Programação I

Conceitos variados

André Grégio, Fabiano Silva, Luiz Albini e Marcos Castilho

DInf UFPR, Curitiba PR

21 de julho de 2020

Objetivos

- Customização do ambiente de trabalho
- Precedência para execução de comandos
- Substituição de comandos
- Um comando diferente (e chique!)
- Listas

Customização do ambiente de trabalho

- A *shell* de cada usuário vem configurada pelo administrador do sistema, mas pode ser alterada pelo usuário;
- Os arquivos importantes são:
  - *.bash_profile*, *.bash_logout* e *.bashrc*;
- Estes recursos estão muito bem explicados na literatura, recomendamos o livro *Learning the bash shell*, capítulo 3.
- Muito do que pode ser feito nesta customização depende do bom entendimento do uso de variáveis da *shell*, que explicaremos a seguir.

Precedência para execução de comandos

- Quando se executa um comando, a ordem de procura é a seguinte:
  1. aliases
  2. palavras chave (ex. function) ou comandos de controle de fluxo (if, for)
  3. funções
  4. *builtins*
  5. scripts e programas executáveis, segundo a variável *PATH*
- o *builtin* `type` mostra o que é associado a um certo nome

```
ci1001@fradim:~/tmp$ type -all echo
echo é um comando interno do interpretador
echo é /bin/echo
ci1001@fradim:~/tmp$
```

Substituição de comandos

- Este tipo de expansão é muito útil e seu uso bastante frequente
- Podemos atribuir como valor de uma variável a saída de um comando!
- `$(comando)`
- Exemplo:
  - `$(ls /home/bcc)` tem como valor todos os nomes de diretórios que estão em `/home/bcc`
  - `$(who | cut -d" " -f 1 | sort -u)` tem como valor os nomes de todos os usuários logados na máquina
  - Assim podemos por exemplo mandar um email para todos estes usuários: `mail $(ls /home/bcc)`

Um comando diferente (e chique!)

- Existe um comando *builtin* na *shell* que é diferente, meio estranho até, mas muito útil
- É o comando com a construção seguinte:
- `[ expr ]`
- Este comando retorna 0 (true) ou 1 (false) dependendo da avaliação da expressão condicional *expr*

Exemplo de uso

- `[ -a arquivo ]`: testa se *arquivo* existe
- `[ -d arquivo ]`: testa se *arquivo* existe e é diretório
- `[ -f arquivo ]`: testa se *arquivo* existe e é um arquivo regular
- `[ -h arquivo ]`: testa se *arquivo* existe e é um link simbólico

Uso do comando

- Ele é usado normalmente em combinação com outros comandos da *shell* que serão melhor explicados na próxima aula.
- Usado sozinho não faz muito sentido
- Veremos hoje um tipo de uso, mas é preciso conhecer um outro conceito antes

Listas

- Uma lista é uma sequência de um ou mais pipelines separados por um dos operadores seguintes:
- `;`, &, &&, `||`
- e opcionalmente terminado por um dentre os seguintes:
- `;`, &, `<newline>`

Listas

- && é um *AND*
  - `comando1 && comando2`
  - *comando2* só executa se *comando1* retornou status de saída zero
- || é um *OR*
  - `comando1 || comando2`
  - *comando2* só executa se *comando1* retornou status de saída diferente de zero

Exemplos

- `[ -d arquivo ] && echo OK || echo "NOT OK"`
- Se *arquivo* existe imprime na tela `OK`, senão imprime `NOT OK`
- Funciona como uma espécie de *if then else*

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.http://creativecommons.org/licenses/by-nc-nd/2.5/br/

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.http://creativecommons.org/licenses/by-nc-nd/2.5/br/