

# [MAC0211] Laboratório de Programação I

## Aula 10

### Interpretadores de Comandos (*Shells*)

Alair Pereira do Lago

DCC-IME-USP

26 de março de 2015

# Programas de sistema

- ▶ **Sistema operacional** – são os responsáveis por prover as funcionalidades disponibilizadas via chamadas aos sistema
- ▶ **Compiladores, montadores, ligadores e interpretadores de comandos** – programas muito importantes e úteis para o funcionamento dos computadores, mas não fazem parte do SO

# Interpretador de comandos

## Definição

É um programa que implementa uma *interface de linha de comando*. Essa, por sua vez, permite que usuários submetam comandos a um programa de computador por meio de linhas sucessivas de texto (ou seja, as linhas de comando)

Shell = interpretador de comandos para um sistema operacional

- ▶ É interface primária existente entre um usuário e um sistema operacional quando o usuário não está usando uma interface gráfica
- ▶ por meio de um Shell, um usuário pode fazer um uso “intenso” das funcionalidades providas por um SO

# Um “parênteses” sobre o termo *shell*

## Em Computação ...

... o termo *shell* geralmente é usado para designar qualquer programa que atue como uma casca (= camada externa) entre usuários e: o núcleo de um SO, ou outros programas, ou até mesmo linguagens. Sob essa perspectiva, mesmo um programa gráfico pode ser chamado de *shell*. Exemplo de *shell* gráfico: o *Explorer.exe* (do Windows).

## Em SOs “originados” do Unix ...

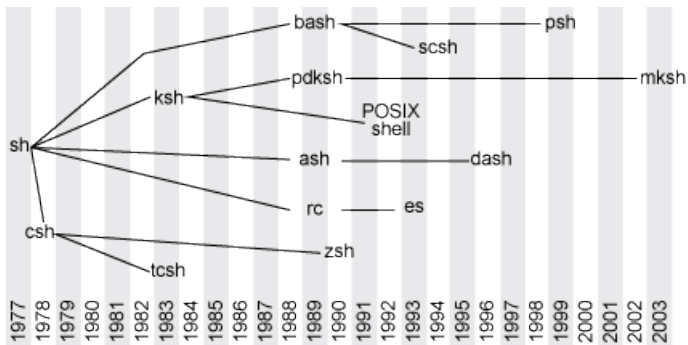
... *shell* assume um significado mais específico – o de interpretador de comandos em modo texto para o SO. Esse é o significado referenciado neste curso.

# Shells

Existem diversas implementações de Shells:

- ▶ no Windows: `command.com` e `cmd.exe` (*Command*), PowerShell, ...
- ▶ no Mac OS X: `Terminal.app` (*Terminal*), ...
- ▶ no Unix (e seus derivados): `sh` (*Bourne shell*), `csh` (*C shell*), `bash` (***Bourne-Again shell***), `ksh` (*Korn shell*), `ash` (*Almquist shell*) ...

# Árvore genealógica dos *Shells* do Linux



Fonte:

<http://www.ibm.com/developerworks/linux/library/l-linux-shells/>

# O bash

- ▶ É o *shell* padrão do GNU/Linux
- ▶ É um sucessor evoluído do `sh` (e bastante compatível com ele)
- ▶ Incorpora as melhores características do `cs`h e do `ks`h
- ▶ Sua implementação respeita as normas POSIX (para a portabilidade entre sistemas operacionais)
- ▶ Assim como os demais *shells* no UNIX, o `bash` combina duas funcionalidades:
  - ▶ interpretador de comandos
  - ▶ linguagem de programação
- ▶ Funciona em dois modos: interativo (executa comandos digitados no prompt) e não-interativo (executa comandos lidos de um arquivo de entrada)

# A linguagem de programação do bash

- ▶ O interpretador de comandos permite que usuários executem programas utilitários
- ▶ A linguagem de programação permite que os utilitários sejam combinados
- ▶ *Scripts* (= arquivos contendo comandos) podem ser criados, e esses arquivos têm status de novos comandos
- ▶ *Scripts* possibilitam que usuários personalizem seus ambientes, criando novos comandos para automatizar suas tarefas rotineiras de manutenção e otimização do funcionamento do computador.



# Funcionamento básico de um *shell*

Um *shell* gerencia a interação entre um usuário e o SO executando (geralmente) o seguinte conjunto de passos:

1. aguarda a entrada de uma ou mais linhas de comando (que podem ser digitados pelo usuário no terminal ou então lidos de um arquivo, chamado de *script*)
2. interpreta uma linha por vez, traduzindo-a em chamadas a *commands* existentes no *shell* ou no SO
3. cria processos-filhos para executar os comandos requisitados
4. espera que a execução dos processos criados termine
5. disponibiliza o resultado da execução (para o usuário ou para um outro programa)
6. volta ao passo inicial

# Utilitários para a manipulação de diretórios

- ▶ `ls <opções> <diretório>`  
Lista as entradas em um diretório.
- ▶ `cd <diretório>`  
Muda de diretório.
- ▶ `mkdir <opções> <nome>`  
Cria um novo diretório.
- ▶ `rmdir <opções> <diretório>`  
Apaga um diretório se ele estiver vazio.

# Utilitários para manipulação de arquivos

- ▶ `cp <opções> <origem> <destino>`  
Copia arquivos ou diretórios.
- ▶ `mv <opções> <origem> <destino>`  
Move arquivos ou diretórios.
- ▶ `rm <opções> <arquivos>`  
Apaga arquivos ou diretórios.
- ▶ `cat <arquivo(s)>`  
Concatena arquivos e imprime na saída padrão.
- ▶ `find -name <arquivo>`  
Localiza arquivos ou diretórios.
- ▶ `sort <arquivo>`  
Ordena alfabeticamente as linhas do arquivo.

# Outros utilitários muito usados

- ▶ **more <opções> <arquivo>**  
Exibe o conteúdo do arquivo, mostrando uma página por vez
- ▶ **less <opções> <arquivo>**  
Exibe o conteúdo do arquivo uma página por vez, possibilitando a navegação e outras funcionalidades (como busca de termos)
- ▶ **grep <opções> <padrão> <arquivo(s)>**  
Procura as linhas do(s) arquivo(s) que contêm o padrão indicado
- ▶ **ps <opções>**  
Exibe informações sobre os processos ativos
- ▶ **top <opções>**  
Provê uma visão dinâmica dos processos em execução
- ▶ **kill <opções> <ID do processo>**  
Envia um sinal ao processo; o sinal padrão é o sinal TERM (que solicita o término do processo)

# Operadores `bash` para redirecionamento



&gt;

Redireciona a saída de um comando para um arquivo em disco. Se o arquivo existir, será sobrescrito. Exemplo:

```
ls -la > dir.txt
```

⇒ redireciona a saída produzida pelo `ls` para o arquivo `dir.txt`.



»

Redireciona a saída, mas acrescentando os dados ao final do arquivo. Exemplo:

```
ps ux » dir.txt
```

⇒ redireciona a saída produzida pelo `ps` para o final do arquivo `dir.txt`, sem sobrescrever o arquivo.

# Operadores `bash` para redirecionamento



&lt;

Redireciona a entrada. Exemplo:

`meu_prog < testes.txt`  $\Rightarrow$  faz com que o arquivo `testes.txt` forneça a entrada para o programa `meu_prog`.



2&gt;

Redireciona a saída de erros. Exemplo:

`find -name Makefile 2> /dev/null`

$\Rightarrow$  faz com que os erros produzidos pelo comando `find` sejam redirecionados para o dispositivo de sistema `/dev/null` (ou seja, as saídas de erro serão descartadas).



&amp;&gt;

Redireciona a saída padrão e a saída de erros. Exemplo:

`ls -R /home/user/ &> /dev/null`

$\Rightarrow$  nenhuma saída aparecerá na tela, pois ambas foram redirecionadas para o “lixo” (`/dev/null`).

# Operadores bash – Pipe (|)

Permite que um programa utilize como entrada a saída de outro programa. Exemplos:

```
ls -a | sort
```

⇒ faz com que o comando `sort` receba como entrada a saída produzida pelo comando `ls`.

```
cat /home/alair/mac211.txt | wc -l > lista.txt
```

⇒ grava no arquivo `lista.txt` o número de linhas do arquivo `mac211` (que contém a lista de todos os matriculados em MAC211 em 2013).

```
cat /home/alair/mac211.txt | sort » lista.txt
```

⇒ coloca no final do arquivo `lista.txt` a lista em ordem alfabética dos alunos de MAC211 em 2013.

```
ls -l /etc | sort | less
```

⇒ mostra, em ordem alfabética, a listagem do diretório `/etc` e permite navegar dentro da listagem (`less`).

# Bibliografia e materiais recomendados

- ▶ Manual do bash  
<http://www.gnu.org/software/bash/manual/bashref.html>
- ▶ Guia para iniciantes do bash  
<http://www.tldp.org/LDP/Bash-Beginners-Guide/html/>
- ▶ *HowTo* da linha de comando do bash  
<http://tldp.org/HOWTO/Bash-Prompt-HOWTO/>
- ▶ Artigo “*Evolution of shells in Linux*”, de M. Tim Jones (da Intel)  
<http://www.ibm.com/developerworks/linux/library/l-linux-shells/>
- ▶ Notas das aulas de MAC0211 de 2010, feitas pelo Prof. Kon  
<http://www.ime.usp.br/~kon/MAC211>



# Cenas dos próximos capítulos...

- ▶ Mais sobre o bash em modo interativo
- ▶ Programação em bash - comandos compostos, estruturas de programação, declaração de variáveis, etc.