Universidade de São Paulo
Brasil

Aula 9
bash - parte 2

MAC0216 - Técnicas de Programação I

Professores: Alfredo, Daniel, Fabio e Kelly

Departamento de Ciência da Computação
Instituto de Matemática e Estatística

1.
Mais sobre bash

Comandos do bash acionados via [combinação de] teclas

- flechas verticais – percorre histórico de comandos
- flechas horizontais – posicionamento na linha corrente
- TAB – completa comando que começou a ser digitado (até onde dá sem ambiguidade)
- TAB TAB – mostra lista das opções de comandos possíveis para completar o que já começou a ser digitado (se houver várias)
- [CTRL-C] – cancela a execução do programa corrente
- [CTRL-D] – gera código de final de arquivo (fechando o terminal); também fim de sessão de login

Comandos do bash acionados via [combinação de] teclas

- [CTRL-Z] – suspende (“congela”) o programa correntemente em execução. O comando `jobs` lista os programas que estão suspensos e seus respectivos números. Para retomar a execução do programa suspenso, há duas possibilidades de comandos:
  - `fg %<num. do prog. suspenso>`

    ou simplesmente

    `%<num. do prog. suspenso>`

    Retoma a execução do programa suspenso em primeiro plano (modo síncrono)
  - `bg %<num. do prog. suspenso>`

    ou simplesmente

    `%<num. do prog. suspenso> &`

    Retoma a execução do programa suspenso em background (modo assíncrono)

A pilha de diretórios do bash

O bash permite manter em uma pilha os diretórios visitados mais recentemente. O seguintes comandos builtin manipulam essa pilha:

- `pushd <dir>`

  Empilha o diretório atual e depois faz um `cd` para `dir`

- `popd <opções>`

  Desempilha um diretório e depois faz um `cd` para ele

- `dirs <opções>`

  Mostra informações sobre o status atual da pilha de diretórios

Arquivos no UNIX – permissões de acesso

Todo objeto do sistema de arquivos do UNIX (arquivo ou diretório):

- pertence a um usuário e a um grupo
- possui 3 conjuntos de permissões, que definem os modos de acesso permitidos:
  - ao usuário dono do objeto (owner, u)
  - aos membros do grupo do objeto (group, g)
  - a todos os usuários do sistema (others, o)

Há três modos de acesso possíveis

- r – leitura (de read)
- w – escrita (de write)
- x – execução (de execution)

Arquivos no UNIX – permissões de acesso

O comando chmod altera as permissões de acesso de objetos do sistema de arquivos. O comando pode ser usado em dois modos: octal ou simbólico.

- `chmod <modo> <arquivo(s)>`

  onde modo é um número de três dígitos obtido por meio da tabela abaixo. Exemplo:

  `chmod 664 arq_compartilhado`

  (o comando atribui as permissões de leitura e escrita para o dono e o grupo, e somente leitura para outros usuários).

```
   dono grupo outros
   rwx  rwx   rwx
0- 000  000   000
1- 001  001   001
2- 010  010   010
3- 011  011   011
4- 100  100   100
5- 101  101   101
6- 110  110   110
7- 111  111   111
```

Arquivos no UNIX – permissões de acesso

O comando `chmod` altera as permissões de acesso de objetos do sistema de arquivos. O comando pode ser usado em dois modos: octal ou simbólico

- `chmod <referência> <operador> <modo> <arquivo(s)>`

  Referência: `u`, `g` ou `o` (e suas combinações)

  Operador: `=`, `+` (acrescenta) ou – (exclui)

  Modo: `r`, `w` ou `x` (e suas combinações)

  Exemplo:

  ```
  chmod ug=rw,o=r arq_compartilhado
  ```

Scripts

- Scripts são arquivos contendo comandos shell
- Para que esses arquivos tenham status de novos comandos, eles devem ser executáveis. Para transformar um script em executável:

```
chmod u+x meu_script
```

- Scripts podem ser iniciados por `#!` – o shebang (também chamado de hashbang, entre outros) : diretiva que indica o caminho do shell que deve ser carregado para executar (interpretar) o script
- O caracter # sozinho delimita o início de uma linha de comentário

Exemplo – programa "Hello World"

```
#!/bin/bash

echo Hello World!  # imprime "Hello World" na saída padrão
```

Variáveis de ambiente

- Variáveis em bash não possuem tipo
- Elas podem conter números, caracteres ou cadeias de caracteres
- Elas não precisam ser declaradas; para criar uma variável, basta atribuir um valor a ela
- Para recuperar o valor armazenado em uma variável, colocar um '`$`' em frente ao seu nome (`${}` para evitar ambiguidade)

Exemplo

```
STR='Hello World!'

echo $STR
```

Obs.: Note que na definição da variável, não pode haver espaços nem antes e nem depois do sinal de "="

Variáveis + expressões

Exemplo – script que faz backup do home

```
#!/bin/bash
ARQ_SAIDA=/var/meu-backup-$(date +%d%m%Y).tgz
tar -czf $ARQ_SAIDA /home/usuario/
```

- A expressão `$(comando)` executa comando e captura o resultado dele
- `$(date +%d%m%Y)` – expressão que executa o comando date, gerando uma string com a data atual no formato dia-mês-ano
- o nome do arquivo com o backup (compactado) dos dados do diretório home do usuário possui como sufixo a data em que o arquivo foi criado

Variáveis de ambiente definidas com export

- Uma variável de ambiente também pode ser definida com o comando `export`. Exemplo:

```
export PATH=/usr/bin
```

- Depois de definida com o `export`, uma variável fica "visível" nos scripts ou programas executados posteriormente

Exemplo (Script "digaoi")

```
#!/bin/bash
echo Oi, $USUARIO
```

Execução e saída

```
$ export USUARIO=Aluno
$ ./digaoi
Oi, Aluno
```

Variáveis de ambiente “famosas”

- `PATH` – caminhos para a busca de programas
- `PWD` – diretório corrente
- `SHELL` – shell padrão
- `HOME` – diretório home do usuário
- `USER` - nome do usuário

Material recomendado

- Bash by example (Partes 1 e 2)

  http://www.ibm.com/developerworks/linux/library/l-bash/index.html

  https://developer.ibm.com/articles/l-bash2/