Lista de Exercícios 08

Estruturas de Fluxo Condicionais

Exercício 01) Escreva um *shell script* que recebe um arquivo como argumento em linha de comando e verifica se existe ou não. Caso exista, verifique se é um arquivo, um diretório ou desconhecido mostrando quais as permissões que o **usuário atual** (**não o dono do arquivo**) possui sobre ele. Caso não exista, imprima que é inválido. **Cuidado**, essas informações não são obtidas com comandos como `ls`, `stat` ou `file`.

```
$ ./checkfile
Usar: ./checkfile [Arquivo]
$ ./checkfile /etc/passwd
/etc/passwd: arquivo (r--)
$ ./checkfile /tmp
/tmp: diretório (rwx)
$ ./checkfile /dev/random
/dev/random: desconhecido (rw-)
$ ./checkfile /private
/private: inválido
```

Exercício 02) Escreva um *shell script* que recebe três números inteiros pela **entrada padrão** (e não via argumentos em linha de comando) e mostre qual o maior deles.

```
$ ./higher
Primeiro número: 5
Segundo número: 12
Terceiro número: 8

O maior número é o 12
```

Exercício 03) Escreva um *shell script* para cumprimentar um usuário com `Bom dia`, `Boa tarde` ou `Boa noite` dependendo do horário do sistema. A saudação deve incluir o nome completo do usuário atualmente logado. Dica: use o comando `whoami` para obter o usuário atual e depois consulte o arquivo `/etc/passwd` na quinta coluna para obter o seu nome completo. As horas do dia são divididas da seguinte maneira:

- **Dia**: até 12:00
- **Tarde**: a partir de 12:00 e até 18:00
- **Noite**: a partir das 18:00

Exemplo de utilização do *shell script*.

```
$ date
Sun Nov 30 12:40:01 BRST 2014
$ ./greeting
Boa tarde João do Caminhão, seja bem vindo.
```

Exercício 04) O comando `expr` é capaz de realizar cálculos com expressões aritméticas inteiras. Reescreva um programa similar em *shell script* para calcular uma expressão com apenas um operador com dois números em ponto flutuante e imprima o resultado em ponto flutuante. Os operadores suportados com seus respectivos símbolos são: adição (`+`), subtração (`-`), multiplicação (`x`), e divisão (`/`). **Você deve usar obrigatoriamente a estrutura** `case` **como controle de fluxo condicional.** O programa deve ler os dois números e o operador via parâmetros em linha de comando. Caso não sejam especificados os três parâmetros, deve-se mostrar uma mensagem de ajuda. Se o operador não for um dos quatro suportados deve-se mostrar uma mensagem de erro. Dica: use o comando `bc` (*basic calculator*) para fazer as operações em ponto-flutuante. Exemplo de uso do comando **`bc`**.

```
$ echo "22 / 7" | bc -l
3.14285714285714285714
```

Exemplo de uso do *shell script*.

```
$ ./calc
Usar: ./calc [Número] [Operador] [Número]
$ ./calc 22 / 7
3.14285714285714285714
$ ./calc 12 - 7.2
4.8
$ ./calc 5 % 4
%: operação inválida
```

Estruturas de Repetição

Exercício 05) Escreva programas em *shell script* para cada um dos itens a seguir. **Cada *script* deve usar obrigatoriamente o comando** `for`.

a) Mostre os sete dias de semana usando código de três dígitos indicando os que são dia de semana e quais são fim de semana.

```
$ ./days
Mon (weekday)
Tue (weekday)
Wed (weekday)
Thu (weekday)
Fri (weekday)
Sat (weekend)
Sun (weekend)
```

b) Mostre todos os usuários do arquivo `/etc/passwd`.

```
$ ./users
User 1: root
User 2: daemon
User 3: nobody
...
```

c) Sorteie `n` rodadas de dados de `1` a `6`, definidas pelo usuário via argumentos em linha de comando, e imprima a frequência de todos os lados.

```
$ ./dicesfreq 25
Dice 1: 4
Dice 2: 5
Dice 3: 7
Dice 4: 2
Dice 5: 4
Dice 6: 3
```

d) Receba um ou mais números via argumentos em linha de comando e mostrar o menor e o maior deles. Restrição: não é permitido usar o comando `sort` nem outro comando similar.

```
$ ./edges 10 7 81 40 74 22 15 21 84 74
Lower: 7
Higher: 84
```

Exercício 06) Escreva um *shell script* para treinar um usuário com operações básicas de aritmética: soma e subtração. Esse *script* poderia ser colocado na inicialização de cada terminal, obrigando o usuário a informar o resultado correto para ter acesso a ele. O *script* deve gerar dois números aleatórios entre `1` e `999` (inclusivos) que serão usados como operandos. O *script* também deve gerar aleatoriamente um dos operandos de adição (`+`) ou subtração (`-`). O *script* deve imprimir exatamente conforme exemplo a seguir, alinhando os números à direita. Usuários não podem abortar o programa com `CONTROL+C`, `CONTROL+D` ou `CONTROL+Z`. **O *script* deve usar obrigatoriamente o comando** `while`. Dica: use o comando `printf` para imprimir a conta e o comando `expr` para calcular o resultado.

```
$ ./basic_math
     43
  - 287
  -----
? -143
Wrong answer
? -244
Right answer
$
```

Exercício 07) Escreva um jogo de adivinhação de um número em *shell script*. O programa deverá selecionar um número aleatório entre `1` e `1000` (inclusivos) sem informar ao usuário qual. Se o usuário acertar o número, o programa deve informar que o usuário venceu; caso contrário, o programa deve informar se o número sorteado é menor ou maior que aquele escolhido pelo usuário. O jogo continua indefinidamente até que o usuário acerte o número. **O *script* deve usar obrigatoriamente o comando** `until`. Dica: não é preciso fazer validação se o valor digitado é numérico.

```
$ ./guess
Try to guess the number I thought!

Choose a number (1 to 1000): 500
  My number is higher than 500
Choose a number (1 to 1000): 750
  My number is lower than 750
Choose a number (1 to 1000): 625

Congratulations, you guessed correctly
$
```

Exercício 08) Desenvolva um *shell script* para verificar se uma frase é um palíndromo, imprimindo se sim ou não. O programa deve receber da entrada padrão frases em cada linha e deve terminar quando receber uma frase vazia. Você deve considerar apenas os caracteres do alfabeto inglês (sem acentos), ignorando todos os outros. Dica: use o comando `sed` para filtrar os caracteres inválidos `sed 's/[^A-Za-z]//g'`.

```
$ cat input.txt
reviver
lamina animal
Ser ou nao ser?
A Daniela ama a lei? Nada!
$ ./palindrome < input.txt
Yes
Yes
No
Yes
```