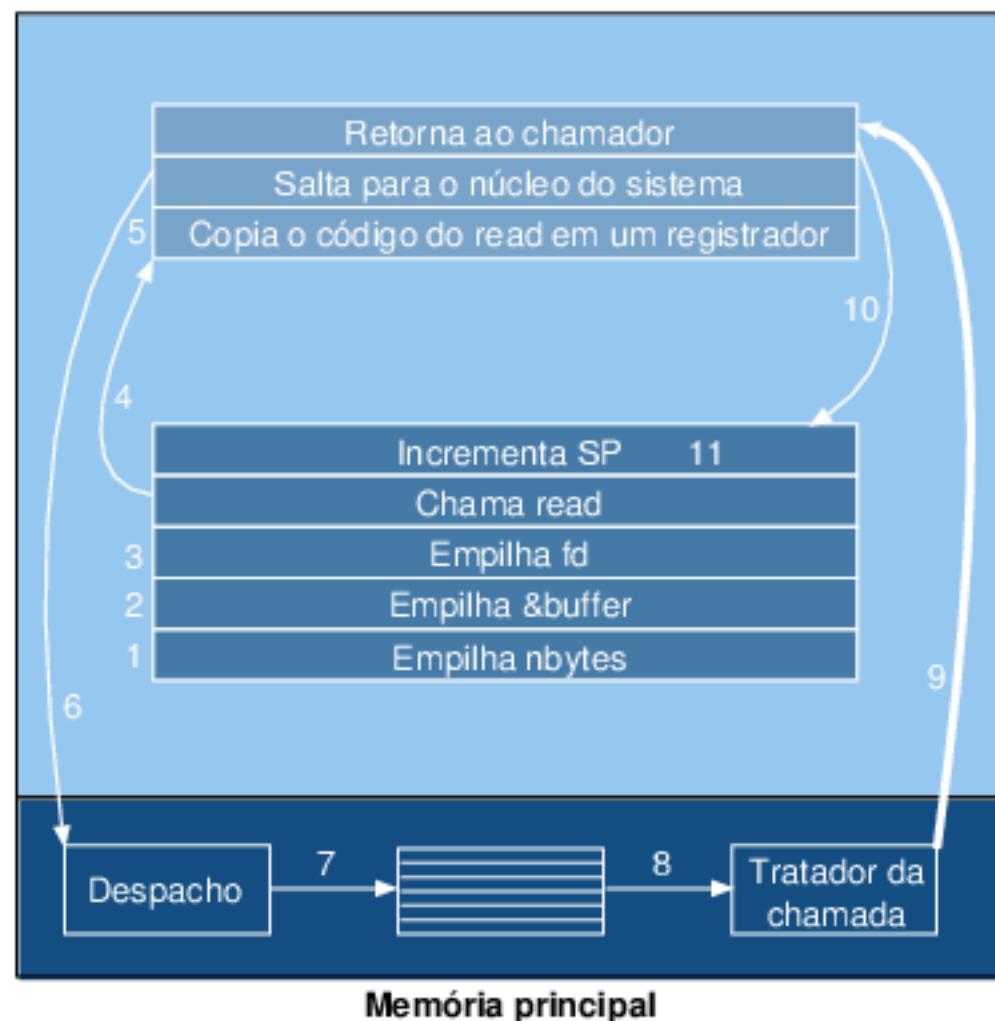
- Biblioteca.
- Programa do usuário.
- Espaço do usuário.
- Espaço do núcleo.

8: Usando o endereço obtido, o núcleo chama este procedimento, denominado o **tratador da chamada**, que executa a chamada ao sistema read.



Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



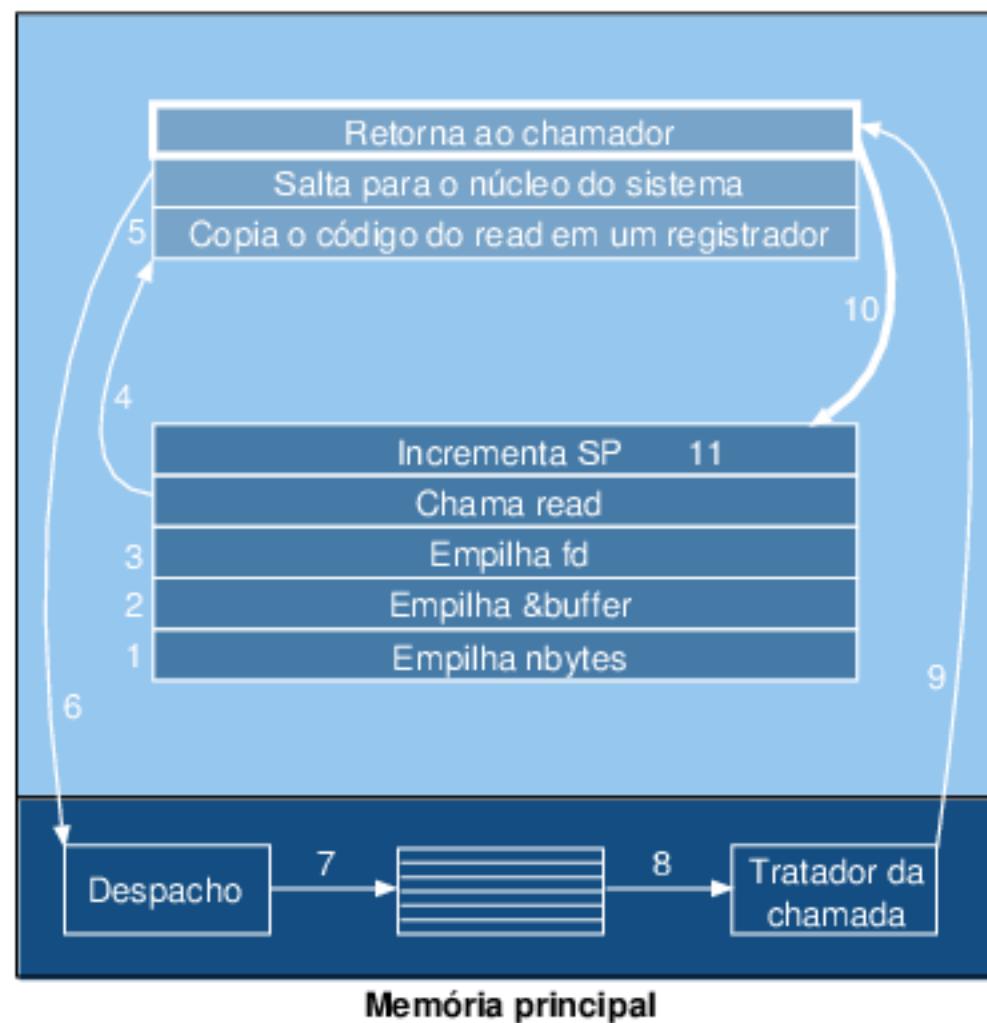
- Biblioteca.
- Programa do usuário.
- Espaço do usuário.
- Espaço do núcleo.

9: Depois de executar a chamada, o núcleo do sistema passará o controle ao procedimento da biblioteca, se o tratador da chamada pôde executar a chamada ao sistema sem precisar esperar a ocorrência de um evento externo, como o término de alguma operação de E/S.



Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



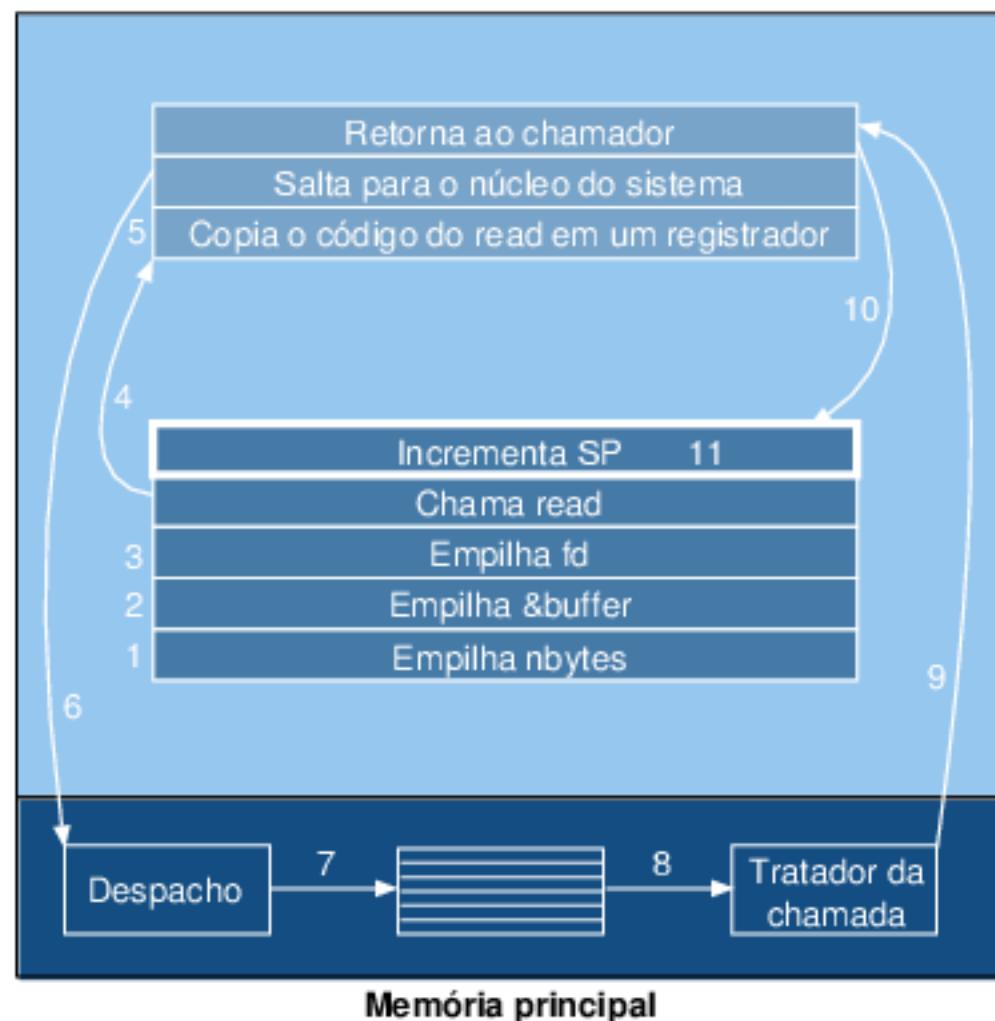
- Biblioteca.
- Programa do usuário.
- Espaço do usuário.
- Espaço do núcleo.

10: O procedimento da biblioteca faz algumas outras tarefas necessárias para concluir sua execução, e retorna o controle ao processo do usuário.



Chamadas ao sistema operacional

Exemplo de uma chamada: `read(fd, &buffer, nbytes)`:



- Biblioteca.
- Programa do usuário.
- Espaço do usuário.
- Espaço do núcleo.

11: O processo do usuário finaliza a chamada à biblioteca incrementando o registrador SP, com o objetivo de retirar da pilha todos os parâmetros empilhados antes de chamar o procedimento da biblioteca.

