

Sistemas Embarcados

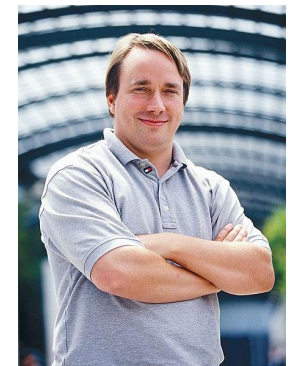
Linux básico

Introdução ao desenvolvimento com Linux

- Conteúdo:
 - Metodologia/Filosofia de desenvolvimento com o Linux/UNIX
 - Licenças de software livre (GPL e LGPL)
 - Organização da estrutura de diretórios
 - Comandos básicos do Linux
 - Obtendo informações sobre o sistema
 - Instalação de programas

O que é Linux?

- Variante *free-software* do Unix, criada por Linus Torvalds em 1990



O que é Linux?

- Disponibilizado em distribuições:



- A diferença entre as distribuições está no conjunto de utilitários oferecidos

O que é Linux?

- O Linux é apenas um bloco de um sistema operacional completo e livre
- Linux = **kernel** (núcleo) do sistema operacional GNU
- Os mais ortodoxos chamam estas distribuições de GNU/Linux: sistema operacional livre com kernel Linux



Como o kernel Linux opera?

- O kernel toma controle do computador
- Gerencia o processador: sistema multi-usuário, multi-processo, multi-processado (*scheduler*)
- Gerencia a memória com precisão: sana as demandas de memória, gerencia o espaço de troca (*swap*)

Como o kernel Linux opera?

- Gerencia dispositivos:
 - Suporta milhares de dispositivos: *drivers*
 - Para não carregar demais o kernel, carrega *drivers* sobre demanda: os módulos
- Gerencia sistemas de arquivos: reconhece vários tipos
 - FAT32
 - NTFS
 - EXT2, EXT3 e EXT4
- Segurança: sistema de permissões para um ambiente multi-usuário

Como o kernel Linux opera?

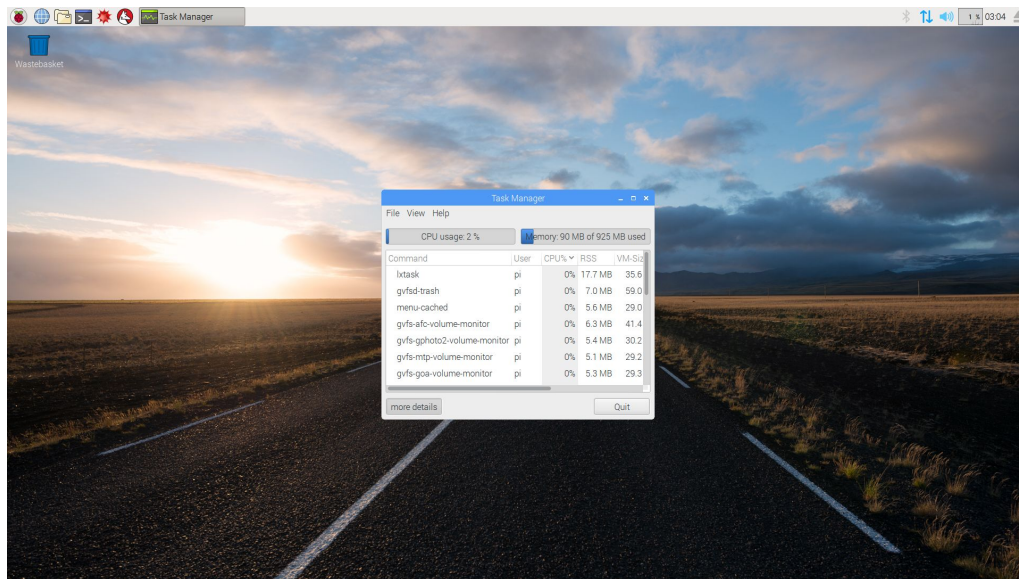
- Em síntese:
 - O kernel inicia quando se liga o computador
 - O kernel gerencia programas
 - Ele permite comunicação simples e eficiente com o hardware

Distribuições Linux

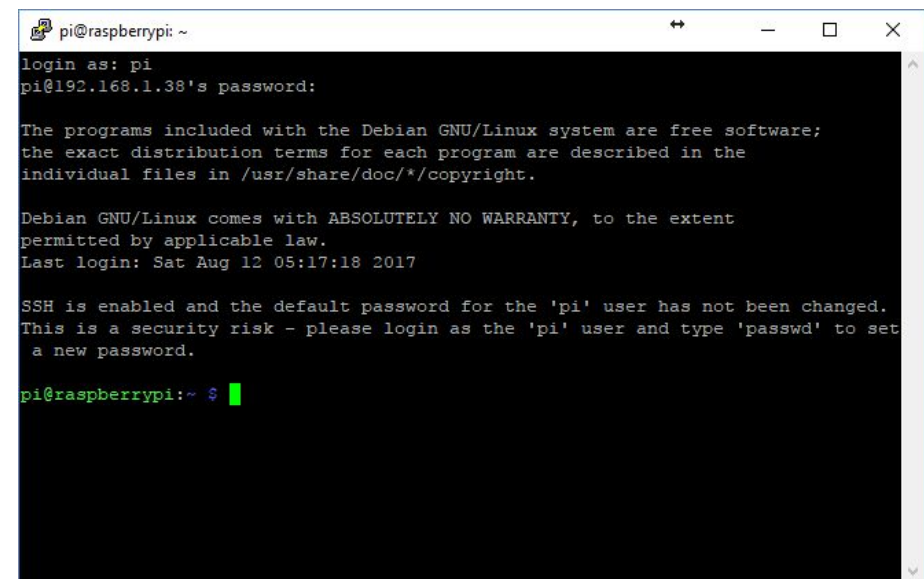
- Sistema operacional possui/pode ter:
 - Kernel
 - Interfaces gráficas com o usuário
 - Utilitários administrativos
 - Aplicações
 - Ferramentas de programação

Distribuições Linux

- Sistema operacional possui/pode ter:
 - Kernel
 - Interfaces gráficas com o usuário



Com interface gráfica



Sem interface gráfica

Distribuições Linux

- Sistema operacional possui/pode ter:
 - Kernel
 - Interfaces gráficas com o usuário
 - Utilitários administrativos
 - Aplicações
 - Ferramentas de programação



Filosofia de desenvolvimento com Linux

- Simplicidade:
 - Ferramentas pequenas, simples e fáceis de entender
 - KISS: Keep It Small and Simple
 - Programas complexos são mais suscetíveis a bugs

Filosofia de desenvolvimento com Linux

- Foco:
 - O programa executa a tarefa com perfeição
 - Manter um programa responsável por muitas tarefas é difícil
 - Modularização e composição são frequentes

Filosofia de desenvolvimento com Linux

- Reuso:
 - Não reinvente a roda!
 - Procure disponibilizar suas aplicações na forma de bibliotecas.

Filosofia de desenvolvimento com Linux

- Filtros:
 - Muitas das aplicações do Linux são filtros;
 - Isso permite a combinação de programas simples em um sistema complexo.

Filosofia de desenvolvimento com Linux

- Formato de arquivo aberto:
 - Uma das razões de sucesso (e ódio também);
 - Arquivos de configuração e dados são codificados em ASCII;
 - Maior liberdade aos usuários.

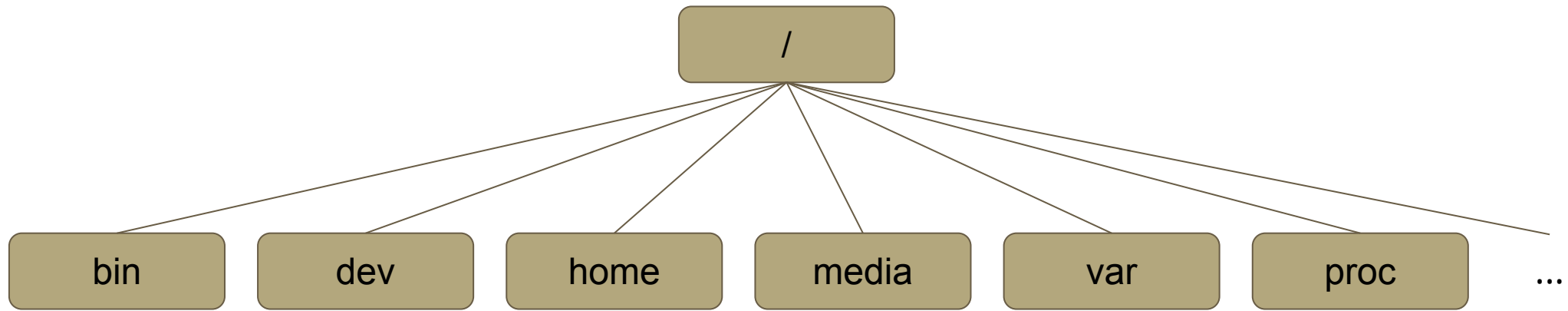
Filosofia de desenvolvimento com Linux

- Flexibilidade:
 - Não se pode prever o que um usuário deseja com um programa
 - Procure ser flexível:
 - evite limites nos tamanhos dos campos
 - tenha em mente um ambiente de rede
 - Documente bem o seu programa

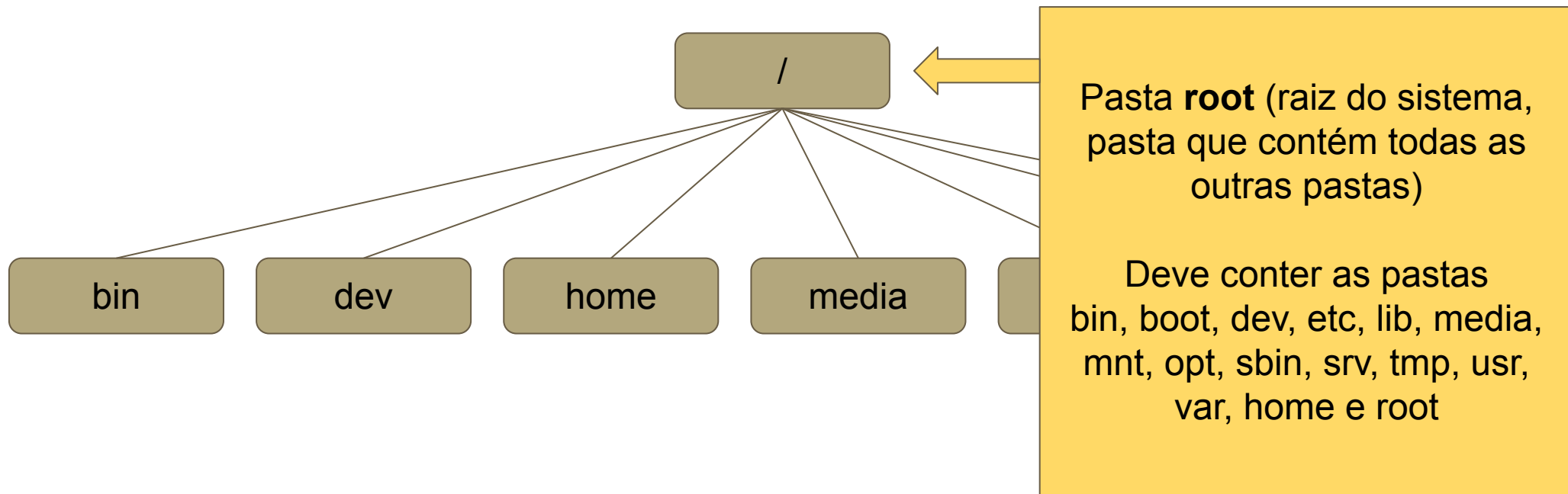
Estrutura de diretórios do Linux

- Nos sistemas de arquivos do Unix podemos encontrar:
 - Arquivos
 - Programas
 - Processos
 - Dispositivos de hardware
 - Canais de comunicação entre processos
 - Segmentos de memória compartilhada

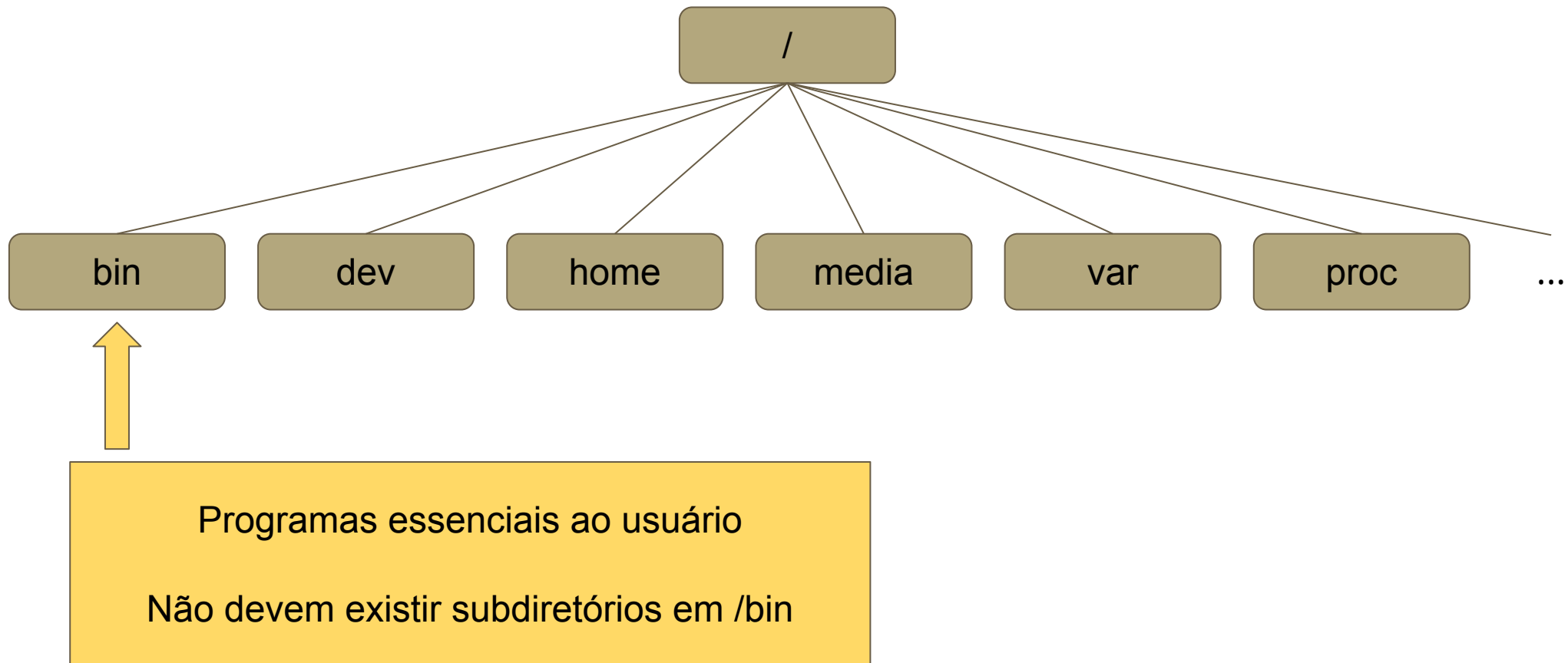
Estrutura de diretórios do Linux



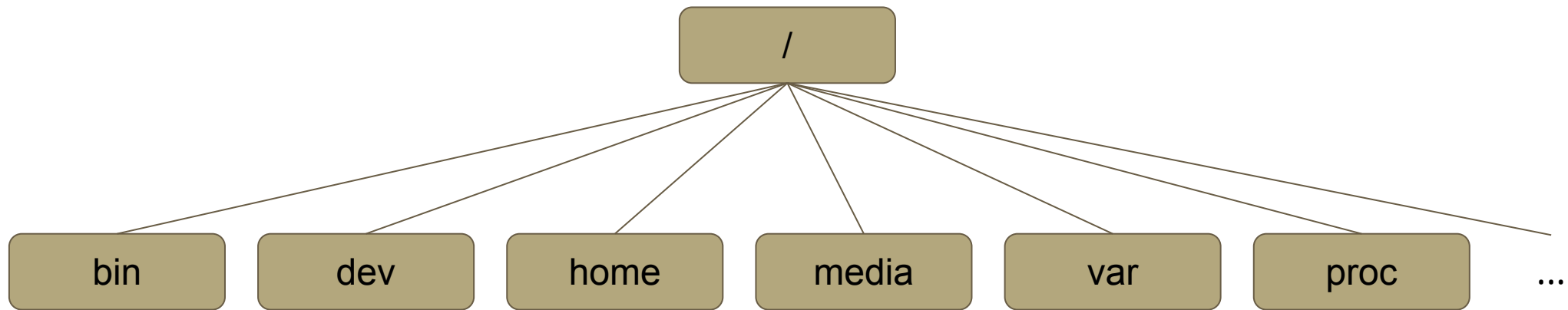
Estrutura de diretórios do Linux



Estrutura de diretórios do Linux



Estrutura de diretórios do Linux

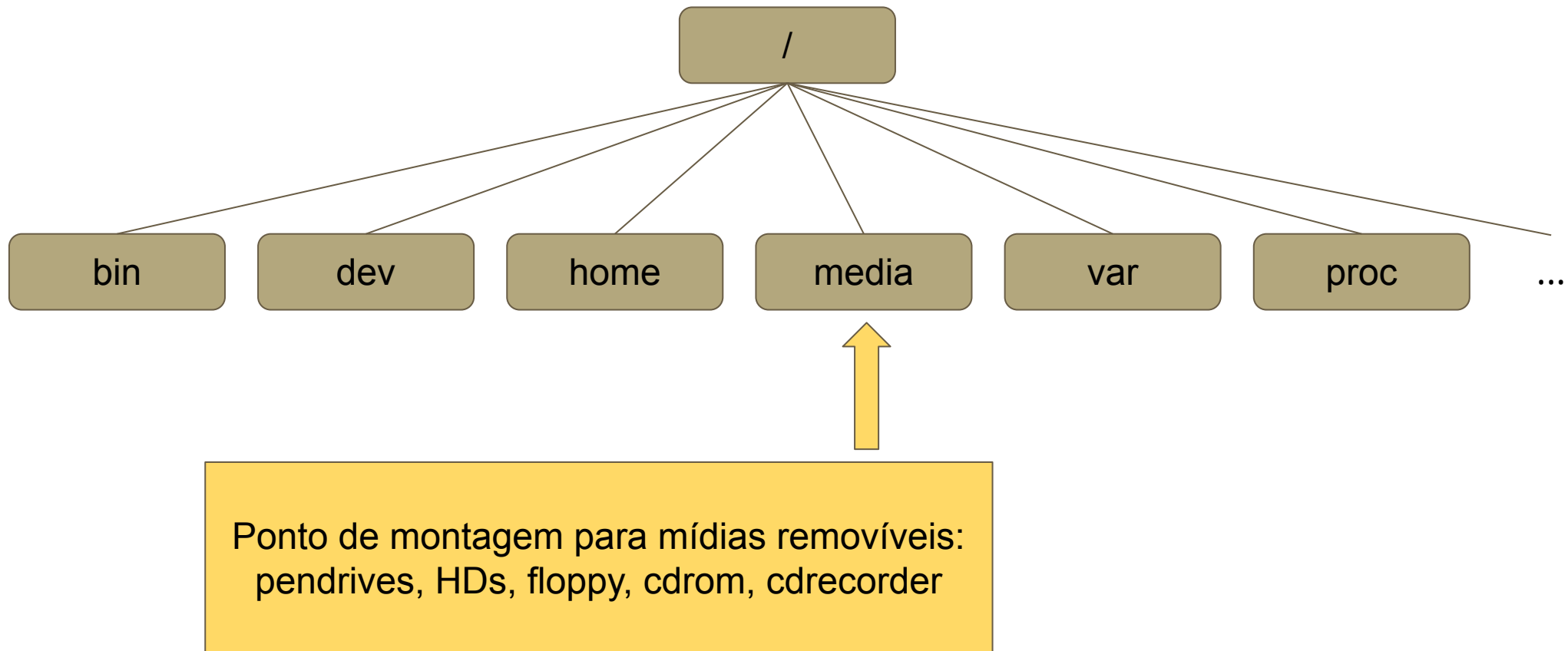


Contém arquivos de dispositivos (hardware)

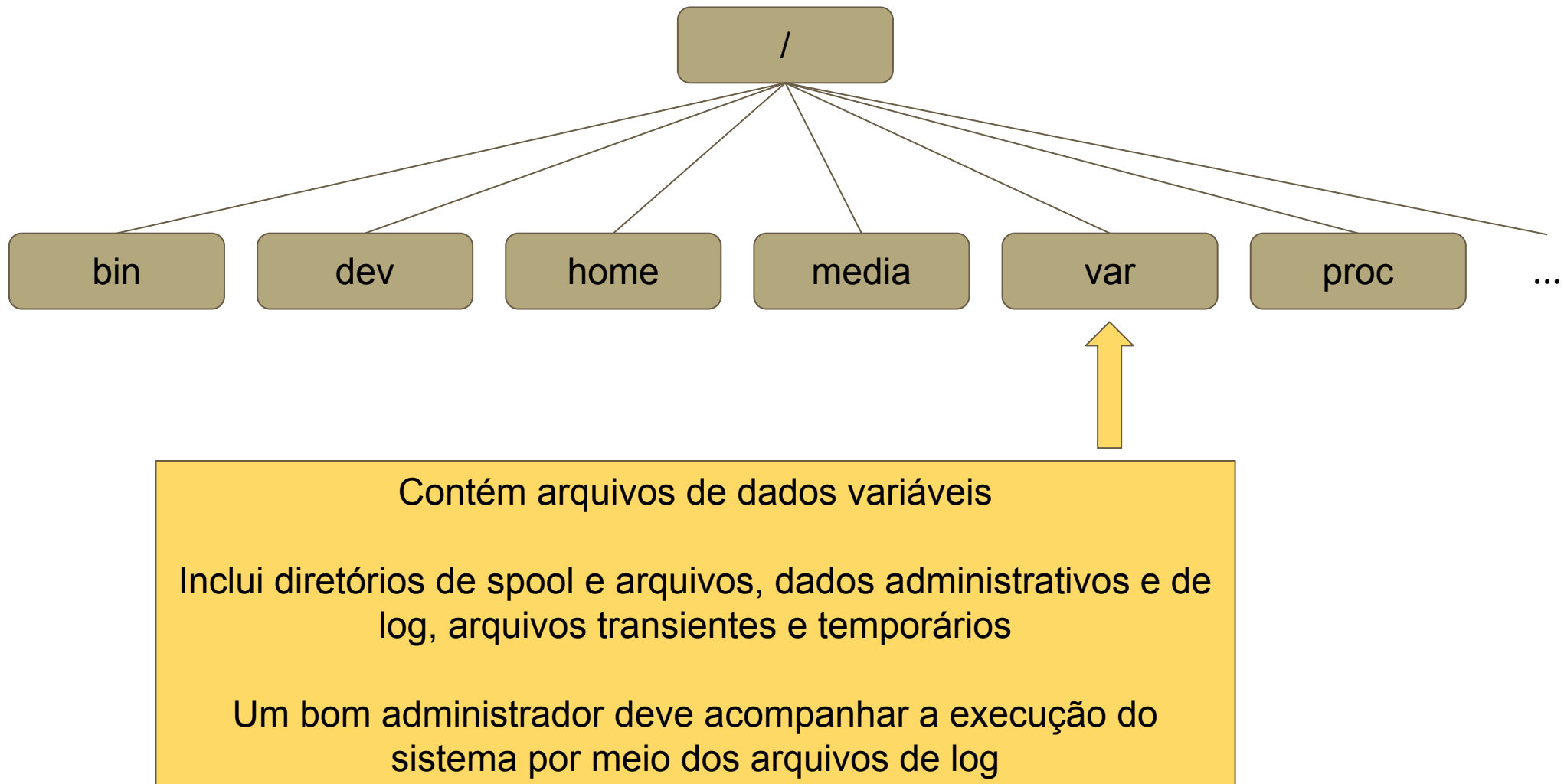
Comunicação com o hardware é feita por
leitura e escrita em arquivos

Programadores procuram usar uma mesma
interface para resolver diversos problemas

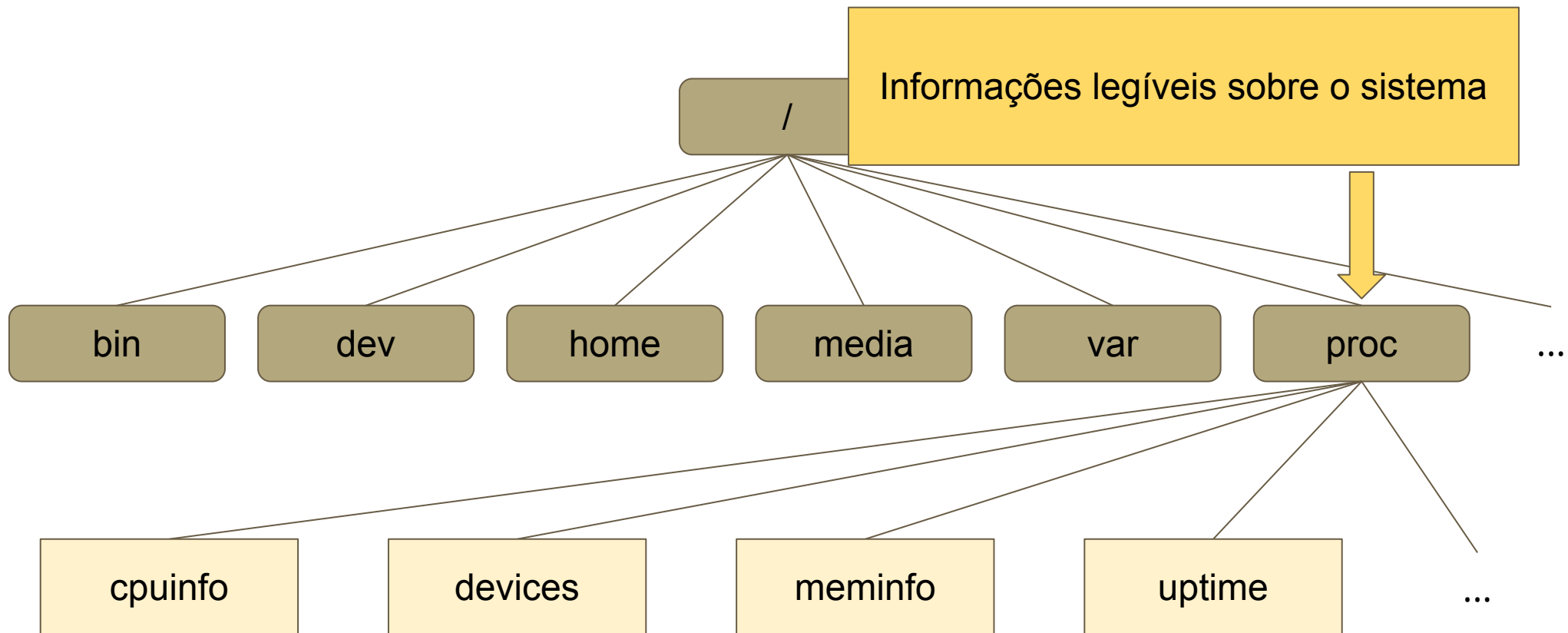
Estrutura de diretórios do Linux



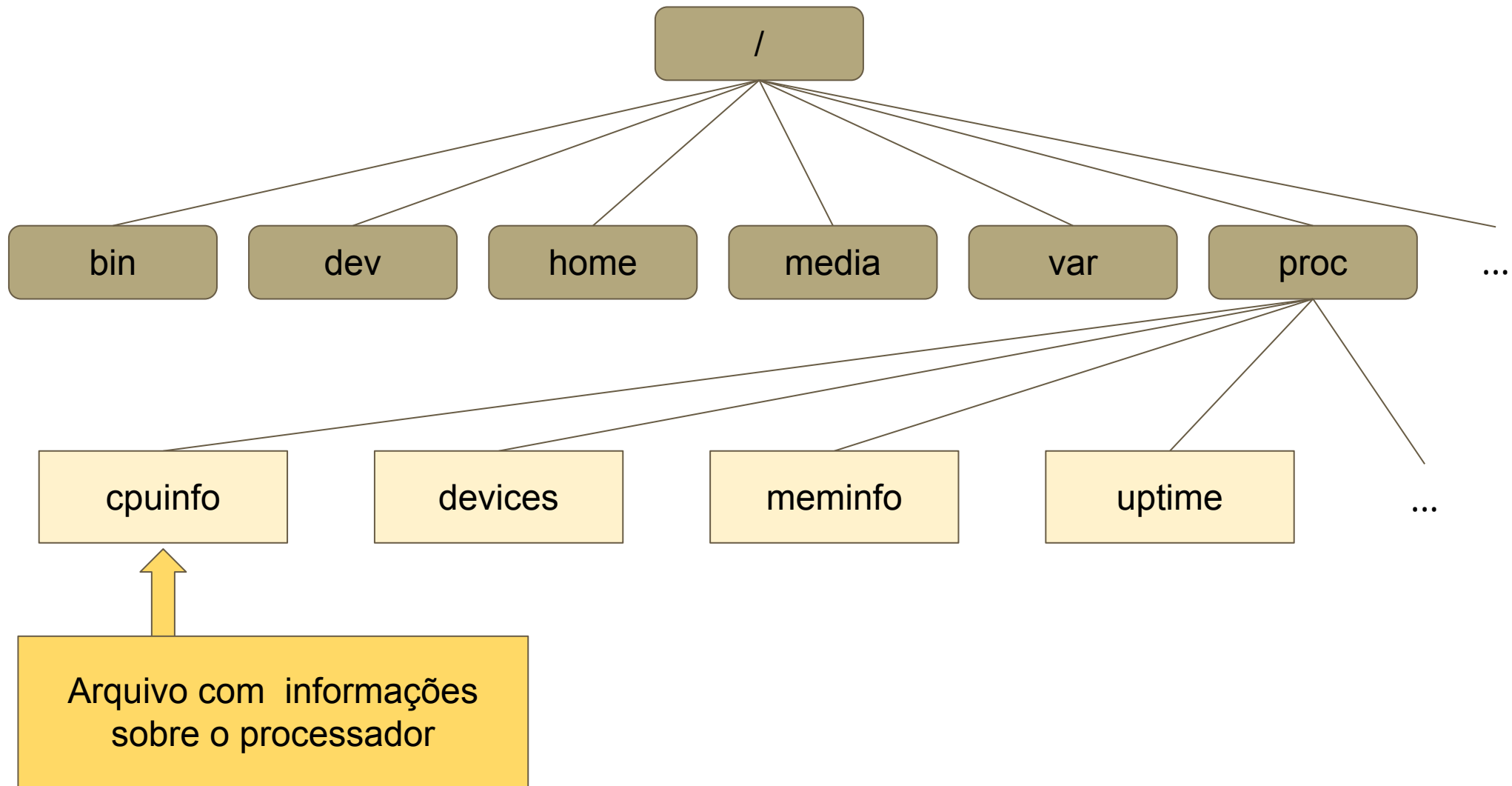
Estrutura de diretórios do Linux



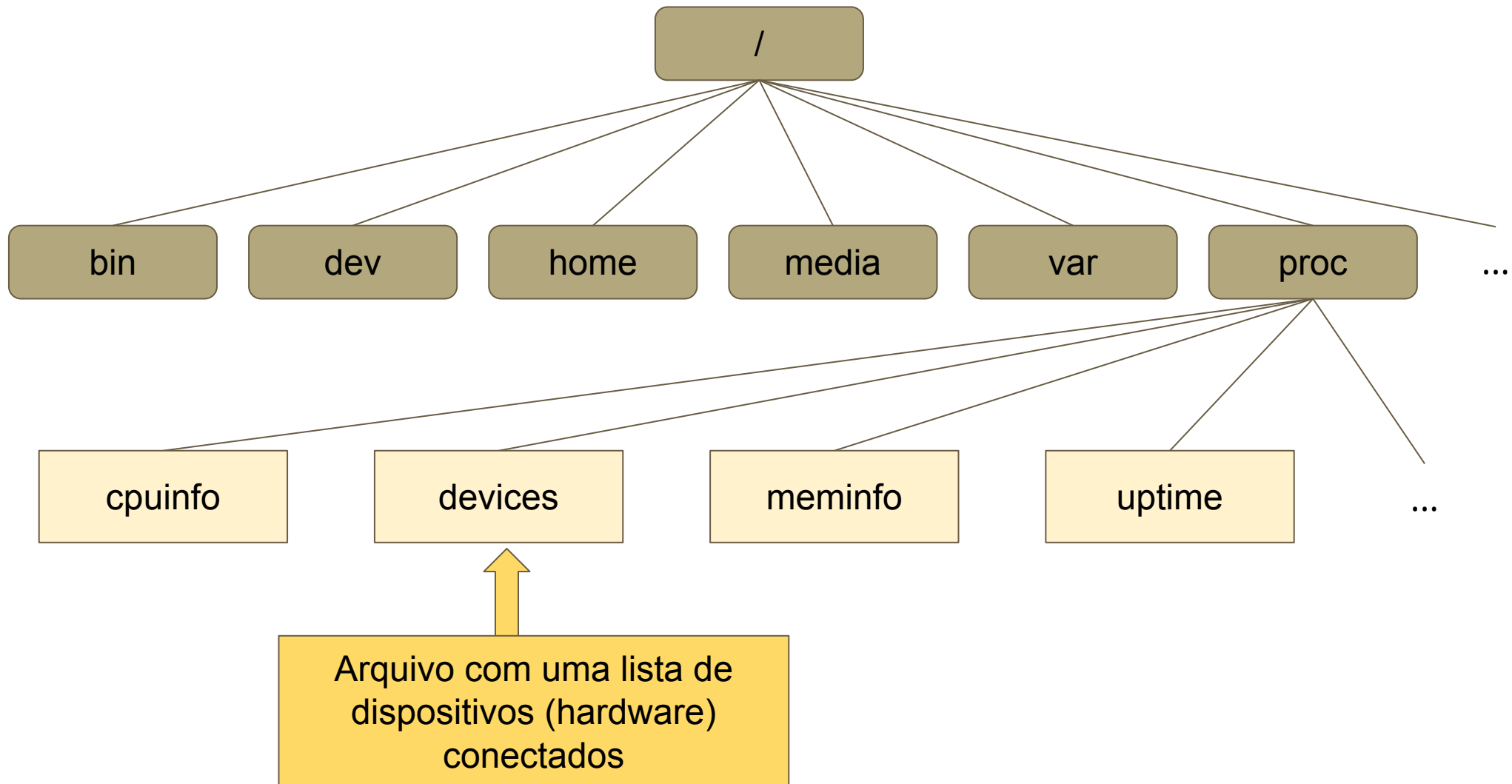
Estrutura de diretórios do Linux



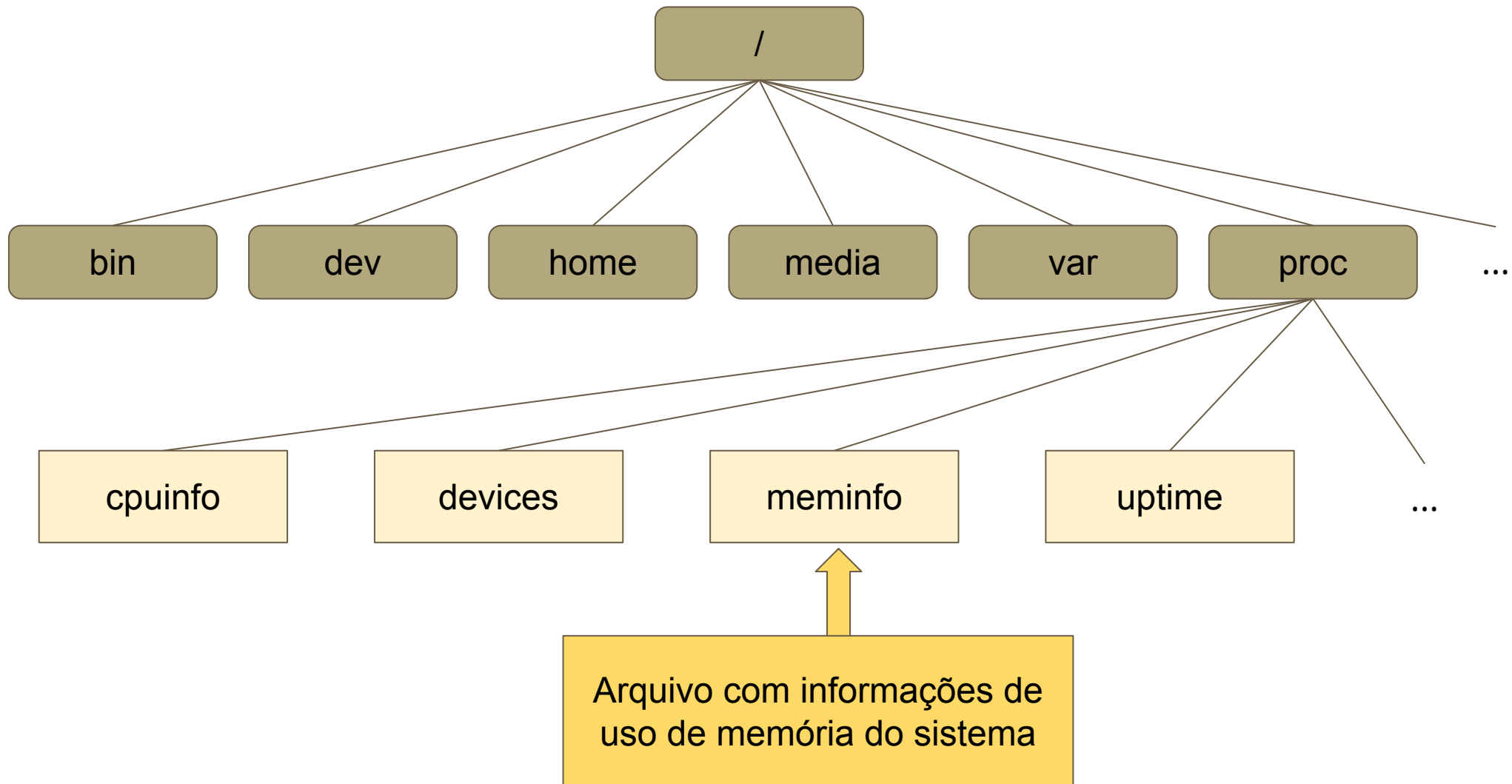
Estrutura de diretórios do Linux



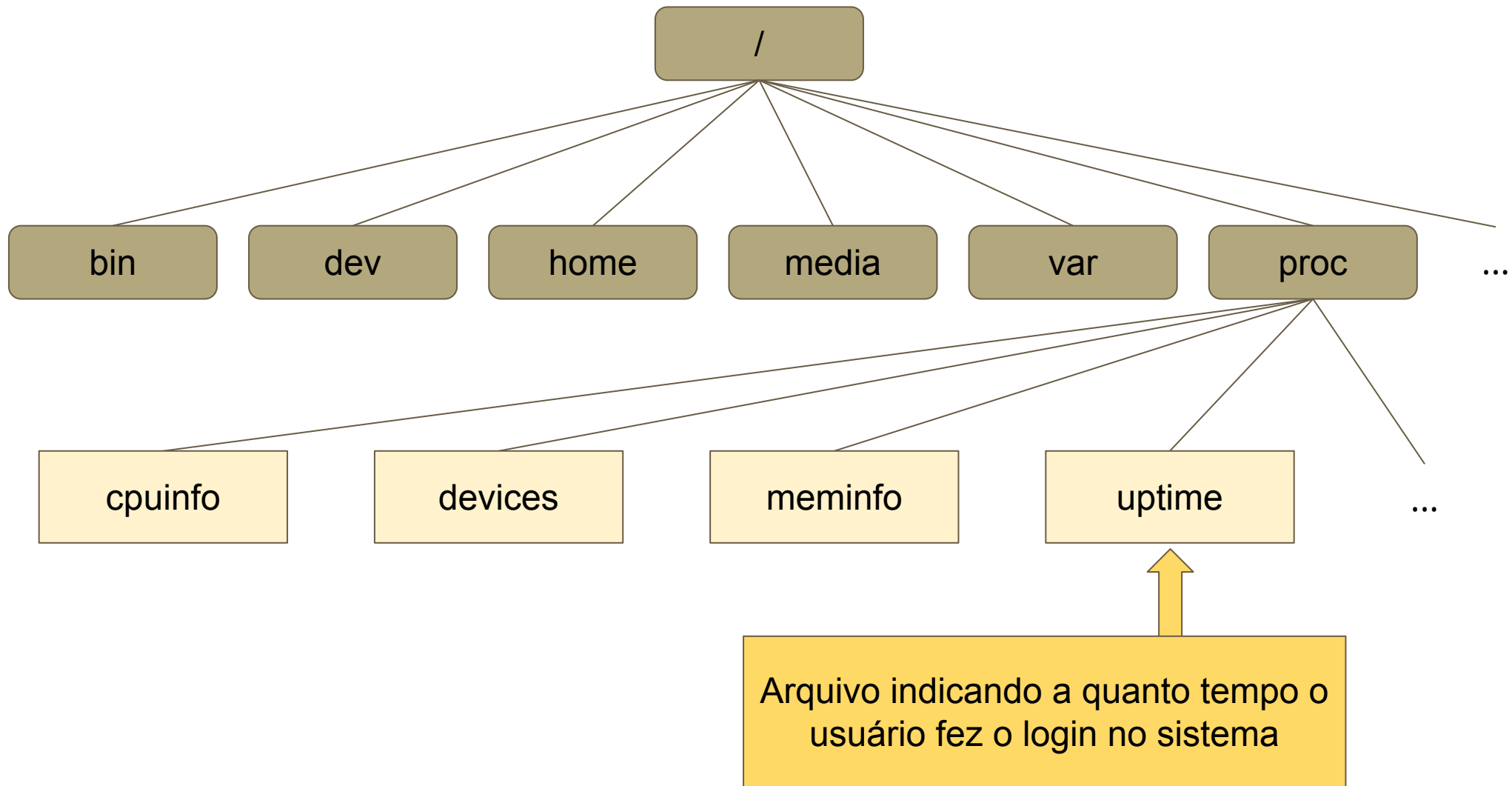
Estrutura de diretórios do Linux



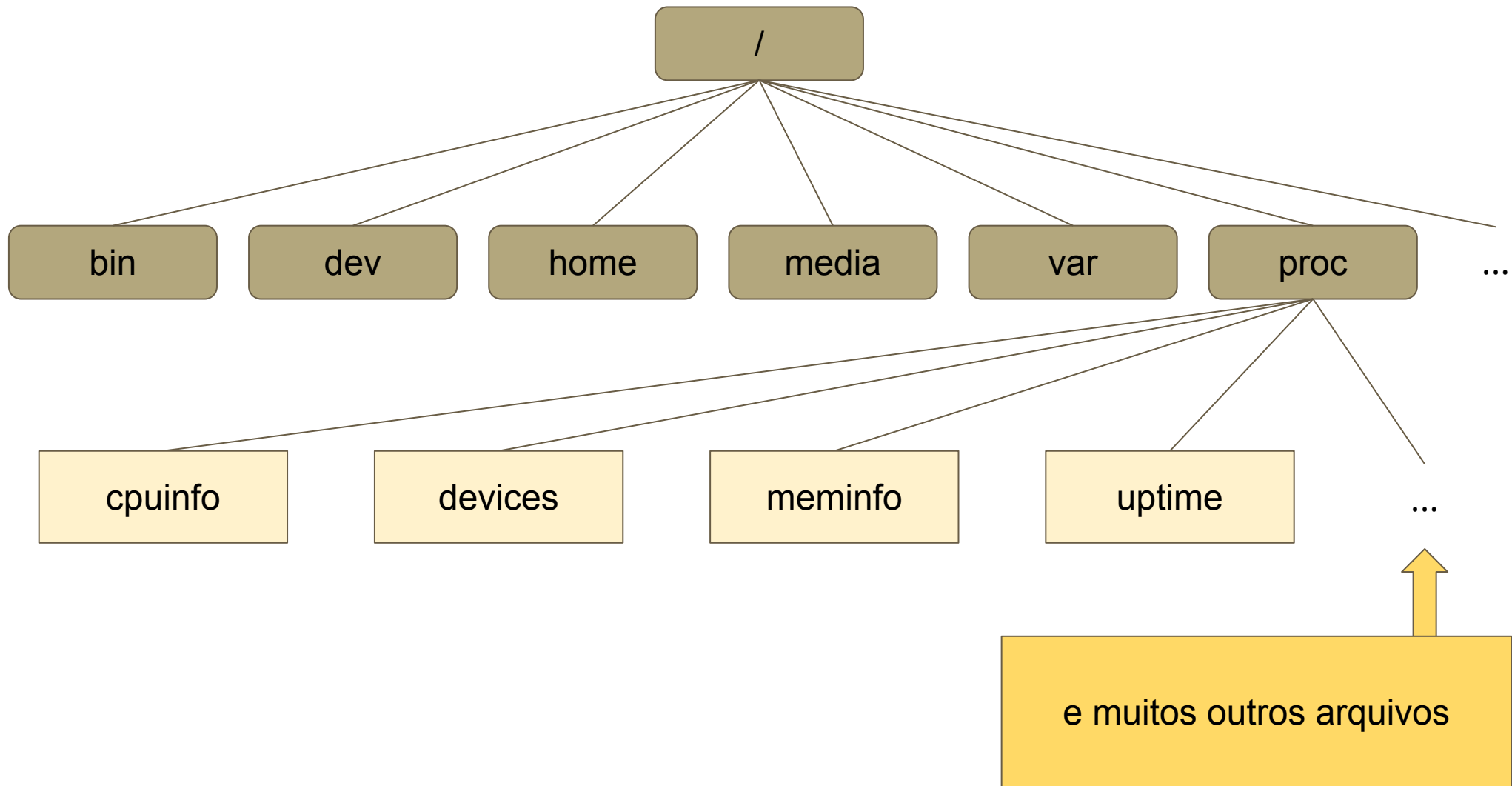
Estrutura de diretórios do Linux



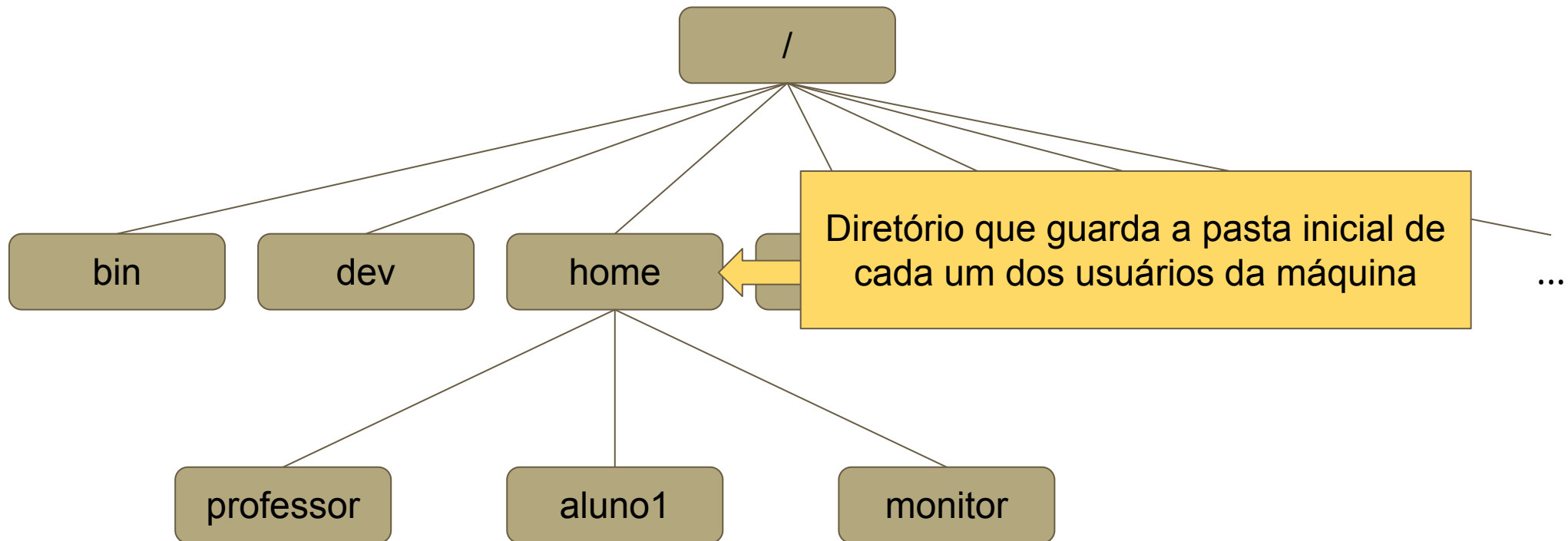
Estrutura de diretórios do Linux



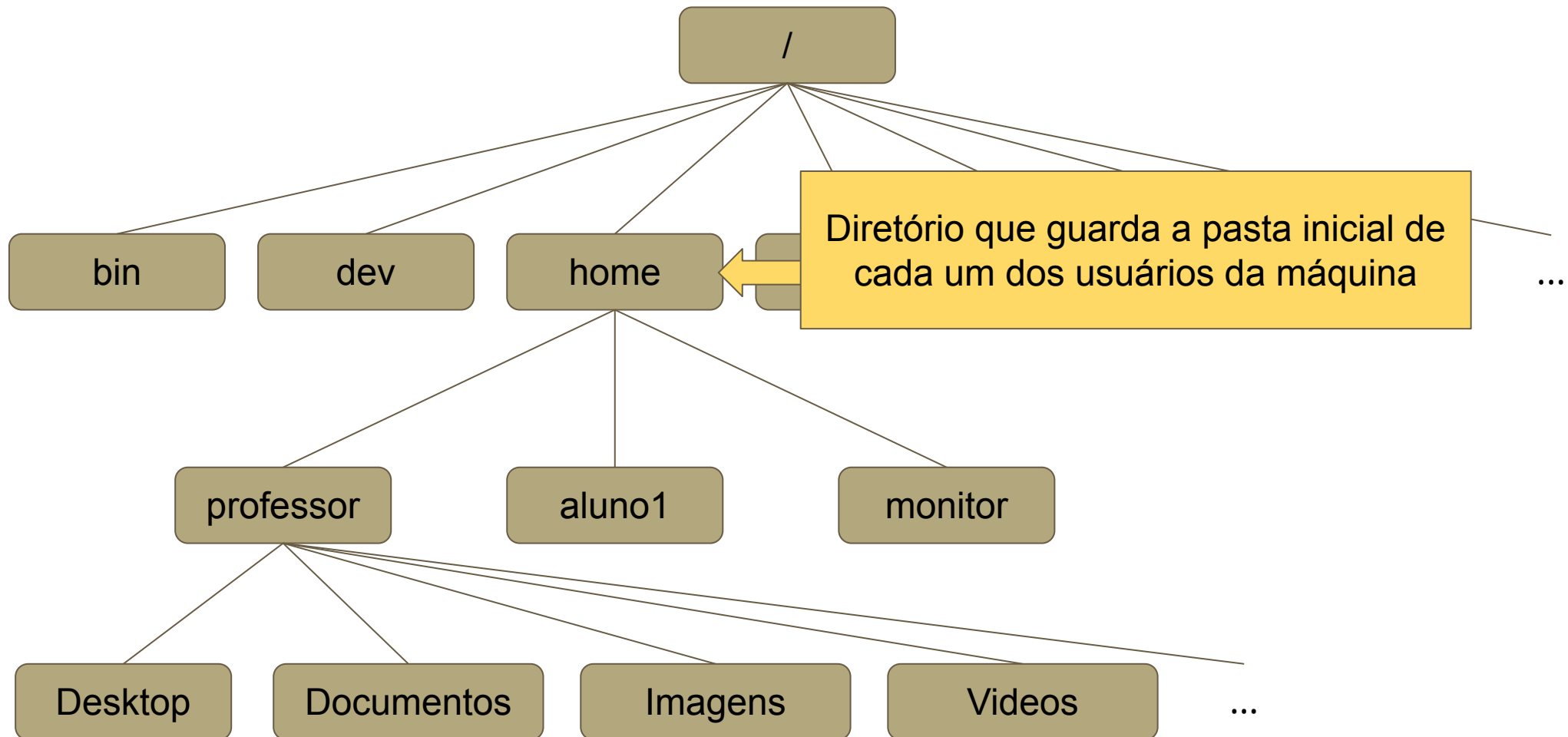
Estrutura de diretórios do Linux



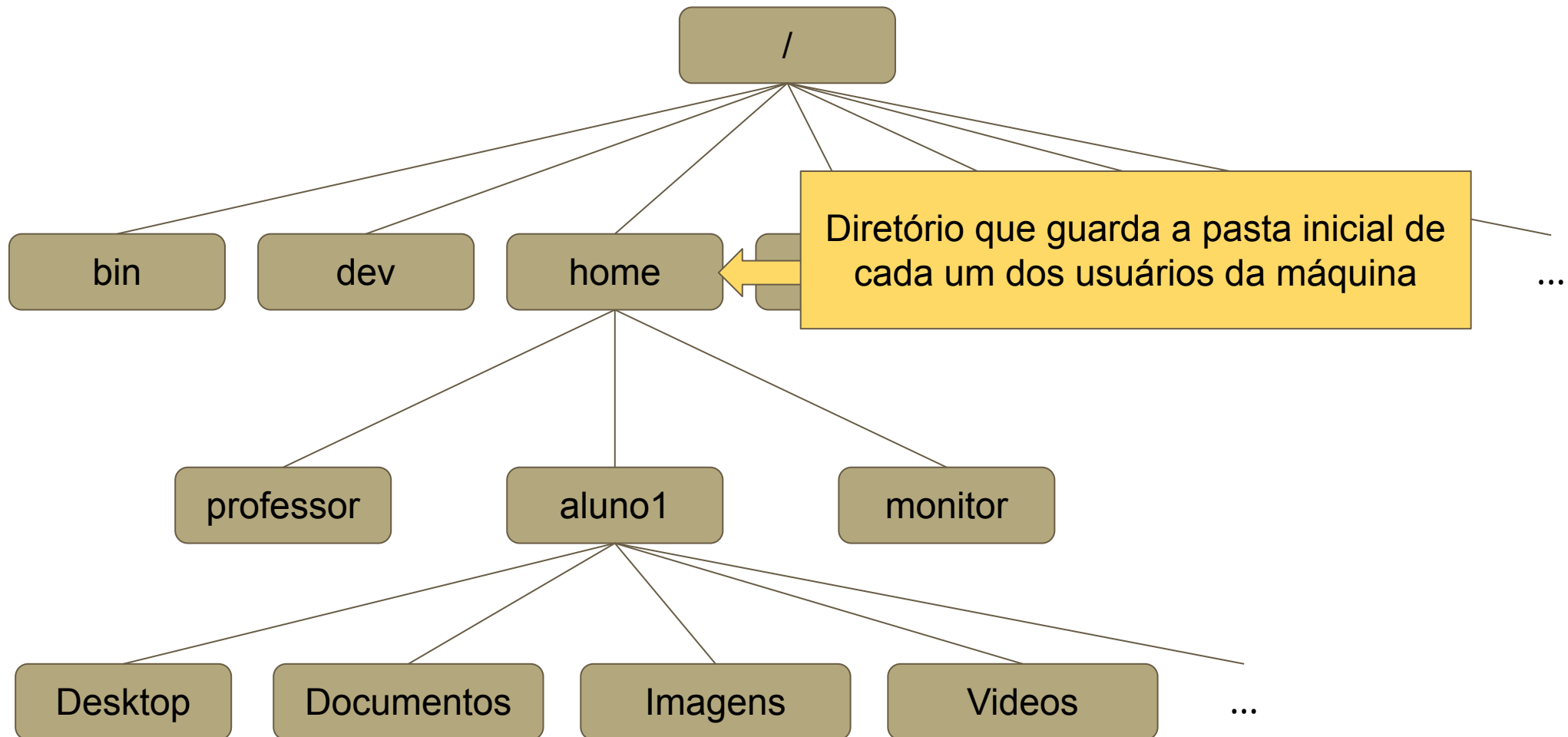
Estrutura de diretórios do Linux



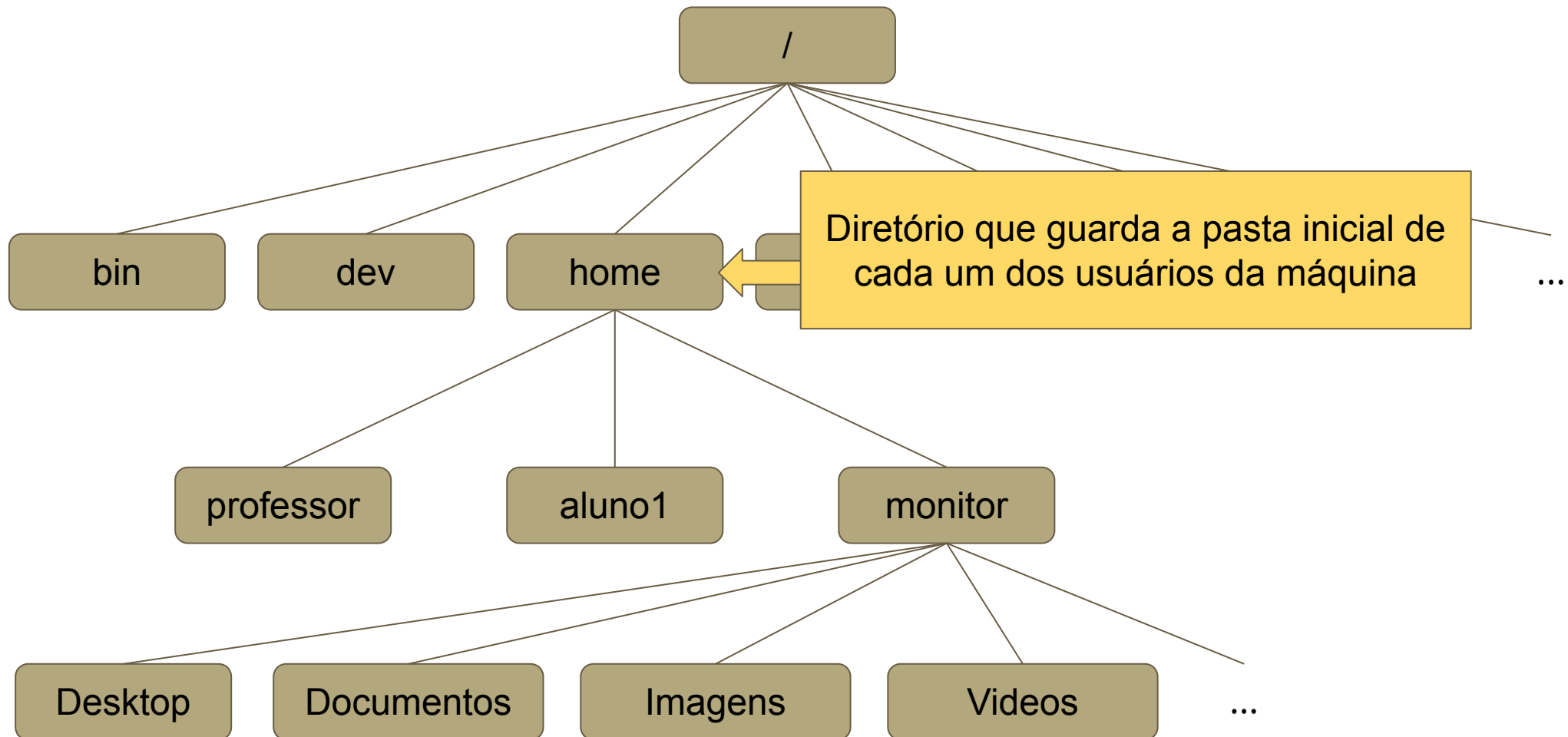
Estrutura de diretórios do Linux



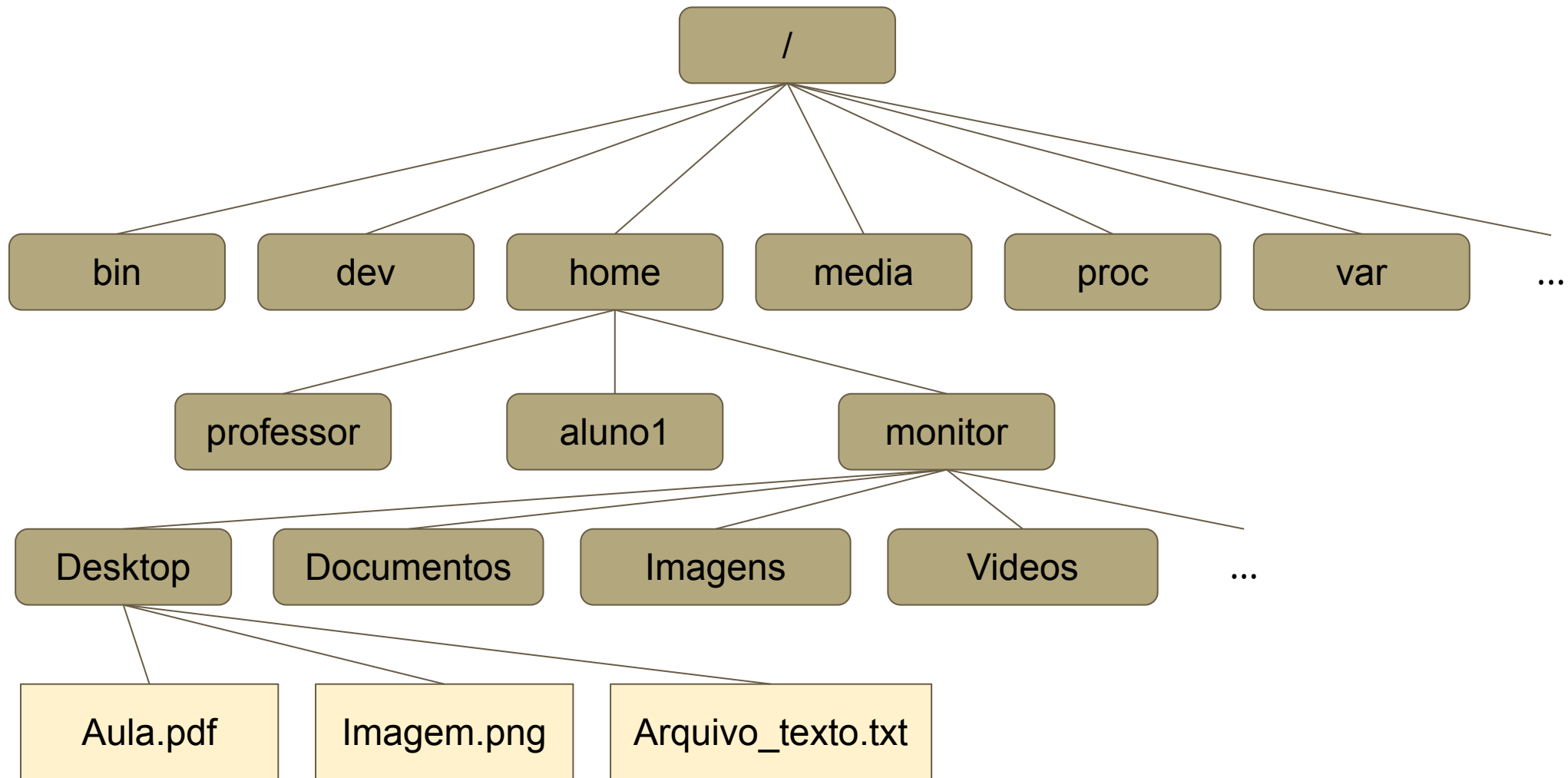
Estrutura de diretórios do Linux



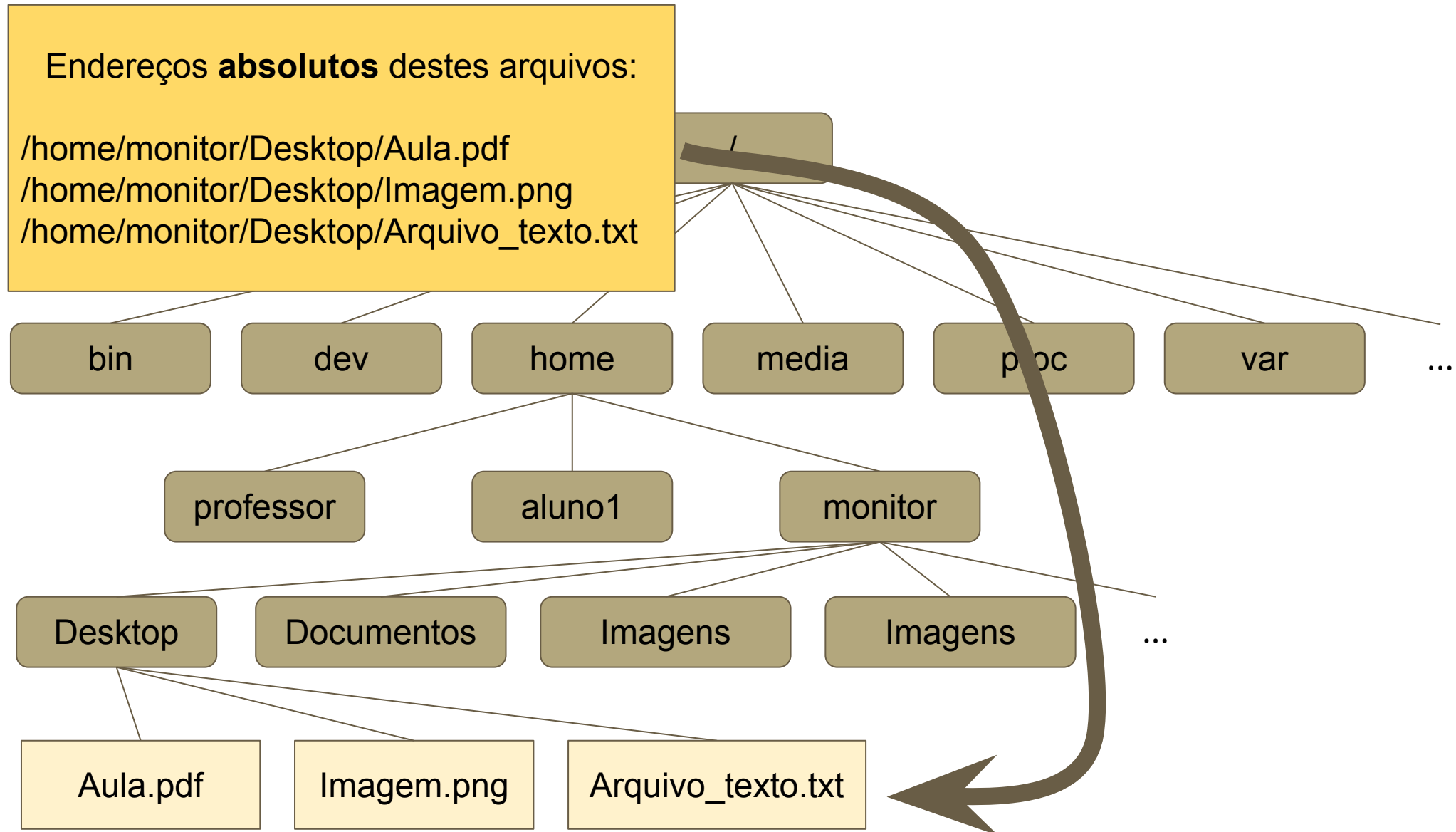
Estrutura de diretórios do Linux



Estrutura de diretórios do Linux



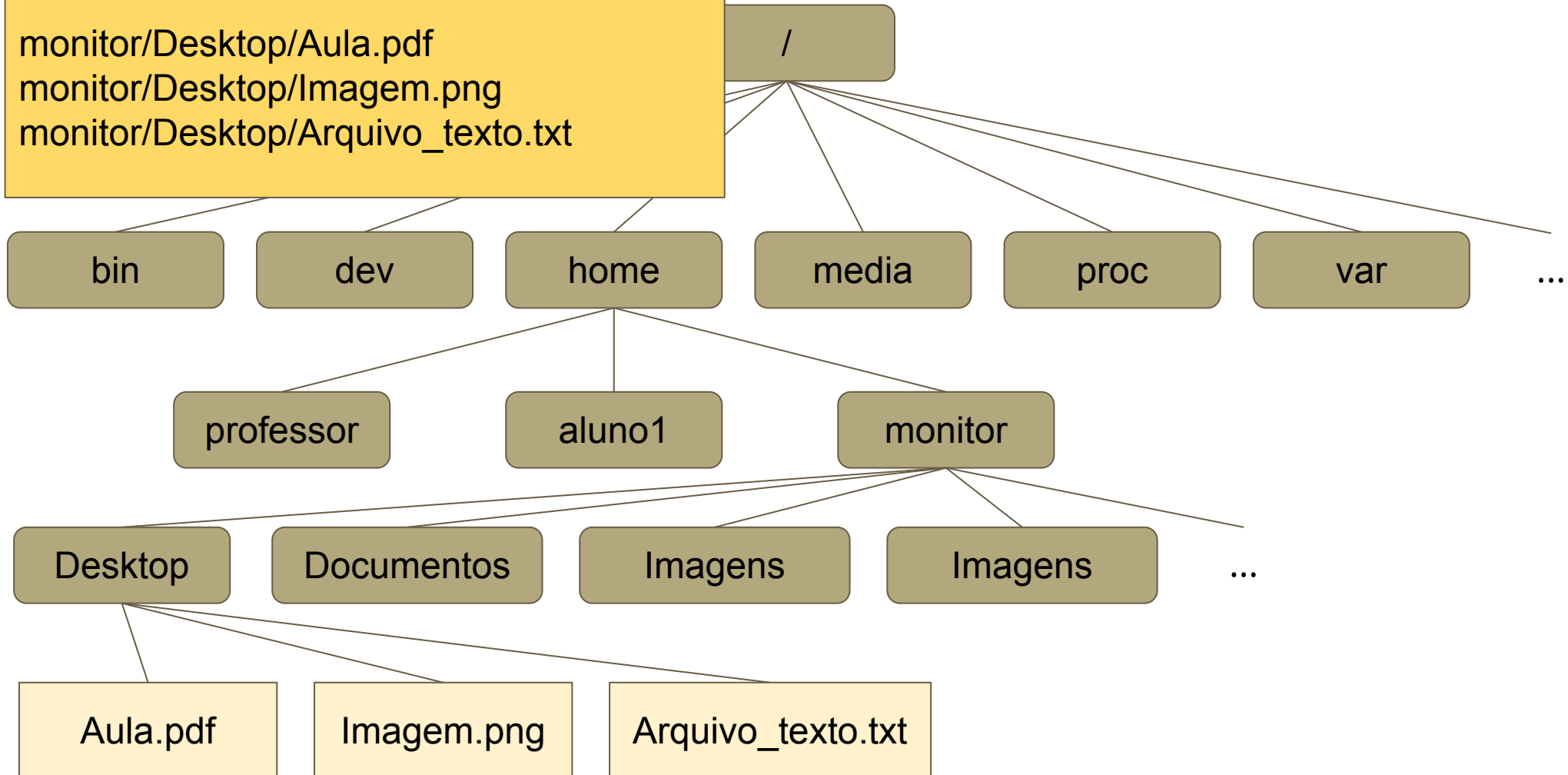
Estrutura de diretórios do Linux



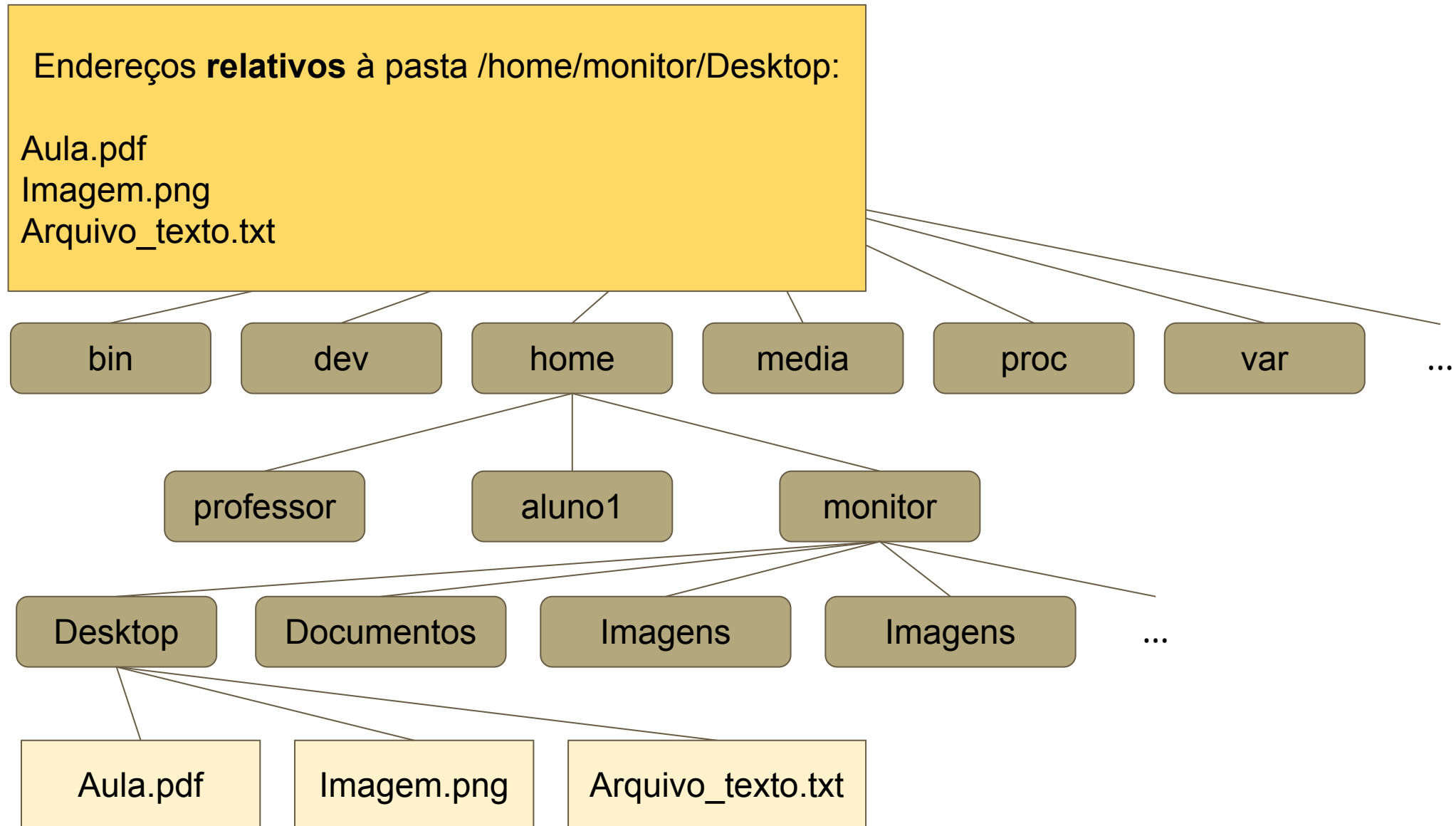
Estrutura de diretórios do Linux

Endereços **relativos** à pasta /home:

monitor/Desktop/Aula.pdf
monitor/Desktop/Imagem.png
monitor/Desktop/Arquivo_texto.txt



Estrutura de diretórios do Linux



Alguns conceitos

- Usuários: o Linux, por ser um sistema multi-usuário, apresenta um conjunto característico de usuários.
- root: o usuário raiz, super-usuário, administrador,
 - Logando como o usuário root você tem controle total sobre o sistema: ligar/desligar, adicionar/remover usuários, alterar senhas, instalar/remover aplicações,
- Outros usuários: definidos sobre demanda
 - usuários comuns;
 - daemons;
 - serviços.

Alguns conceitos

- Cada usuário possui um conjunto de permissões e privilégios
- Isso faz com que um sistema de grande porte seja mais seguro intrinsecamente: nativamente o sistema disponibiliza um conjunto de privilégios
- Uma conta para cada usuário
- Não é elegante operar como root ou deixar usuário sem senha

Iniciando o uso do Linux

- Interfaces gráficas maduras: KDE, GNOME, Wmaker.
- Aplicações em linha de comando:
 - Os comandos são os mesmos, independente do UNIX
 - São disponibilizadas poderosas ferramentas de filtragem de dados
- Alguns motivos para aprender a usar o shell (linha de comando):
 - Shell inteligente: histórico e auto-completar;
 - Sistema eficiente de automação de tarefas;

Iniciando o uso do Linux

- Simule um terminal Linux online:

https://www.tutorialspoint.com/unix_terminal_online.php

- Utilize o Ubuntu dentro do Windows (aplicativo Ubuntu)

<https://www.microsoft.com/pt-br/p/ubuntu/9nblggh4msv6>

- Execute o Ubuntu (ou o instale) a partir de um pendrive

<https://ubuntu.com/tutorials/tutorial-create-a-usb-stick-on-windows>

Comandos simples

- Considere que o usuário “professor” fez login em um terminal no Linux:

```
~/Desktop $
```

- O texto “~/Desktop” indica que o terminal está sendo executado a partir da pasta “/home/professor/Desktop”.
 - O terminal sempre é executado a partir de alguma pasta específica.
 - “~” é uma forma abreviada de representar a pasta inicial do usuário. Se o usuário “aluno” fizer login, “~” representa a pasta “/home/aluno”.

Comandos simples

- O comando “pwd” indica a partir de qual pasta o terminal está sendo executado.

```
~/Desktop $ pwd  
/home/professor/Desktop
```

- O comando “ls” lista o conteúdo da pasta a partir de onde o terminal está sendo executado.

```
~/Desktop $ ls  
Aulas      Imagem.png  Notas.txt   Prova1      Prova2      Prova3
```

Comandos simples

- Se quisermos entrar na pasta “Prova1”, podemos usar o comando “cd”

```
~/Desktop $ cd Prova1  
~/Desktop/Prova1 $
```

- Neste exemplo, usamos o endereço **relativo**, já que a pasta Prova1 se encontra dentro da pasta “Desktop”. Duas formas de obter o mesmo resultado usando o endereço **absoluto** são:

```
~/Desktop $ cd /home/professor/Desktop/Prova1  
~/Desktop/Prova1 $
```

```
~/Desktop $ cd ~/Desktop/Prova1  
~/Desktop/Prova1 $
```

Comandos simples

- O comando “ls -l” lista o conteúdo da pasta com mais riqueza de detalhes:

```
~/Desktop/Prova1 $ ls -l
-rwxr-xr-x 1 professor professor 150 mar 17 15:31 Media.sh
-rw-r--r-- 1 professor professor  51 mar 17 15:15 Notas.txt
-rw-r--r-- 1 professor professor 167 mar 17 15:08 Prova1.txt
```

- Neste caso, continuamos chamando o comando “ls”, e indicamos que queremos a lista completa (argumento “-l”).
- Repare que são indicados 3 arquivos dentro da pasta “Prova1”.
- Além disso, a linha de cada arquivo começa com uma sequência de caracteres “-rwx”, referentes às **permissões de cada arquivo**.

Manipulando arquivos

O sistema de permissões Linux

- As permissões são associadas a arquivos, diretórios e programas, indicando o que pode ser feito com cada um destes (leitura, escrita e execução) por diferentes usuários da máquina
- String de permissão: **rwXrwxrwx**
 - As **três primeiras letras** definem as permissões do proprietário
 - As **3 letras do meio**, do grupo proprietário
 - As **últimos três letras**, de todos os outros usuários
- Somente o proprietário (e o usuário “root”) podem mudar as permissões de um arquivo
- Mantém a privacidade dos usuários e a integridade do sistema

Manipulando arquivos

O sistema de permissões Linux

- Podemos ver que o arquivo “Media.sh” pode ser:
 - Lido, escrito e executado pelo usuário “professor” (**rw****x**)
 - Lido e executado pelos outros usuários da máquina (**r**-**x**)

```
~/Desktop/Prova1 $ ls -l
-rwxr-xr-x 1 professor professor 150 mar 17 15:31 Media.sh
-rw-r--r-- 1 professor professor  51 mar 17 15:15 Notas.txt
-rw-r--r-- 1 professor professor 167 mar 17 15:08 Prova1.txt
```

- Os arquivos “Notas.txt” e “Prova1.txt” podem ser lidos e escritos pelo usuário “professor”, mas não executados (**rw**-), e os outros usuários só podem ler (**r**--).

Manipulando arquivos

O sistema de permissões Linux

- Para fazermos com que somente o usuário “professor” consiga ler o arquivo “Notas.txt”, podemos usar o comando “chmod”:

```
~/Desktop/Prova1 $ chmod 600 Notas.txt
~/Desktop/Prova1 $ ls -l
-rwxr-xr-x 1 professor professor 150 mar 17 15:31 Media.sh
-rw----- 1 professor professor  51 mar 17 15:15 Notas.txt
-rw-r--r-- 1 professor professor 167 mar 17 15:08 Prova1.txt
```

- Para compreender o uso do número “600”, deve-se converter cada algarismo para a representação binária:

6	0	0
110	000	000
rw-	---	---

Ou seja, o usuário “professor” pode ler e escrever neste arquivo (**rw-**), e os outros usuários não podem fazer nada (---).

Manipulando arquivos

O sistema de permissões Linux

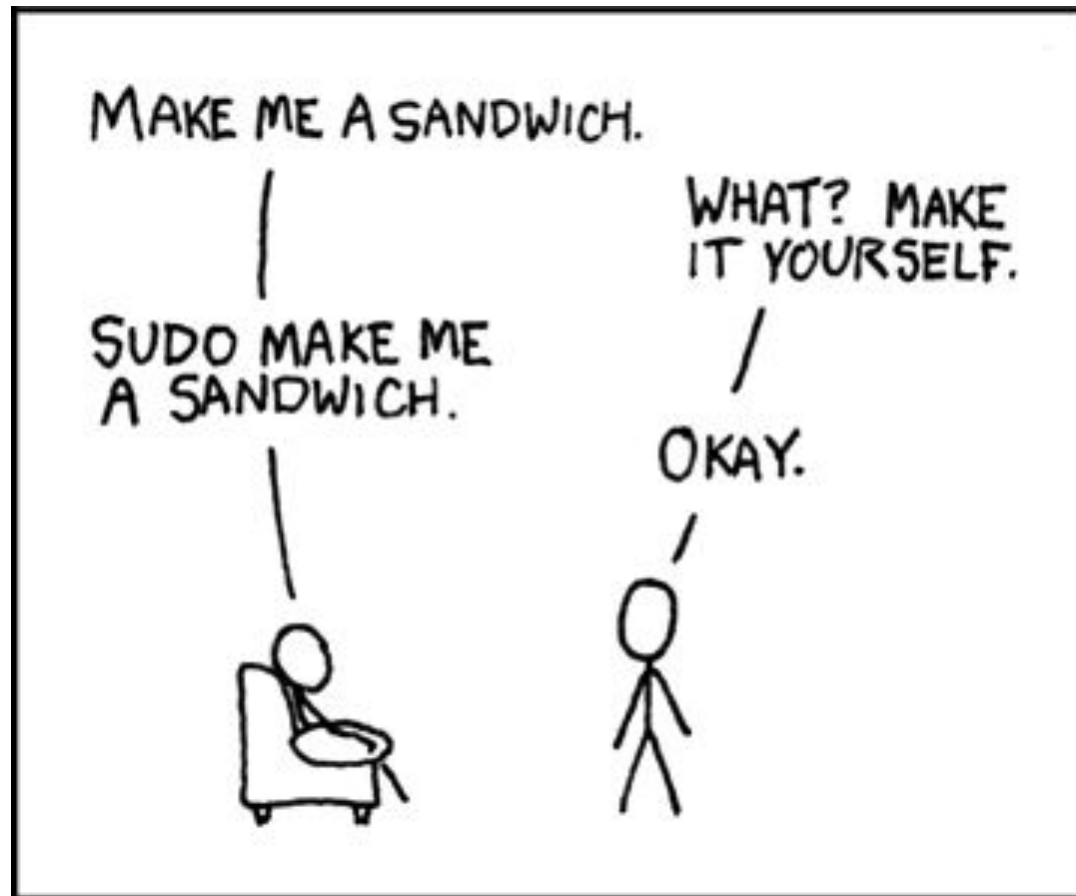
- O comando “poweroff” desliga o computador, e o comando “reboot” reinicia o computador. O comando “shutdown” permite agendar o desligamento e o reinício do computador, dentre outros.
- Eles só podem ser executados pelo “root”.
- O comando “su” permite logar como “root”.
- O comando “sudo” antes de outro comando permite executa-lo como “root”:

```
~/Desktop/Prova1 $ poweroff  
bash: poweroff: Permission denied  
~/Desktop/Prova1 $ sudo poweroff
```

(O computador é desligado em seguida.)

Manipulando arquivos

O sistema de permissões Linux



<https://xkcd.com/149/>

Comandos simples

- Para lermos o conteúdo de um arquivo, usamos o comando “cat”:

```
~/Desktop/Prova1 $ cat Notas.txt
Fulano de Tal - 8
Beltrano de Tal - 9
Sicrano - 10
```

```
~/Desktop/Prova1 $ cat Media.sh
#!/bin/bash
nf=0
N=0
for ns in $(grep -oP '[$0-9]+' $1)
do
    N=$((N+1))
    nf=$((nf+ns))
done
nf=$((nf+N/2))
nf=$((nf/N))
echo "Media($N alunos) = $nf"
```

Comandos simples

- O arquivo “Media.sh” é um *script* que executa uma série de comandos de terminal. Mais especificamente, ele procura números no final de cada linha de um arquivo, e depois calcula a média destes:

```
~/Desktop/Prova1 $ ./Media.sh Notas.txt  
Media(3 alunos) = 9
```

- Repare que o *script* foi executado acrescentando “./” ao nome do mesmo. Sem o “./”, o terminal procura por um programa chamado “Media.sh”, que provavelmente não existe na sua máquina:

```
~/Desktop/Prova1 $ Media.sh Notas.txt  
Media.sh: command not found
```

(Veremos *scripts* com mais atenção em uma próxima aula.)

Comandos simples

- Para voltarmos para a pasta anterior, usamos o caminho “../”:

```
~/Desktop/Prova1 $ cd ../  
~/Desktop $
```

(Ou seja, “./” se refere ao diretório atual, e “../”, ao diretório-pai.)

(O comando “cd ..” funciona da mesma forma que “cd ../”)

- Para copiar um arquivo, utilize o comando “cp”:

```
~/Desktop $ ls  
Aulas  Imagem.png  Notas.txt  Prova1  Prova2  Prova3  
~/Desktop $ cp Imagem.png Imagem2.png  
~/Desktop $ ls  
Aulas          Imagem.png  Prova1  Prova3  
Imagem2.png    Notas.txt   Prova2
```

Comandos simples

- Para mover um arquivo, utilize o comando “mv”:

```
~/Desktop $ mv Imagem2.png Prova1/Imagem2.png
~/Desktop $ cd Prova1
~/Desktop/Prova1 $ ls
Imagem2.png  Media.sh  Notas.txt  Prova1.txt
```

- Para renomear um arquivo, use o próprio comando “mv”:

```
~/Desktop/Prova1 $ mv Imagem2.png Imagem.png
~/Desktop/Prova1 $ ls
Imagem.png  Media.sh  Notas.txt  Prova1.txt
```

(Como o “mv” apaga a “fonte” da cópia, sobra apenas o arquivo novo.)

- Para remover um arquivo, use o comando “rm”:

```
~/Desktop/Prova1 $ rm Imagem.png
~/Desktop/Prova1 $ ls
Media.sh  Notas.txt  Prova1.txt
```

Comandos simples

- Para mover um arquivo, utilize o comando “mv”:

```
~/Desktop $ mv Imagem2.png Prova1/Imagem2.png
~/Desktop $ cd Prova1
~/Desktop/Prova1 $ ls
Imagem2.png  Media.sh  Notas.txt  Prova1.txt
```

- Para renomear um arquivo, use o próprio comando “mv”:

```
~/Desktop/Prova1 $ mv Imagem2.png Imagem.png
~/Desktop/Prova1 $ ls
Imagem.png  Media.sh  Notas.txt  Prova1.txt
```

(Como o “mv” apaga a “fonte” da cópia, sobra apenas o arquivo novo.)

- Para remover um arquivo, use o comando “rm”:

```
~/Desktop/Prova1 $ rm Imagem.png
~/Desktop/Prova1 $ ls
Media.sh  Notas.txt  Prova1.txt
```

O comando “rm” não manda o arquivo para a lixeira! Ele o apaga completamente!!!

Comandos simples

- O comando “date” apresenta o horário e a data atual:

```
~/Desktop/Prova1 $ date  
ter mar 17 14:49:31 -03 2020
```

- O comando “cal” apresenta um calendário:

```
~/Desktop/Prova1 $ cal  
      Março 2020  
do se te qu qu se sá  
  1   2   3   4   5   6   7  
  8   9  10  11  12  13  14  
15  16  17  18  19  20  21  
22  23  24  25  26  27  28  
29  30  31
```

- O comando “echo” escreve uma mensagem na tela:

```
~/Desktop/Prova1 $ echo Olá Mundo!  
Olá Mundo!
```

Recursos do shell

- O caractere “>” redireciona a saída de um comando para um arquivo:

```
~/Desktop/Prova1 $ date > data.txt
~/Desktop/Prova1 $ cat data.txt
ter mar 17 15:04:31 -03 2020
```

- Se o arquivo já existe, ele é sobrescrito:

```
~/Desktop/Prova1 $ echo Olá! > data.txt
~/Desktop/Prova1 $ cat data.txt
Olá!
```

- O caractere “>>” também redireciona a saída para um arquivo, mas não o sobrescreve:

```
~/Desktop/Prova1 $ echo Tudo bem? >> data.txt
~/Desktop/Prova1 $ echo Como vão as coisas? >> data.txt
~/Desktop/Prova1 $ cat data.txt
Olá!
Tudo bem?
Como vão as coisas?
```

Recursos do shell

- O caractere “|” (pipe) redireciona a saída de um comando para outro:

```
~/Desktop/Prova1 $ cal 3 2020
    Março 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
~/Desktop/Prova1 $ echo 3 2020 > mes_ano.txt
~/Desktop/Prova1 $ cat mes_ano.txt | cal
    Março 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

Recursos do shell

- O caractere “|” (pipe) redireciona a saída de um comando para outro:

```
~/Desktop/Prova1 $ cal 3 2020
```

```
Março 2020
```

```
do se te qu qu se sá
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

```
~/Desktop/Prova1 $ echo 3 2020 > mes_ano.txt
```

```
~/Desktop/Prova1 $ cat mes_ano.txt | cal
```

```
Março 2020
```

```
do se te qu qu se sá
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

Usuário pediu o calendário
de março de 2020

Recursos do shell

- O caractere “|” (pipe) redireciona a saída de um comando para outro:

```
~/Desktop/Prova1 $ cal 3 2020
```

```
    Março 2020
```

```
do se te qu qu se sá
```

```
 1  2  3  4  5  6  7
```

```
 8  9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

```
~/Desktop/Prova1 $ echo 3 2020 > mes_ano.txt
```

```
~/Desktop/Prova1 $ cat mes_ano.txt | cal
```

```
    Março 2020
```

```
do se te qu qu se sá
```

```
 1  2  3  4  5  6  7
```

```
 8  9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

Usuário escreveu “3 2020”
no arquivo “mes_ano.txt”



Recursos do shell

- O caractere “|” (pipe) redireciona a saída de um comando para outro:

```
~/Desktop/Prova1 $ cal 3 2020
```

```
Março 2020
```

```
do se te qu qu se sá
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

```
~/Desktop/Prova1 $ echo 3 2020 > mes_ano.txt
```

```
~/Desktop/Prova1 $ cat mes_ano.txt | cal
```

```
Março 2020
```

```
do se te qu qu se sá
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

Usuário mandou ler o arquivo “mes_ano.txt”, e ao invés de escrever o resultado na tela, mandou o resultado ser passado para o programa “cal”

Recursos do shell

- O caractere “|” (pipe) redireciona a saída de um comando para outro:

```
~/Desktop/Prova1 $ cal 3 2020
```

```
Março 2020
```

```
do se te qu qu se sá
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

```
~/Desktop/Prova1 $ cat mes_ano.txt
```

```
~/Desktop/Prova1 $ | cal
```

```
Março 2020
```

```
do se te qu qu se sá
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

Mesmo resultado

Recursos do shell

- O caractere “;” permite executar comandos em sequência:

```
~/Desktop/Prova1 $ echo "Dia e hora:"; date; echo "FIM"  
Dia e hora:  
ter mar 17 15:07:29 -03 2020  
FIM
```

- Se você se esqueceu do nome de algum comando ou arquivo, pode iniciar o nome do comando/arquivo e pressionar TAB:

```
~/Desktop/Prova1 $ cat No
```

(PRESSIONE TAB)

```
~/Desktop/Prova1 $ cat Notas.txt  
Fulano de Tal - 8  
Beltrano de Tal - 9  
Sicrano - 10
```


Recursos do shell

- Pressionando as setas para cima e para baixo, seu histórico de comandos executados vai sendo mostrado.
- Pressionando CONTROL+R, você pode fazer uma busca por comandos já executados.
- Executando o comando “history”, você vê seu histórico de comandos já executados.
- Executando o comando “clear” ou pressionando CONTROL+L, você limpa a tela do terminal.
- Para ver um manual de um comando, executando “man CMD”, onde “CMD” é o comando de seu interesse.

Recursos do shell

- Executando o comando “ps”, você vê uma lista de programas em execução. Execute “ps -aux” para maior riqueza de detalhes.
- Executando o comando “top”, você vê uma lista dinâmica de programas em execução.
- O comando “kill” pode parar a execução de outros programas.
- O comando “mkdir” cria um diretório.
- O comando “rmdir” apaga um diretório.

Recursos do shell

- Metacaracteres:

*	qualquer string
?	qualquer caracter isolado
[...]	qualquer caracter isolado dentro dos colchetes
[a-z]*	qualquer caracter alfabético minúsculo
*	o caractere '*'
;	separador seqüencial
&	comando em segundo plano
#	comentário

Recursos do shell

- Metacaracteres:


```
~/Desktop/Prova1 $ ls
data.txt  Media.sh  mes_ano.txt  Notas.txt  Prova1.txt
~/Desktop/Prova1 $ ls *.txt
data.txt  mes_ano.txt  Notas.txt  Prova1.txt
~/Desktop/Prova1 $ ls ?.txt
ls: cannot access '?.txt': No such file or directory
~/Desktop/Prova1 $ ls [Mm]e*
Media.sh  mes_ano.txt
```

Recursos do shell

- Metacaracteres:

Liste todos os arquivos
que começam com
qualquer texto ("*"), e
terminam em ".txt"

```
~/Desktop/Prova1 $ ls
data.txt  Media.sh  mes_ano.txt  Notas.txt  Prova1.txt
~/Desktop/Prova1 $ ls *.txt
data.txt  mes_ano.txt  Notas.txt  Prova1.txt
~/Desktop/Prova1 $ ls ?.txt
ls: cannot access '?.txt': No such file or directory
~/Desktop/Prova1 $ ls [Mm]e*
Media.sh  mes_ano.txt
```



Recursos do shell

- Metacaracteres:

Liste todos os arquivos que começam com somente um caracter ("?"), e que terminam em ".txt"

```
~/Desktop/Prova1 $ ls
data.txt  Media.sh  Prova1.txt
~/Desktop/Prova1 $ ls *.txt
data.txt  mes_ano.txt  Notas.txt  Prova1.txt
~/Desktop/Prova1 $ ls ?.txt
ls: cannot access '?.txt': No such file or directory
~/Desktop/Prova1 $ ls [Mm]e*
Media.sh  mes_ano.txt
```

Recursos do shell

- Metacaracteres:

Liste todos os arquivos
que começam com as
letras “M” ou “m”,
seguido da letra “e”, e
que terminam de
qualquer jeito

```
~/Desktop/Prova1 $ ls  
data.txt  Media.sh  Prova1.txt  
~/Desktop/Prova1 $ ls  
data.txt  mes_ano.txt  
~/Desktop/Prova1 $ ls  
ls: cannot access '?..txt': No such file or directory  
~/Desktop/Prova1 $ ls [Mm]e*  
Media.sh  mes_ano.txt
```

Comandos para verificar arquivos

- Às vezes, não precisamos verificar todo o conteúdo de um arquivo:

```
$ head -atributos arquivo
```

```
Atributo comum: -f (continuamente)
```

```
$ tail -atributos arquivo
```

```
Atributo comum: -f (continuamente)
```


Operações sobre arquivos

- O comando “sort” ordena as linhas de um arquivo:

```
~/Desktop/Prova1 $ cat Notas.txt
Fulano de Tal - 8
Beltrano de Tal - 9
Sicrano - 10
~/Desktop/Prova1 $ sort Notas.txt
Beltrano de Tal - 9
Fulano de Tal - 8
Sicrano - 10
```

- O comando “grep” busca padrões em um arquivo:

```
~/Desktop/Prova1 $ grep Fulano Notas.txt
Fulano de Tal - 8
~/Desktop/Prova1 $ grep Tal Notas.txt
Fulano de Tal - 8
Beltrano de Tal - 9
```

Operações sobre arquivos

- O comando “grep” aceita metacaracteres:
 - . para representar um caractere
 - [] para representar um intervalo
 - “^padrao” para procurar por “padrao” no início da linha
 - “padrao\$” para procurar por “padrao” no final da linha

```
~/Desktop/Prova1 $ grep “.ano” Notas.txt
Fulano de Tal - 8
Beltrano de Tal - 9
Sicrano - 10
~/Desktop/Prova1 $ grep “[8-9]” Notas.txt
Fulano de Tal - 8
Beltrano de Tal - 9
~/Desktop/Prova1 $ grep Tal Notas.txt
Fulano de Tal - 8
Beltrano de Tal - 9
~/Desktop/Prova1 $ grep “^Tal” Notas.txt
~/Desktop/Prova1 $ grep “9$” Notas.txt
Beltrano de Tal - 9
```

Operações sobre arquivos

- O comando “sed” substitui ocorrências de um padrão em um arquivo:

```
~/Desktop/Prova1 $ sed -e 's/Tal/Talz/' Notas.txt
Fulano de Talz - 8
Beltrano de Talz - 9
Sicrano - 10
~/Desktop/Prova1$ sed -e '/Sicrano/d' Notas.txt
Fulano de Tal - 8
Beltrano de Tal - 9
```

- Os comandos “sort”, “grep” e “sed” são poderosíssimos se usados corretamente, evitando muito trabalho manual.

Instalando programas

apt-get

- Distros mais tradicionais disponibilizam pacotes pré-compilados e sistemas de gerenciamento de pacotes
 - O Debian usa o formato *.deb para disponibilizar pacotes pré-compilados e o sistema apt para gerenciamento dos mesmos;
 - O Fedora usa o formato rpm e o sistema yum.
- Os sistemas de gerenciamento de pacotes permitem a seleção e instalação automática dos mesmos.
- Se o pacote estiver no computador, o sistema irá buscá-lo e instalá-lo.
- Se o pacote não estiver no computador, o sistema procurará na Internet as versões mais atuais do pacote e, encontrando-os, irá iniciar o processo de download e instalação.

Instalando programas

apt-get

- Usando o apt-get:

\$ apt-get update

Atualiza base de dados de pacotes

\$ apt-cache search pacote

Busca pacote na base de dados

\$ apt-get install pacote

Baixa o pacote e o instala

\$ apt-get remove pacote

Desinstala o pacote

\$ apt-get upgrade

Atualiza o sistema, baixando e instalando novas versões

Instalando programas *.deb

- Existe a opção de baixar os pacotes *.deb e instalá-los manualmente:

```
$ dpkg -i pacote.deb
```

Instala o pacote

```
$ dpkg -r pacote
```

Desinstala o pacote: note que não há extensão nesse comando.

```
$ dpkg -l [pacote]
```

Lista todos os pacotes instalados ou verifica se `pacote` está instalado