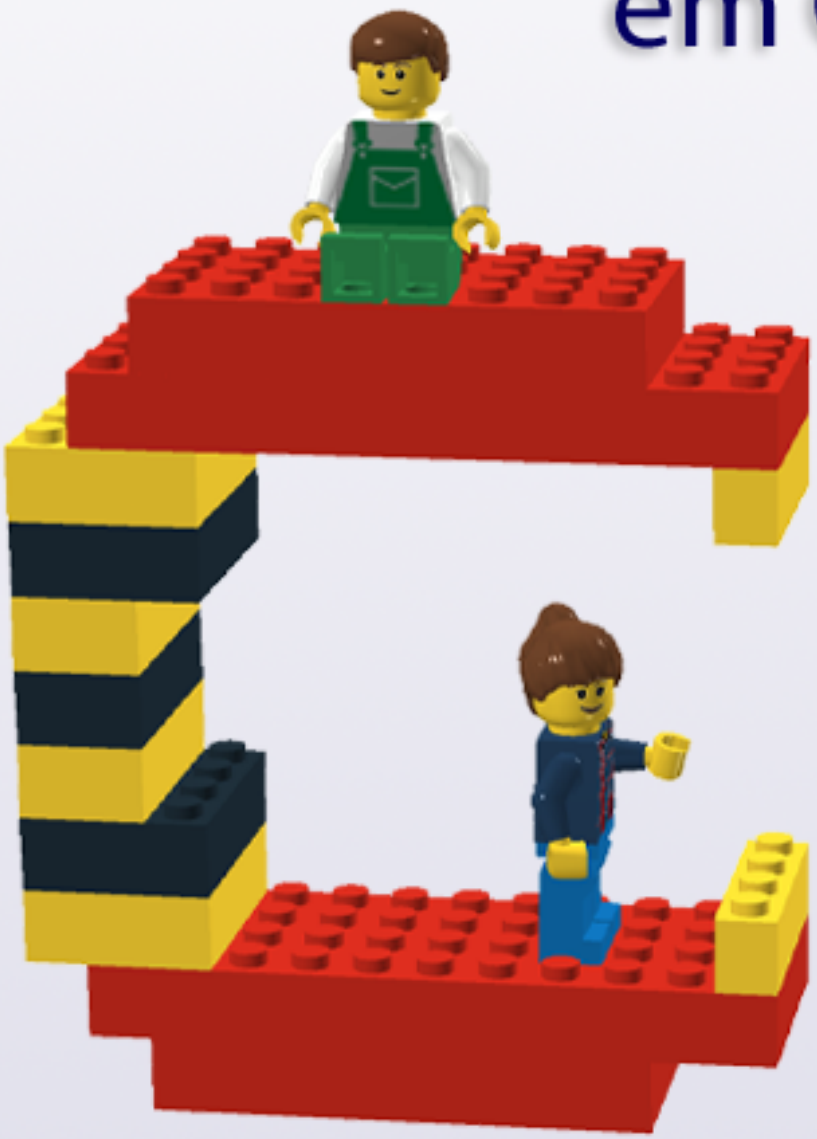
PROGRAMAR

A REVISTA PORTUGUESA DE PROGRAMAÇÃO

Revista nº12 - Janeiro de 2008

www.portugal-a-programar.org

programação orientada a objectos em C#



e ainda...

Programação Saudável // GeexBox - Faça o seu Media Center

índice

equipa PROGRAMAR

contacto
revistaprogramar
@portugal-a-programar.org

website
www.revista-programar.info

Programadores que não sabem programar?

Enquanto lia alguns dos tópicos mais populares do ano no que toca à programação na popular comunidade Digg (www.digg.com), deparei-me com um artigo intitulado "Why Can't Programmers... program!", em que se referia que na maior parte das entrevistas de emprego a programadores precedidas de recolha de currículos, a maior parte dos entrevistados não superava algumas das mais básicas tarefas solvidas com recurso a uma linguagem de programação, da ordem das que alguém que começou a programar há dias tenta fazer.

Pode-se argumentar que bons programadores são desde logo assediados pelas empresas enquanto tiram os seus cursos, não precisando deste género de entrevistas, mas mesmo os que não conseguem tal sorte, não me parece, pelo menos em Portugal, que falhassem em questões básicas sobre programação, como no exemplo dado no artigo original: pedir ao programador que faça um output de todos os números de 1 a 100, excepto os múltiplos de 3, que seriam imprimidos como "Fizz" e os de 5 que seriam "Buzz" (os múltiplos de ambos como "FizzBuzz").

Ou há algo de errado no mundo, ou há algo de errado nesta profissão. Será que com a divulgação dos mais diversos tutoriais e snippets na web com a característica comum de quererem dar a solução em vez de explicar como se chega a tal, a profissão de programador começa a ter, dentro de si, uma larga percentagem de indivíduos que, na mais fiel das denominações, seriam intitulados programadores do "desenrasque"? Mas mais surpreendente do que haver tal percentagem é de facto ter programadores dessa ordem empregados. Terá a programação já se especializado assim tanto que nem a reprovação na concepção do mais básico algoritmo seja indicador de mau programador? Será preocupante se sim, pois teremos muitos dos nossos serviços do dia a dia mantidos e assegurados por programadores que numa possível situação inesperada não saberão tomar as medidas adequadas para resolver o problema.

No entanto, o mundo emprega-os...

Queres ajudar a Revista PROGRAMAR?
Vê como na última página.

Novo MacBook Air

A Apple conseguiu fazer um portátil incrivelmente fino e ainda usar um ecrã de 13.3 polegadas e um teclado no seu tamanho normal. Tem apenas menos de 2 cm de espessura na parte superior e 0.4 cm na parte inferior. Pesa 1,36 kg. O ecrã de 13.3" é em vidro, 1200 por 800 widescreen e com backlit LED. Este ecrã é do mesmo tamanho do do Macbook.

As teclas do teclado do Air são como as do Macbook, na cor preta. A diferença está em que este é um teclado retro-iluminado que se acende quando o sensor detecta a diminuição de luz. O trackpad também é especial - integra vários movimentos multi-touch também utilizados no iPhone como o scroll, o pinch para o zoom, ou o rotate.

Os únicos 3 conectores do Macbook Air estão escondidos. Basta deslizar uma espécie de tampinha e eles surgem: o buraco dos headphones, do usb e micro-DVI. A saída micro-DVI suporta DVI, VGA, vídeo composto e S-video. Até o carregador MagSafe teve que ser redesenhado e sofrer uma dieta para o Air.
Está também equipado com 2Gb de RAM incorporados, 80Gb de hard drive ou, por opção, uma drive de 64Gb solid-state drive sem partes movíveis. O processador é agora uma versão mais pequena do Intel Core 2 Duo a 1.6GHz ou, opcionalmente, a 1.8GHz. Como todos os outros Macbook tem câmera iSight que agora é circular. A bateria não é removível e tem uma durabilidade de até 5 horas.

O Macbook Air está previsto para entrega dentro de uma a duas semanas com o preço a rondar os 1800€ na versão standard.

Oracle compra BEA Systems

Depois de uma primeira oferta sem sucesso, a Oracle conseguiu agora ir ao encontro dos desejos da administração da BEA Systems e chegar a um preço que permitirá concretizar o negócio de aquisição da empresa.

O novo preço, de 8,5 mil milhões de dólares, representa um prémio de 14 por cento face à primeira oferta apresentada em Outubro passado e resulta numa proposta de 19,3 dólares por acção, contra os anteriores 17 dólares.

A BEA Systems opera no mercado de middleware, área onde a Oracle tem vindo a reforçar competências, nomeadamente pela via das aquisições e que pretende liderar a nível mundial. O middleware é aliás o foco do projecto Fusion em que a Oracle tem vindo a trabalhar para integrar o seu software com o das empresas que tem vindo a adquirir nos últimos anos, nomeadamente a PeopleSoft.

Sun adquire MySQL AB

A Sun adquiriu a MySQL AB o que pode ter um impacto grande no mercado de bases de dados.

A empresa referiu que está apostar milhares de milhões com o MySQL. Referiu também que vai oferecer um novo suporte ao mercado do MySQL, mas também vai dar suporte à comunidade com o objectivo de acelerar o processo das empresas saírem do software proprietário.

Segundo a Sun: "As pessoas do MySQL, desde os empregados, utilizadores e patrocinadores - bem-vindos, estamos contentes por nos juntar-mos convosco. Esta aquisição significa o começo duma nova era na Internet."

Apesar de ser um mercado onde a Oracle lidera indiscutivelmente, o facto de ter a Sun por trás e ter um modelo diferente pode chegar para começar a "assustar" a Oracle.

Quanto à utilização hoje feita do MySQL em plataformas Web, o impacto não será tão grande, já que o MySQL é hoje o líder. Curiosamente, tanto nos grandes (Flickr, Sapo, Google...) como nos pequenos.

Programação Orientada a Objectos em C#

C# é uma linguagem de programação relativamente recente, criada pela Microsoft, desenvolvida juntamente com a arquitectura .NET. C# pronuncia-se 'cê sharp' e não 'cê cardinal', foi criada praticamente do zero para funcionar na nova plataforma, sem preocupações de compatibilidade com código de legado. Ultimamente, as empresas de software têm apostado cada vez mais na plataforma .NET e consequentemente no C#. Isto deriva do facto de ser uma linguagem simples, orientada a objectos (como se irá ver mais à frente), que tira partido dos serviços de segurança, da gestão automática de memória e do vasto conjunto de bibliotecas da plataforma .NET. A linguagem foi baseada em Java e C++, tirando partido das melhores características das duas, pelo que qualquer programador que esteja à vontade nestas duas linguagens não vai sentir muitas dificuldades em aprender C#. É portanto, uma linguagem que permite desenvolver software com grande robustez, respondendo aos exigentes requisitos das empresas.

Não é objectivo deste artigo ensinar e descrever as características aprofundadas da linguagem C#, no entanto pode-se dizer que tem uma sintaxe bastante parecida à do Java. Tem as mesmas expressões de controlo de fluxo que as linguagens derivadas do C, como por exemplo o if-else, o while, o for, entre outros. Também possui bastantes estruturas próprias da linguagem, mas esse é um assunto que o leitor deverá aprofundar se não estiver à vontade.

Esta pequena descrição da linguagem C# serviu de introdução para o apoio ao tema central deste artigo, que é a Programação Orientada a Objectos. Para que o leitor fique a perceber melhor como funciona este paradigma vão ser mostrados alguns exemplos de código C#.

Programação Estruturada vs Programação Orientada a Objectos

Hoje em dia, praticamente todo o software está a ser escrito em linguagens de programação orientadas a objectos (POO). Linguagens de programação estruturada como o C e o Pascal estão a ser deixadas para trás, pois funcionam bem para pequenos programas mas quando a complexidade começa a aumentar podem surgir vários problemas. Por exemplo, num programa em C todas as operações têm acesso a todos os dados, e uma mudança num dos módulos pode ter implicações em todo o programa. A programação orientada a objectos alivia alguns desses problemas, encapsulando cada uma das estruturas de dados num objecto. O objecto também possui funções, chamadas em POO de métodos, que permitem interagir com o objecto.

POO – Características

A POO assenta em três conceitos base:

- Encapsulamento de informação;
- Herança;
- Polimorfismo.

- Encapsulamento de informação

Um dos pontos fulcrais do POO é esconder as estruturas de dados em objectos, aos quais são associados métodos que manipulam essas estruturas. Para este processo o programador tem de definir classes, que são um tipo abstracto de dados ou uma família de dados. Cada classe costuma ter as variáveis, método(s) construtor(es), métodos que modificam ou devolvem o valor das variáveis. Antes de entrar em detalhes sobre o que é cada um, é melhor tomar em atenção o seguinte código:

```
class Aluno
{
    private int Numero; //Número do
aluno
    private string Nome; //Nome do
aluno
    private string NomeEscola; //Nome
da escola

    public Aluno(int numeroAluno,
string nomeAluno, string nomeEscola)
    {
        Numero = numeroAluno;
        Nome = nomeAluno;
        NomeEscola = nomeEscola;
    }

    public void MostraAluno()
    {
        Console.WriteLine("O aluno
número {0} chama-se {1} e estuda na
escola {2}.", Numero, Nome,
NomeEscola);
    }
```

```
    public int DevolveNumero()
    {
        return Numero;
    }

    public void ModificaNumero(int
Numero)
    {
        this.Numero = Numero;
    }
}
```

A classe `Aluno` possui três variáveis declaradas como `private`: `Numero`, `Nome` e `NomeEscola`. Isto significa que só podem ser acedidas a partir da própria classe, ou seja, a informação é escondida dentro da classe. Se estivesse declarada como `public` podia ser acedida a partir de uma classe externa. Pode-se observar a existência de um construtor: `public Aluno()`. A sua função é a de criar uma nova instância de uma classe, ou seja, cria um objecto dessa classe. Também irá guardar a informação dos atributos que são passados no construtor no interior da classe, tal como é visto no código, já que as variáveis da classe tomam o valor das passadas no construtor. Para criar o objecto do tipo `Aluno` basta fazer:

```
Aluno aluno1 = new Aluno(10, "Filipe",
"Escola XPTO");
```

Depois existem três métodos. O primeiro, `MostraAluno()`, quando é chamado imprime a frase com os dados do aluno, ou seja, ao invocar `aluno1.MostraAluno()` vai imprimir a seguinte frase:

"O aluno número 10 chama-se Filipe e estuda na escola Escola XPTO."

O método `DevolveNumero()` é chamado sempre que seja necessário saber o número do aluno. Como a variável `Numero` da classe `Aluno` está declarada como `private`, é necessário criar esse método para saber o seu valor. A seguinte invocação:

```
Console.WriteLine("O aluno tem o
número {0}.", aluno1.DevolveNumero())
```

resulta na frase: "O aluno tem o número 10."

O último método, `ModificaNumero(int Numero)`, é usado para modificar o número do aluno, já que a variável `Numero` está declarada como `private` e é necessário este método que para entidades exteriores à classe possam modificar o seu conteúdo. Desta forma, a seguinte sequência:

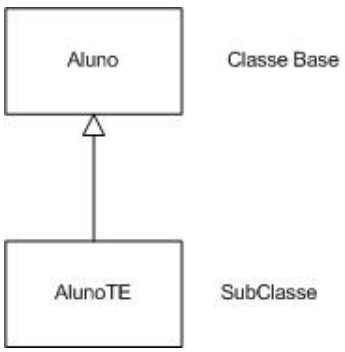
```
aluno1.ModificaNumero(20);
aluno1.MostraAluno();
```

imprime a frase: "O aluno número 20 chama-se Filipe e estuda na escola Escola XPTO."

De notar que no método que modifica a idade do aluno foi usado a cláusula `this`, que significa o próprio objecto. Como a variável da classe e a passada por referência no método tinham o mesmo nome foi necessário indicar que quem tomava o novo valor era a variável do objecto. É de notar que as outras classes podem chamar estes métodos porque estão declarados como `public`; caso estivessem declarados como `private` os métodos só podiam ser acedidos a partir da própria classe.

- Herança

Esta é uma das características mais visíveis e com mais uso quando se desenvolve uma aplicação usando POO. Para compreender melhor o conceito de Herança, vai-se tomar em conta a classe Aluno anteriormente descrita. Imagine-se, agora, que é necessário criar mais uma classe para os alunos que são trabalhadores estudantes e que essa classe apenas vai ter mais um campo (o nome da empresa onde trabalha) que a classe Aluno. Será que faz sentido estar a criar uma classe AlunoTE que repita os mesmos atributos que a classe Aluno? Afinal de contas, um trabalhador estudante é um aluno, e como tal possui todas as características de um aluno normal. Assim surge o conceito de herança, ou seja, é possível criar uma nova classe que derive de outra. Contudo nem sempre o conceito de herança é bem utilizado. Apenas se deve usar quando uma classe é uma especialização da outra, isto é, quando possui todos os elementos que a outra possui mais alguns, sejam estes métodos ou dados. É usual chamar à classe geral de superclasse ou de classe base, enquanto as que derivam destas são chamadas de subclasses.



Para perceber melhor o conceito de herança, em seguida está exemplificado o código da classe `AlunoTE` que deriva da classe `Aluno`:

```
class AlunoTE : Aluno
{
    private string NomeEmpresa; //Nome
da empresa

    public AlunoTE(int numeroAluno,
string nomeAluno, string nomeEscola,
string NomeEmpresa)
            : base(numeroAluno,
nomeAluno, nomeEscola)
    {
        this.NomeEmpresa =
NomeEmpresa;
    }

    public void MostraAlunoTE()
    {
        Console.WriteLine("O aluno
trabalha na empresa {0}.",
NomeEmpresa);
    }
}
```

Como se pode observar, a classe `AlunoTE` é definida como sendo subclasse de `Aluno` logo no início: `class AlunoTE : Aluno`. Desta forma herdou todas as variáveis e métodos presentes na classe `Aluno`. Em seguida é definido um atributo, que é o nome da empresa em que o aluno trabalha. Depois vem o método construtor, que recebe como argumentos todos os atributos da classe `Aluno` mais o atributo `NomeEmpresa`. Nota-se uma novidade que é o uso do `: base()`. O que isto faz é inicializar as variáveis na classe base antes de fazer o mesmo na própria classe. Depois disso é feita a inicialização da variável `NomeEmpresa` no corpo do construtor da própria classe. Também foi criado um método local que imprime o nome da empresa onde o aluno trabalha. Por fim, já se podem criar objectos da classe `AlunoTE`, e fazer uso das funcionalidades que a herança permite, como por exemplo, o seguinte código:

```
AlunoTE aluno2 = new AlunoTE(15,
"Joel", "Escola XPTO",
"SoftRevistaPROGRAMAR");
aluno2.MostraAluno();
aluno2.MostraAlunoTE();
```

vai imprimir: "O aluno número 15 chama-se Joel e estuda na escola Escola XPTO." "O aluno trabalha na empresa SoftRevistaPROGRAMAR."

- Polimorfismo

Por polimorfismo entende-se a capacidade de objectos diferentes se comportarem de modos distintos quando lhes é chamado o mesmo método. Por exemplo, a classe `AlunoTE` possui um método que imprime o nome da empresa, para imprimir o número, o nome e o nome da escola tem de chamar o método da classe base. Mas pode ser feita a redefinição do método `MostraAluno()` e fazer com que se imprimam todos os dados, tanto da classe base como da derivada. Em C# basta criar um método com o mesmo nome na classe derivada (declarado como `override`), declarando ao mesmo tempo o método da classe base como virtual. Na classe `Aluno` o método ficaria assim:

```
public virtual void MostraAluno()
{
    Console.WriteLine("O aluno número
{0} chama-se {1} e estuda na escola
{2}", Numero, Nome, NomeEscola);
}
```

Na classe `AlunoTE` seria criado o novo método da seguinte forma:

```
public override void MostraAluno()
{
    Console.WriteLine("O aluno nÃúmero
{0} chama-se {1}, estuda na escola {2}
e trabalha na empresa {3}.", Numero,
Nome, NomeEscola, NomeEmpresa);
}
```

Desta forma quando for criado um aluno do tipo `AlunoTE` e invocar esse método:

```
AlunoTE aluno3 = new AlunoTE(5,
"Pedro", "Escola XPTO",
"SoftRevistaPROGRAMAR");
aluno3.MostraAluno();
```

vai obter a mensagem:

"O aluno número 5 chama-se Pedro, estuda na escola Escola XPTO e trabalha na empresa SoftRevistaPROGRAMAR."

Níveis de acesso a membros

Ao longo deste artigo têm sido dados exemplos de atributos, métodos e classes declarados como public e private. No entanto existem outros níveis de acesso a membros, como é o caso do `protected`, `internal` e `protected internal` (os dois últimos são características do C# - em Java, por exemplo, não existem).

Nível de acesso	Visibilidade
Private	Dentro da própria classe
Protected	Dentro da própria classe e em classes derivadas
Public	Dentro e fora da própria classe
Internal	Como os membros public, mas apenas dentro do módulo corrente
Protected Internal	Como os membros protected, mas apenas dentro do módulo corrente

Membros estáticos

Por vezes surge a necessidade que determinadas classes partilhem informação. Por exemplo, imaginando que todos os alunos andam na mesma escola, no caso da classe `Aluno` existe um campo que é comum a todos os alunos da mesma escola, que é o nome da escola (`NomeEscola`). Sempre que haja uma alteração no nome da escola, deve ser visível em todas as instâncias da classe. Surge então o conceito de `static`. No exemplo da classe `Aluno`, imagine-se que o nome da escola era declarado como `static`:

```
private static string NomeEscola;
```

Este atributo vai ser partilhado por todas as instâncias da classe Aluno. Para alterar o seu conteúdo, declara-se também um método estático:

```
public static void
ModificaEscola(string novoNome)
{
    this.NomeEscola = novoNome;
}
```

Se houver dois objectos e o método for chamado apenas num, o outro objecto também vai alterar o nome da escola.

Construtores estáticos

Tal como foi dito anteriormente, um membro estático não está associado a nenhum objecto em concreto, por isso faz todo o sentido o membro estático ser inicializado dentro de um construtor estático. Um construtor estático apenas difere dos apresentados anteriormente pelo facto de ser declarado como `static`.

```
private static string Atributo;
static ClasseXPTO()
{
    Atributo = "AtributoX";
}
```

Conclusão

Com este artigo pretendeu-se que o leitor fique a perceber, de uma forma superficial, o paradigma de programação orientado a objectos. Partiu-se do princípio que o leitor já possui alguns conhecimentos em C#, ou então em linguagens cuja sintaxe é parecida, podendo desta forma dar alguns exemplos práticos. Nem todos os aspectos foram tratados neste artigo, pelo que se aconselha a pesquisar sobre este assunto, nomeadamente no que diz respeito a classes seladas, classes abstractas e interfaces.

Bibliografia

MARQUES, Paulo; PEDROSO, Hernâni - C# 2.0 . Lisboa: FCA, 2005. ISBN 978-972-722 208-8

http://pt.wikipedia.org/wiki/C_Sharp
http://msdn.microsoft.com/vcsharp/
http://en.wikipedia.org/wiki/Object-oriented_programming

SOBRE O AUTOR

Residente em Amarante, Pedro Teixeira é actualmente finalista do curso de Engenharia Informática. Colaborador da revista PROGRAMAR quase desde o início, este é o 4º artigo que produz para todos os que desejam saber mais sobre programação e afins. Gosta bastante da área de redes de computadores, seja na vertente programação, seja na de segurança.

Pedro Teixeira

Shell Script

Shell é a linha de comandos de Linux e Unix, é ela que interpreta todos os comandos inseridos pelo utilizador. Para além de executar comandos do sistema esta também tem comandos de programação tais como IF, FOR, WHILE, variáveis e também funções, permitindo desta forma criar chamadas de comandos mais flexíveis.

Shell Script é basicamente um ficheiro com uma determinada sequência de comandos Shell, estes scripts podem ser executados de diversas formas podemos criar o nosso script e executá-lo usando os comandos "sh" ou "bash".

```
sh Script.sh
```

ou

```
bash Script.sh
```

Ambos executam o script, com uma diferença, o "sh" é o interpretador standard de comandos Shell, por sua vez o "bash" é o interpretador de comandos Shell tal como o "sh", mas com mais algumas funcionalidades, nomeadamente o suporte de variáveis do tipo Array, Select, Read, etc... Por esse motivo vamos usar o "bash".

Uma outra forma de executar o nosso script é tornando-o executável. Para isso na primeira linha do nosso script colocamos o caminho para o interpretador, por exemplo:

```
#!/bin/bash
printf "Hello World"
```

Em seguida executamos o comando

```
chmod +x Script.sh
```

para tornar o nosso script executável e depois basta corrê-lo.

Vamos agora passar à prática, começando por ver como definir variáveis para o nosso Shell Script. A declaração de variáveis é bastante simples, basta colocar o "nome=valor" sem espaços. Vamos ver vários exemplos de declaração de variáveis.

```
#!/bin/bash

Nome="White Magician";
NUMERO=70101;
_ID1=1;
_id2="ABFISOEEQS";

echo $Nome
echo $NUMERO
echo $_ID1;
echo $_id2;
```

Como podemos ver as variáveis podem ter nomes com letras de a..z e A..Z, números e "_". Embora possam conter números estes não podem ser colocados no início, pois variáveis iniciadas com um número estão reservadas para a Shell. O uso de ';' não é obrigatório como podemos ver, mas existem casos em que o seu uso faz parte da sintaxe. Veremos esses casos mais à frente. No exemplo para aceder ao valor guardado numa variável usamos o prefixo '$'.

As variáveis podem aí ser de valor constante, ou seja, o seu valor não pode ser alterado, para isso usamos o comando readonly como podemos ver em seguida.

```
#!/bin/bash

var=1000;
readonly var;
readonly var2=12;
var=9;
```

Ao executar este script ele irá retornar um erro dizendo que a variável var é apenas de leitura. Podemos declarar a variável e em seguida declará-la como readonly ou então fazer os dois passos de uma vez como podemos ver.

As variáveis podem aí ser eliminadas, ou seja apagadas da lista de variáveis da Shell. Para isso basta usar comando unset, mas este comando apenas funciona em variáveis não declaradas como readonly.

```
#!/bin/bash

var1=100;
readonly var2=10;
unset var1;
unset var2;

echo $var1;
echo $var2;
```

Ao executar este script podemos ver duas operações importantes. Primeira, ao fazer echo $var1 não será impresso nenhum valor, visto esta variável já não existir. Segunda, o retorno de um erro devido à tentativa de eliminar uma variável readonly e que ao ser impresso o seu valor foi mantido.

Uma estrutura muito útil para a criação de scripts dinâmicos e mais flexíveis é o array, também disponível na nossa Shell, mas apenas no "bash" e não no "sh". Os arrays podem ser facilmente usados com a seguinte sintaxe "nome[índice]=valor". Vamos agora ver o seu funcionamento.

```
#!/bin/bash

lista[0]=9999;
lista[1]=1111;
lista[10]=333;

echo ${lista[0]};
echo ${lista[1]};
echo ${lista[10]};
echo ${lista[@]};
```

Aqui podemos ver um pequeno exemplo da utilização de Array em Shell Script. O índice '@' selecciona todos os índices, ou seja, retorna todos os valores guardados no Array.

O próximo exemplo mostra algumas das mais usadas variáveis de ambiente da Shell.


```
#!/bin/bash

echo $PWD   #Caminho para a directoria
actual.
echo $UID   #ID do utilizador que iniciou
a Shell
echo $SHLVL     #Número de sessões Shell
abertas.
echo $REPLY     #Contém o último input
recebido pela função read não suportado
pelo "SH"
echo $RANDOM    #Cria um número aleatório
entre 0 e 32767 não suportado pelo "SH"
echo $PATH  #Caminhos de pesquisa para
comandos das Shell.
echo $HOME  #Caminho para a HOME do
utilizador actual.
```

Ainda no âmbito das variáveis é-nos possível aceder a variáveis especiais. Podemos ver uma pequeno exemplo do uso dessas variáveis:

```
#!/bin/bash

echo $0     #Nome do Script.
echo $1     #Valor do primeiro argumento
do Script.
echo $2     #Valor do segundo argumento
do Script.
echo $#     #Número de argumentos do
Script.
echo $@     #Todos os valores dado ao
Script.
echo $$     #Número do processo do
Script.
echo $!     #Número do último comando em
background.
```

Vamos agora ver uma alternativa ao comando echo para imprimir dados. O comando printf permite, à semelhança do comando echo, imprimir dados com a vantagem de que consegue fazê-lo de forma formatada, sendo o seu funcionamento é praticamente igual ao printf da linguagem C/C++. Vamos ver então alguns exemplo do nosso printf.

```
#!/bin/bash

printf "Olá %s bem vindo a %s\n"$USER
$HOME
printf "Basta colocar o texto entre \"\"
\n"
printf "E em seguida as variáveis
separadas por um espaco!\n"
```

Como podemos ver é bastante simples usar o comando printf. É de referir que a ordem pela qual as variáveis são

colocadas será a mesma na altura da impressão ou seja colocar $USER $HOME tem um resultado diferente de colocar $HOME $USER.

Algo muito importante para o nosso código são os comentários em Shell Script, que podem ser feitos com o símbolo #."Este caracter indica que o restante da linha, a partir do caracter, deve ser ignorada pelo bash. Mas como já devem ter reparado os nossos scripts começam com #!/bin/bash, que é um caso especial - esse comentário é usado para indicar qual a shell que deve ser usada para interpretar o script, no nosso caso a "bash". Esta linha deve ser a primeira do nosso script, caso contrário será ignorada pelo interpretador. Vamos agora ver alguns exemplos.

```
#!/bin/bash

#Todo o texto aqui escrito não será
impresso.
printf "Podemos ainda comentar ..."#
apenas parte da linha
```

Infelizmente os comentários de Shell Script não permitem o comentário em bloco por isso apenas podemos comentar um linha na totalidade ou parte dela, indicando o início do comentário mas não o seu fim na linha.

Já sabemos como imprimir dados - vamos agora ver como pedir dados ao utilizador. Para isso, temos o comando read, que permite ler o input do utilizador:

```
#!/bin/bash

printf "Utilizador : ";
read usr;
printf "Password : ";
read pass;

printf "%s a sua password é %s\n"$usr
$pass;
```

Passemos agora a estruturas condicionais. Shell Script dispõe de estruturas como IF e ELSE, existentes em praticamente todas as linguagens de programação, bem como o CASE (também conhecido por SWITCH noutras linguagens). Vamos ver um exemplo da sua utilização:

```
#!/bin/bash

read USER;

case "$USER" in
    magician) printf "Magician seja bem
vindo.\n";;
    root) printf "Bem vindo ao painel de
Admin.\n";;
    *) printf "Utilizador não
reconhecido\n";;
esac;
```

Como podemos ver a utilização do CASE é bastante simples. Damos uma variável e várias opções, dependendo do valor da variável o resultado será diferente. A opção * serve como um default para quando todas as outras falham. Na estrutura IF ELSE iremos usar um outro comando, o TEST. Este comando permite fazer testes booleanos em ficheiros, strings e números. Mas vamos ver este comando após o IF:

```
#!/bin/bash

read usr;

if test $usr = "magician";
        then
                printf "Bem vindo
Magician\n";
    else
        printf "%s não
reconhecido!\n"$usr;
fi
```

O comando TEST realiza testes entre dois valores e tem várias opções. Vamos ver todas as opções que temos, comecemos então pelas opções em ficheiros.

Opções em Ficheiros:
-b FILE True se o ficheiro existe e é um ficheiro bloco especial.
-c FILE True se o ficheiro existe e é um ficheiro de caracteres especiais.
-d FILE True se o ficheiro existe e é um directório.
-e FILE True se o ficheiro existe.
-f FILE True se o ficheiro existe e é um ficheiro regular.
-g FILE True se o ficheiro existe e contém o seu SGID.
-h FILE True se o ficheiro existe e é um link simbólico.
-k FILE True se o ficheiro existe e contém "sticky" bit set.
-p FILE True se o ficheiro existe e é um named pipe.
-r FILE True se o ficheiro existe e permite a sua leitura.
-s FILE True se o ficheiro existe e o seu tamanho é maior que zero.
-u FILE True se o ficheiro existe e contém o seu SUID.
-w FILE True se o ficheiro existe e permite ser escrito.
-x FILE True se o ficheiro existe e é executável.
-O FILE True se o ficheiro existe e o seu dono é o utilizador corrente.

Opções em Strings:
-z string True se a String tem tamanho igual a zero.
-n string True se a String tem tamanho diferente de zero.
string1 = string2 True se as duas Strings são iguais.
string1 != string2 True se as duas Strings são diferentes.

Opções em Números:
int1 -eq int2 True se int1 é igual a int2.
int1 -ne int2 True se int1 não é igual a int2.
int1 -lt int2 True se int1 é menor que int2.
int1 -le int2 True se int1 é menor ou igual a int2.
int1 -gt int2 True se int1 é maior que int2.
int1 -ge int2 True se int1 é maior ou igual a int2.

Opções em Expressões:
! expr True se expr é false.
expr1 -a expr2 True se expr1 e expr2 são ambas true.
expr1 -o expr2 True se expr1 ou expr2 é true.

Passemos agora aos ciclos existentes em Shell Script, nomeadamente FOR e WHILE. Comecemos pelo WHILE:

```
#!/bin/bash

x=0;
while [$x -lt 10]
do
    echo $x;
    x=`expr $x + 1`
done
```

A linha "x=`expr $x + 1`" pode parecer estranha, mas o que esta linha faz é incrementar um valor à variável x e enviar a expressão $x+1 para o comando expr, que processa a expressão matemática e retorna um valor numérico. O comando é colocado entre `` para dizer ao interpretador que deve executar a expressão e aguardar o valor retornado.

Temos agora o FOR. Este ciclo é bastante simples e menos dinâmico que o WHILE - basicamente o FOR percorre uma sequência de valores e coloca cada um dos valores numa variável temporária que muda de valor a cada iteração.

```
#!/bin/bash

for i in 1 9 2 4 3 5 7
do
    echo $i
done
```

Vamos agora ver como criar funções em Shell Script. A declaração de funções é bastante simples, basta colocar o nome da função seguido de () e um bloco limitado por {}:

```
#!/bin/bash

mg="Magician";
imprime(){
    printf "Olá %s\n"$mg;
}

imprime;
```

As funções, ao contrário da maioria das linguagens de programação, não são chamadas sob a forma imprime(), mas apenas com o nome da função sem o par de parêntesis.

Para terminar o nosso artigo irei mostrar como redireccionar o Input/Ouput dos nossos scripts. É possível passar prints para ficheiros e o conteúdo de ficheiros para outros ficheiros ou até mesmo para variáveis do nosso script.

```
#!/bin/bash

Text1="Este texto será gravado dentro do ficheiro texto1.txt";
echo $Text1 > texto1.txt;

Text2="Este texto será concatenado ao texto já existente no ficheiro texto1.txt";
echo $Text2 >> texto1.txt
```

Como podemos ver, é possível redireccionar o output de variáveis do nosso script para ficheiro usando um símbolo >, que apaga todo o conteúdo do ficheiro e insere o valor enviado. Também se podem usar os símbolos », que adicionam o valor enviado ao já existente no ficheiro. Em ambos os casos se o ficheiro não existir ele é criado.

Espero que tenham gostado do artigo. A partir deste momento têm o conhecimento básico sobre Shell Script que permite a criação de scripts para as mais diversas tarefas.

SOBRE O AUTOR

Fábio Correia é estudante de Engenharia Informática na Universidade de Évora. Partilhando o estudo com a moderação do fórum Portugal-a-Programar e a participação na Revista Programar, como um dos redactores mais activos, ainda tem tempo para explorar algumas das suas linguagens preferidas: Java, PHP e a recente D.

Fábio Correia

Grafos
2ª Parte

Depois de na primeira parte (ver Edição 10) termos feito uma introdução ao mundo dos grafos e de termos lançado as primeiras bases para uma melhor compreensão destas estruturas, chegou a altura de perceber, realmente, que problemas podem ser resolvidos.

Pode ficar já claro que o número de algoritmos que resolvem problemas baseados em grafos é enorme, demasiado extenso para ficar completo um artigo, portanto nada melhor do que começar pelos mais simples (e importantes). Preferi manter o nome dos algoritmos e dos problemas em inglês para ser mais fácil encontrá-los numa pesquisa.

Shortest Path (o caminho mais curto)

Este é certamente um dos problemas mais simples de enunciar. Por exemplo: "Dado um conjunto de cidades, ligadas por estradas, e o respectivo comprimento de cada uma, calcular a distância mínima a ser percorrida entre duas dadas cidades".

Podemos modelar isto com recurso a um grafo pesado, é demais evidente. Cada cidade representa um nó e cada estrada que liga dois nós representa um arco. Primeiro que tudo, é importante reparar: o facto de existir uma estrada "directa" entre a cidade A e a cidade B, não significa que o caminho mais curto entre as duas cidades seja essa estrada. Pode existir uma cidade C, intermédia, cujo caminho mais curto passe por ela.

Vamos então ver os principais algoritmos:

- Algoritmo de Dijkstra

Descrição

O algoritmo de Dijkstra é muito provavelmente o algoritmo mais conhecido no tratamento de grafos. Este algoritmo não encontra só o caminho mais curto entre duas cidades, mas sim o caminho mais curto entre uma cidade (chamada origem) e todas as outras.

Parte da seguinte importante e simples propriedade: - Se de todos os arcos que saem do nó A, o arco A-B é o mais curto, então a ligação mais curta entre estes dois nós é através desse arco. Bastante óbvio, uma vez que qualquer outro caminho usando um outro arco seria obrigatoriamente maior, partindo do princípio que todos os arcos têm um peso positivo, claro.

Pseudocódigo:

```
# distancia(j) distância do nó de origem ao nó
j
# pai(j) nó anterior ao j, do caminho mais
curto (para podermos reconstruir o caminho)

 1 Para todos os nós i
 2     distancia(i) = infinito         # não
alcancável
 3     visitado(i) = Falso
 4     pai(i) = nil # ainda nenhum caminho até
ao nó

 5 distancia(origem) = 0 # origem -> a origem
do percurso

 6 enquanto(nos_visitados < tamanho_do_grafo)
     # encontrar o vértice não visitado com a
distância mínima à origem; chamemos-lhe i
 7   assert (distancia(i) != infinito, "Grafo
não é conexo")

 8   visitado(i) = Verdadeiro # marcamos o
vértice i como visitado

    # actualizamos as distância para todos os
vizinhos de i
 9   Para todos os vizinhos j do vértice i
10       se distancia(i) + peso(i,j) <
distancia(j) então
11           distancia(j) = distancia(i) +
peso(i,j)
12           pai(j) = i
```

Análise de complexidade

Compreender esta secção implica ter conhecimentos de análise de complexidade que não vou explicar neste artigo.

Como podem ter constatado, o pseudo-código está "incompleto" por não explicar como encontrar o vértice não visitado de distância mínima.
Foi proposto por existirem duas possibilidades:
- uma pesquisa linear, percorrendo todos os vértices (a mais normal), originando numa complexidade $O(V^2)$
- utilizar uma heap para encontrar o mínimo, tendo uma complexidade O(V log E), significativamente mais rápido se o grafo for esparso (tiver uma relação arcos/vértices baixa)

Extensões

- Grafos direccionados não são um problema para este algoritmo.

- Como é possível concluir partindo da descrição, grafos com arcos com pesos negativos produzem respostas incorrectas, de modo que este algoritmo não pode ser usado nessa situação, devendo-se recorrer ao algoritmo de Bellman-Ford.

- Apesar de mais rápido (principalmente se utilizarmos uma heap), encontrar o caminho mais curto entre todos os vértices é mais simples recorrendo ao algoritmo de Floyd-Warshall.

Problemas Modelo

Movimentos a cavalo (Knight Moves)

Problema

Dadas duas posições num tabuleiro de xadrez, determinar o número mínimo de movimentos que um cavalo deve fazer para chegar de uma casa à outra.

Análise

O tabuleiro pode ser modelado como um grafo, sabendo que existe ligação entre duas casas se um cavalo se puder deslocar entre elas (e não se forem adjacentes, como acontece por exemplo num floodfill bfs).
Repare-se que dada uma posição sabemos imediatamente para quais se pode deslocar o cavalo, de modo que não precisamos de o representar em memória do modo "clássico", mas usar uma representação implícita.

Percursos do caminho de ferro (Railroad Routing - USACO Training Camp 1997, Contest 1 - simplificado)

Problema

É nos dado um mapa (matriz) onde m[i][j] representa uma dada elevação nesse quilómetro quadrado. Pretendemos construir um caminho de ferro que conecte 2 pontos.
O custo normal por quilómetro é de 100€. Carris que necessitem de ganhar ou perder elevação entre quadrículas, são cobrados 100€ + 3x variação_na_elevação. Uma mudança de direcção de 90º dentro de uma quadrícula implica um custo de 40€.
Qual o custo mínimo para construir o percurso?

Análise

Este é um problema standard de shortest path.
Contudo, em vez de criarmos um nó por quadrícula, devemos criar 4, representando as 4 possíveis direcções de entrar numa dada quadrícula. Agora podemos criar as ligações correspondentes, e resolvê-lo.

- Algoritmo de Floyd-Warshall

Descrição

E se em vez do caminho mais curto entre duas cidades, pretendemos construir uma matriz de adjacência m onde m[i][j] representa o caminho mais curto da cidade i à cidade j?
Nesse caso, esta é a solução mais simples.

A ideia é a seguinte: pegando num outro vértice, K, caso a distância m[i][j] seja maior do que a distância m[i][k]+m[k][j], podemos afirmar que encontrámos um novo caminho melhor que o anterior, e portanto podemos actualizar m[i][j]. Provar que esta formulação produz a resposta óptima já é mais complicado, e não vai ser abordado.

Pseudo-código

```
# dist(i,j) é a "melhor" distância até agora
do vértice i ao vértice j

# Comecemos com todas as ligações passando por
um único caminho

 1 Para i = 1 até n fazer
 2   Para j = 1 até n fazer
 3       dist(i,j) = peso(i,j)

 4 Para k = 1 até n fazer # k é o vértice
"intermédio"
 5   Para i = 1 até n fazer
 6       Para j = 1 até n fazer
 7           se (dist(i,k) + dist(k,j) <
dist(i,j)) então # caminho mais curto
 8               dist(i,j) = dist(i,k) +
dist(k,j)
```

Análise de complexidade

Este algoritmo é claramente $O(V^3)$, muito provavelmente o algoritmo com análise mais simples que existe.

Extensões

- Para este algoritmo pesos negativos não são tão devastadores como para o algoritmo de Dijkstra.

Desde que não existam ciclos negativos, o algoritmo funciona como pretendemos (caso exista um ciclo negativo ele pode ser percorrido tantas vezes quantas quisermos para ter sempre caminhos mais curtos).

Problemas Modelo

Diâmetro de um grafo

Problema
Dado um grafo conexo, não direccionado e pesado, encontrar os dois vértices que estão mais afastados.

Análise
Calcula-se o caminho mais curto entre todos os vértices e acha-se o máximo da matriz.

Minimal Spanning Tree (árvore de extensão mínima)

Numa "Minimal Spanning Tree" nós procuramos o conjunto mínimo de arcos necessários para conectar todos os nós.

Suponham que trabalham numa companhia que gere uma rede de auto-estradas. Recentemente, saiu uma lei que obriga a que os camionistas percorram os seus trajectos em estradas iluminadas com postes de 10m. Como os vossos postes só têm 7m, estão numa boa alhada.
Pretendem então substituir os postes em algumas das vossas auto-estradas, aquelas necessárias para que todas as cidades possam ser abastecidas convenientemente. Qual a distância mínima de autoestrada em que terão que substituir os postes?

Existem dois algoritmos muito usados na procura da árvore de extensão mínima, neste artigo decidi não abordar o algoritmo de Kruskal.

- Algoritmo de Prim

Descrição

As semelhanças entre o algoritmo de Prim e o algoritmo de Dijkstra são tantas que eu quando os aprendi facilmente os trocava. Para tal não acontecer, efectuava sempre o seguinte raciocínio, que no fundo é a sua explicação:

Suponhamos que tenho todos os nós conectados, excepto um. Saber qual o arco a usar de modo a conectar este nó aos restantes é fácil - posso escolher o de tamanho menor.
Vamos então partir de um ponto. Este ponto vai ficar unido ao resto da árvore através da ligação ao arco mais próximo.
Temos agora dois pontos. Qual o próximo ponto a unir? Fácil. O ponto mais próximo dos dois que ainda não foi unido.
Esta metodologia obviamente leva-nos à solução óptima. (e por ser tão simples preferi explicá-la a abordar o algoritmo de Kruskal).

Pseudo-código

```
  # dist(j) é a distância do nó j à árvore
  # pai(j) representa o nó até agora conectado
à MST mais próximo do nó J
 1   Para todos os nós i
 2     dist(i) = infinito        # sem
ligações
 3     intree(i) = Falso             # sem nós
na árvore
 4     pai(i) = nil

 5   tamanho_da_árvore = 1
# adicionar um nó à árvore
 6   custo_da_árvore = 0
 7   intree(1) = Verdadeiro
 8   Para todos os vizinhos j do nó i   #
actualizar as distâncias
 9      dist(j) = peso(1,j)
10     pai(j) = 1

11   enquanto (tamanho_da_árvore <
tamanho_do_grafo)
       # encontrar o nó com a menor distância
à árvore; chamar-lhe nó i
       # obviamente um nó que não esteja
"intree"
12     assert (distance(i) != infinity, "Grafo
não conexo")

    # adicionar arco pai(i),i à MST
13     tamanho_da_árvore = tamanho_da_árvore +
1
14     custo_da_árvore = custo_da_árvore +
dist(i)
15     intree(i) = Verdadeiro              #
marcar o nó i como estando na árvore

    # actualizar todas as distâncias depois do
nó i ter sido adicionado
16     para todos os vizinhos j de i
17       se (dist(j) > peso(i,j))
18         dist(j) = peso(i,j)
19         pai(j) = i
```

Análise de complexidade

Tal como para o algoritmo de Dijkstra, tempo de execução $O(V^2)$. Podemos obter O(V log E) se recorrermos a uma heap.

Extensões

- Funciona bem com grafos não pesados.

- Não há problema com múltiplos arcos entre dois nós (ignoramos todos menos o menor, obviamente)

- Se sofrer outras restrições (como 2 nós não poderem estar mais afastados do que X) então fica muito difícil de alterar.

- Não funciona com grafos direccionados (em que pretendemos que sejam fortemente conexos - seja possível aceder de um vértice a todos os outros).

Problemas modelo

Highway Building (construindo uma autoestrada)

Problema

Pretende-se construir uma rede de autoestradas que ligue as K maiores cidades de um país.
Sendo um país pobre, eles pretendem reduzir ao máximo o custo da obra. Sabendo que o custo de uma autoestrada é proporcional ao seu comprimento, e dadas as coordenadas das cidades, quais as cidades que devemos ligar para formar a requerida via rápida? (Nota: não se podem usar outros pontos como nós)

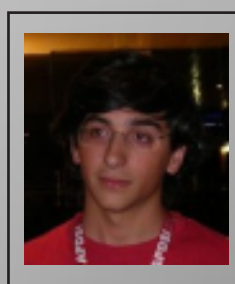
Análise

Um problema simples de MST, usamos as cidades como nós e consideramos todas as ligações possíveis como arcos.

Referências

USACO training pages - http://train.usaco.org

SOBRE O AUTOR



O especial interesse pela algoritmia nasceu enquanto frequentava no secundário o curso Científico-Tecnológico de Informática, que terminou recentemente com média de 19,5 valores. Também recentemente representou Portugal nas Olimpíadas Internacionais de Informática, na Croácia, em 2007, após vencer as Olimpíadas Nacionais, e em Mérida, no México, em 2006.

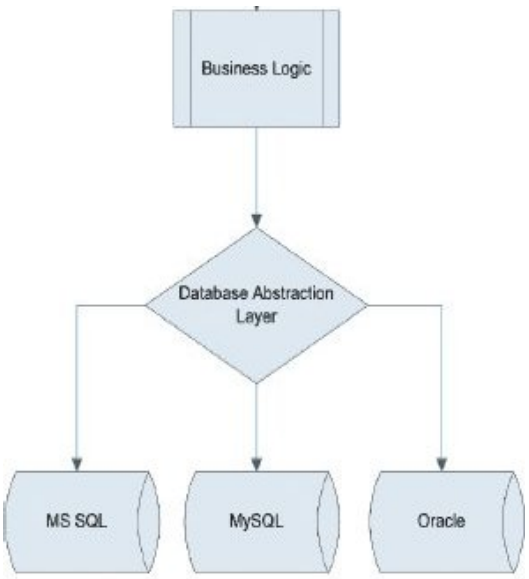
Miguel Araújo

Abstracção de SGDBs em PHP - AdoDB

Nos dias que correm, são muitos os que já se aventuraram no desenvolvimento de aplicações web-based. Quando essas aplicações começam a ganhar notoriedade, há uma grande possibilidade de serem usadas noutro ambiente diferente daquele em que sempre desenvolvemos. Um dos problemas comuns é o facto de termos de usar um software de gestão de base de dados (SGDB) diferente e, como escrevemos a aplicação para usar um determinado SGDB, somos obrigados a modificá-la para podermos utilizar outro. Para evitar estes incómodos, existem bibliotecas que servem de camada de abstração de SGDBs.

O que é uma camada de abstracção?

Uma camada de abstracção é simplesmente um elo de ligação entre uma aplicação e dados externos (sejam bases de dados, documentos de texto, etc). Ou seja, a aplicação não liga directamente aos dados externos, mas faz um pedido à camada de abstracção e esta, por sua vez, faz o pedido aos dados, independentemente do seu tipo. A classe ADOdb tem precisamente essa função, é precisamente igual escrevermos uma aplicação que use um determinado SGBD ou outro, o resultado será sempre igual e, as alterações mínimas.



Uso da class

Para utilizarmos esta classe, a primeira coisa que temos que fazer é o download da mesma, seguidamente descomprimi-la e guardá-la num directório acessível pelo servidor HTTP. Após a execução dos passos acima referidos devemos proceder à inclusão da class na nossa aplicação, da seguinte forma:

```
include "pasta/adodb.inc.php";
include "pasta/adodb-
exceptions.inc.php";
```

Depois, devemos criar a nova instância do objecto ADO (ActiveX Data Object) e fazer a conexão à Base de Dados pretendida, por exemplo (neste exemplo vou usar o conector de MySQL, mas poderia usar outro qualquer):

```
// Criação da nova instância do
objecto
$BD = NewADOConnection('mysql');
// Dados do servidor de MySQL
$servidor="localhost";
$utilizador="root";
$password="123456";
$nomebd="adodb_pap";
// Conexão à base de dados
$BD->Connect($servidor, $utilizador,
$password, $nomebd) or die("Erro ao
ligar à Base de dados!");
```

Após aberta a ligação já podemos inserir, alterar e remover dados. Para esse efeito vamos criar um recordset, que é um array com os dados consultados. Para criá-lo utilizamos o seguinte bloco de código:

```
$rs = $BD->Execute("select * from
alunos where codaluno=?",1);

// ou

$rs = $BD->Execute("select * from
alunos where nome='José' ")
```

Na primeira consulta, vamos ter todos os dados do aluno cujo código é 1, já na segunda vamos obter todos os dados de todos os alunos que se chamam José. Para imprimir todos os registos da consulta realizada temos que percorrê-la do início ao fim, com uma das seguintes formas:

```
//1ª Forma

while (!$rs->EOF)
{
    print_r($rs->fields);
    $rs->MoveNext();
}

//2ª Forma

while ($array = $rs->FetchRow())
{
    print_r($array);
}
```

A diferença entre os métodos é que o primeiro custuma ser usado nas linguagens criadas pela Microsoft (ASP) e o segundo método pelo php .

Esta classe permite-nos também o tratamento de excepções quando, por exemplo, fazemos uma consulta sobre uma tabela que não existe:

```
try
{
     $rs=$BD->Execute("select * from
alunox where codaluno=5");
}
catch (exception $e)
{
     print_r("<b>Erro</b>:".$e->msg);
}
```

O bloco de código supracitado tenta executar a consulta mas, se não o conseguir, vai dar o seguinte output: " Erro: Mensagem de Erro " .

Para verificar se o recordset contém algum registo existem várias formas, podemos utilizar:

```
$rs= $BD->Execute("Select * from
alunos where nota>20");

//1ª Forma

if($rs->RecordCount()>0)
   echo "Existem ".$rs-
>RecordCount()." registos que
correspondem à consulta!";
else
   echo "Não existem registos nesta
consulta!";

//2ª Forma

if($rs->EOF)
   echo "Não existem registos nesta
consulta!";
else
   echo "Existem ".$rs-
>RecordCount()." registos que
correspondem à consulta!";
```

No fim de realizarmos todas as tarefas na base de dados, é um bom hábito fechar a ligação, para não sobrecarregarmos o servidor:

```
$BD->Close();
```

Segurança

Quando estamos a programar para web, cada vez mais temos de ter em atenção a segurança. Para tal, convém evitar falhas de SQL Injection (ver edição anterior) e, mais uma vez, esta classe só facilita essa tarefa, tendo o método Quote($string) para fazer o "escape" de caracteres potencialmente perigosos. Suponhamos que temos uma página de consulta por Código de Aluno:


```
if($_POST['injeccao']) {
    echo "Dados recebidos por POST:".
$_POST['injeccao']

    is_numeric($_POST['injeccao']) ?
$codaluno = $_POST['injeccao'] :
$codaluno = $BD-
>Quote($_POST['injeccao'])

    echo "Dados tratados para a
consulta:". $codaluno;

    $rs = $BD->Execute("Select * from
alunos where codaluno=?", $codaluno);

    if($rs->EOF)
        echo "Erro: não foram
encontrados registos que
correspondessem à consulta realizada";

    while(!$rs->EOF) {
```

Dados recebidos por POST:1 OR 1=1 #
Dados tratados para a consulta:'1 OR 1=1 #'
Erro: Não foram encontrados registos que correspondessem à consulta realizada!

Caso permitissemos *SQL injection*, obteriamos este resultado:
José
António Carlos Manuel Silva

```
        print_r($rs->fields("Nome"));
        $rs->MoveNext();
    }

    echo "Caso permitissemos SQL
injection, obteriamos este
resultado:";

    $rs = $BD->Execute("Select * from
alunos where codaluno=?", $codaluno);

    while(!$rs->EOF) {
        print_r($rs->fields("Nome"));
        $rs->MoveNext();
    }
}
```

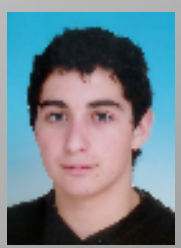
Bibliografia

- http://phplens.com/adodb/
- http://adodb.sourceforge.org

Conclusão

O objectivo deste artigo é dar a conhecer esta classe ainda pouco explorada. Devido à sua facilidade de uso e segurança torna-se indispensável conhecê-la minimamente. No entanto, não foram abordados todos os aspectos neste artigo e, por isso, recomenda-se que o leitor leia a documentação da classe nos links acima referidos.

SOBRE O AUTOR



João Alves é um jovem que dedica o seu tempo livre ao andebol e às novas tecnologias. Começou a programar com 12 anos e, hoje em dia, frequenta o Curso Tecnológico de Informática na Ancorensis, uma escola no Norte do país e, futuramente, tenciona ingressar em Engenharia Informática e de Computação na FEUP.

João Alves

Revista Linux

A Verdadeira Revista Portuguesa de Linux

Edição Bimestral em formato PDF

Download Gratuito

www.revista-linux.com

Programação Saudável

No dia-a-dia de um programador, são demasiadas as horas passadas à frente de um ecrã, a cansar os pulsos e os dedos, enquanto se está sentado, muitas vezes desconfortavelmente, numa cadeira pouco ergonómica. Daí, surgem primeiro as dores nas costas, que associamos à posição torta que assumimos durante as longas horas à frente do monitor. Depois, o ardor nos olhos de quem esteve 3, 4 ou 5 horas sem tirar os olhos das linhas de código só porque "tinha uma coisa para acabar". Por fim, desenvolvem-se várias patologias associadas intimamente à profissão e que vão destruir a carreira de qualquer programador caso não sejam prevenidas. Como se pode observar, programar é uma actividade de risco. Não de risco directo como um bombeiro ou um militar, mas não menos grave e com consequências menos dolorosas.

Porém, há esperança! Apesar de, mais tarde ou mais cedo, o programador vir a sofrer de uma ou outra "lesão" típica, há rotinas que se podem adoptar para diminuir a probabilidade das mesmas acontecerem. Ao contrário do que pode parecer, pausas frequentes tornam mais produtiva a actividade do programador. Contudo, não é saudável profissionalmente ser-se hipocondríaco e abusar destas pausas em prol da saúde! Mas adiante, passemos a assuntos concretos.

RSI, ou Lesão por Esforço Repetitivo, é a denominação geral dada a todas as doenças que advêm de uma actividade regular e repetitiva, nomeadamente lesões do túnel carpal, do túnel ulnar e tendinites, e que são caracterizadas clinicamente pelo afectar de músculos, tendões e nervos das mãos, braços e costas por uma tensão excessivamente longa devido quer a uma má postura, quer a movimentos repetitivos. Exemplos de actividades que levam ao seu aparecimento são: uso do computador (profissionalmente claro), tocar guitarra e escrever (intensamente, como é óbvio). Porém, nem todas estas doenças são exclusivamente causadas por actividades como programar. A genética pessoal tem um papel importante na determinação da probabilidade de eu, ou o leitor vir a desenvolver uma destas maleitas. Contudo, o passo mais importante a tomar quanto a estas doenças é a prevenção, uma vez que, quando desenvolvidas, torna-se complicado o tratamento, sendo por vezes ineficaz de todo.

O que se pode então fazer para prevenir o aparecimento da RSI? Primeiro, há que olhar para a secretária e para a cadeira e pensar no que vamos precisar de comprar. Cadeiras de madeira ou cadeiras rijas são catalisadores de dores de costas. Uma cadeira ergonómica, de altura e inclinação reguláveis, de preferência almofadada, é um óptimo investimento. A título de exemplo, quando comecei o meu estágio, sentei-me numa cadeira e andei semanas com dores nas costas. Passado uns tempos, troquei fortuitamente de cadeira e em menos de três dias recuperei das dores. Quanto à secretária, há que olhar para ver alguns parâmetros que, à partida, parecem insignificantes:

1. Altura do monitor: o monitor não deve estar abaixo nem acima do nível dos olhos. Devemos olhar directamente em frente. O facto de estarmos horas a fio de pescoço baixo ou esticado para cima a mirar o ecrã leva a dolorosos torcicolos. Colocar um livro que não usamos regularmente, ou mesmo até a caixa da motherboard ou da gráfica a servir de suporte ao monitor, poupa-nos o pescoço a esforços desnecessários.

2. Luminosade e contraste do monitor: quantas vezes não temos a luminosidade do monitor nos 100 ou nos 90, e o contraste alto, porque assim "vê-se melhor"? Claro que se vê, mas também somos bombardeados com muito mais intensidade pelo monitor. É o mesmo que estar a ler com uma lâmpada incandescente normal, ou com um holofote. Há que encontrar um equilíbrio entre o não estar muito claro e o não ver nada!

3. Rato e teclado: A altura e a distância a que estes periféricos se devem encontrar são fulcrais para o bem-estar dos nossos pulsos e braços. Não ter o teclado demasiado longe, evitando esticar os braços, é boa jogada. Quanto a altura, deve estar de modo aos músculos dos braços estarem relaxados, caso contrário, chegam ao fim do dia

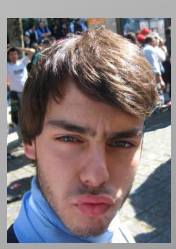
como quem teve a levantar pesos de 20Kg, e não, não ganham mais músculo por isso. Para o rato, as recomendações são as mesmas. Não estar nem muito longe, nem muito perto, de modo a que possamos movimentar a setinha no ecrã sem andar a fazer malabarismos com o pulso. Um tapete com almofada pode também ajudar a quem depende do rato como da água, uma vez que o esforço causado aos tendões do pulso pode ser excessivo.
Agora, alguns conselhos mais gerais. Faça pausas. Beba água. Venha à entrada do edifício regularmente apanhar ar.

Poderá perder um pouco da sua performance, mas o ganho em saúde vai compensar as vezes em que se senta ao monitor e já lhe dói tudo, esvaindo-se a vontade de trabalhar. E como é que se vai lembrar de fazer uma pausa quando está entusiasmado a teclar as últimas 100 linhas de código do programa, ou a fazer os últimos ajustes à imagem no Photoshop para o tal site? Fácil. Há a opção física e a virtual. A física passa por comprar um relógio com alarme ou então uma maçã-contador, como as usadas na cozinha. Regular o tempo para 30-45 min e quando tocar, fazer uma pausa de 10. Virtualmente, podemos instalar algum software no computador que nos avisa de quando devemos fazer a pausa (exemplo disso é o Workrave). Alguns softwares levam até ao bloqueio do rato e do teclado enquanto não passar o tempo da pausa, por isso, em vez de refilar com o rato não dar, mais vale levantar-se e ir arejar. Porém, o que fazer durante as pausas? Olhar para o tecto não parece boa opção. Comece por arranjar um sítio onde haja água disponível. Caso não haja, leve uma garrafa para o escritório. Beber água durante o dia de trabalho, no mínimo um litro e meio, serve de muito para ajudar a limpar o organismo de todos os detritos resultantes do metabolismo. Caso não aprecie gua, um chá serve igualmente. Agora, refrigerantes não ajudam em nada! Outra opção é andar. Vá dar um passeio até à entrada do edifício, apanhe ar, espreguice-se. Agora repita isso pelo menos cinco vezes em cada pausa. No secundário aprendeu que as veias funcionam por válvulas e que é preciso a contracção muscular para que essas válvulas funcionem na perfeição. Ora, a contracção muscular deve-se ao movimento (ou vice-versa) e por isso, sem movimento, acumulam-se fluidos nas pernas que levam, por exemplo, a tromboses (porque acha que no avião deve andar de um lado para o outro quando faz uma viagem longa?). Por fim, apanhar ar fresco e arejar a cabeça ajuda sempre à concentração.

Contudo, como tudo na vida, todas estas recomendações são genéricas e dependem de cada um para serem adoptadas e, como cada um é como é e há pessoas mais propensas a determinadas doenças que outras, o melhor cuidado que se pode ter é fazer um check-up regular no médico de família. Peça-lhe também conselhos para melhorar a sua postura no emprego e verá que, seguindo-os, tal como seguindo alguns destes, terá um aumento na produtividade, que, apesar de contraditório, visto dispender mais tempo fora do computador, tirará melhor proveito do que à frente dele passa.

Em última análise, se se sentir céptico em relação a todo este texto, só lhe resta uma opção: experimente. Mal não fará. Depois tire as suas próprias conclusões e partilhe com os seus colegas os resultados. Verá que mais tarde ou mais cedo, irá beneficiar de ter experimentado esta "tolice".

SOBRE O AUTOR



Estranhamente, João Rodrigues não vem das hostes dos informáticos tendo tido o primeiro contacto com a programação no 3º ano do curso. Um descontraído finalista da Licenciatura em Bioquímica em Coimbra, com particular interesse pelas áreas da Bioinformática e da Proteómica, entrou para a Revista PROGRAMAR e com o objectivo de mostrar que há mais na informática para além de bits e bytes.

João Rodrigues

Faça o seu media center com o GeexBox

Na 7ª edição desta revista, em Março de 2007, trouxemos até si um artigo que possivelmente adiou a compra de um novo computador: Ambientes gráficos em sistemas de fraco desempenho. No entanto, esse artigo apenas adiou o momento em que arrumou num canto da casa o seu fiel companheiro de tantos anos. Passado quase 1 ano, queremos levá-lo a procurar esse seu companheiro e levá-lo para a sua sala, usando-o como um media center.

De certeza que já ouviu falar de media centers e já os viu em lojas de electrodomésticos. Um media center é nada mais nada menos que um computador adaptado para ouvir música, ver filmes, TV, imagens e algumas coisas mais. Ou seja, um media center é na sua essência igual ao seu computador de secretária ou portátil, sendo possível usar qualquer computador dos referidos como media center.

Por isso, vamos pôr as mãos à obra e tornar o PC do século passado num media center de meter inveja! Para tal, vamos recorrer a uma distribuição GNU/Linux, chamada GeexBox.

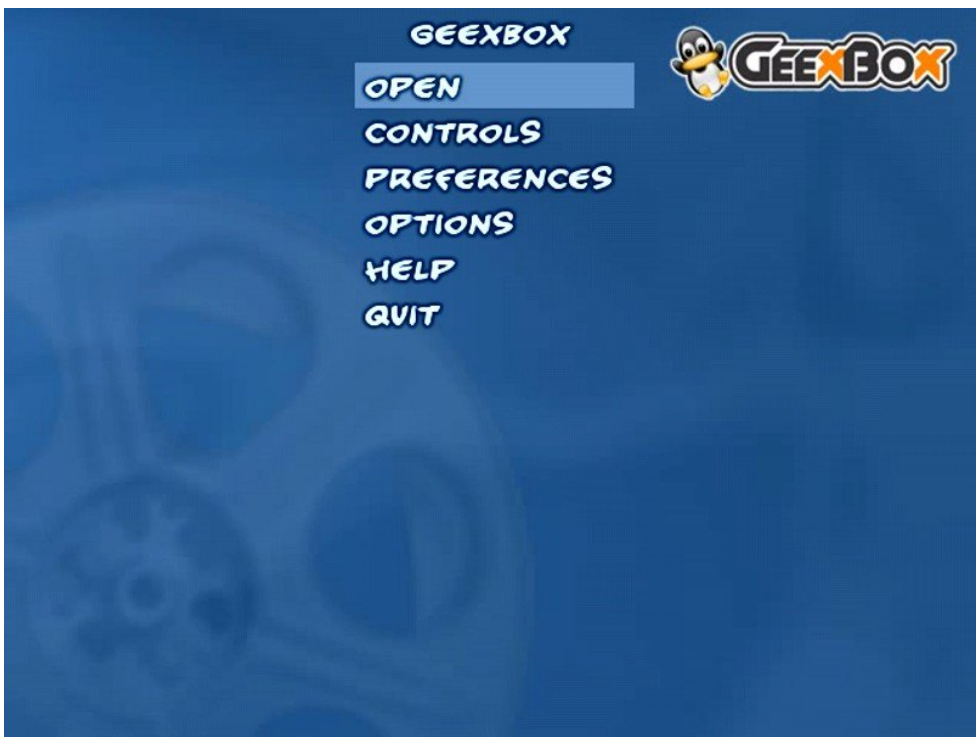
GeexBox

O Geexbox suporta placas de captura de TV/rádio e comandos por infravermelhos, que usa o MPlayer para reproduzir ÿlmes e
música. Esta distribuição é bastante personalizável, sendo possível fazê-lo usando um gerador de ISOs bastante simples de usar, tanto em Windows, Linux ou Mac.

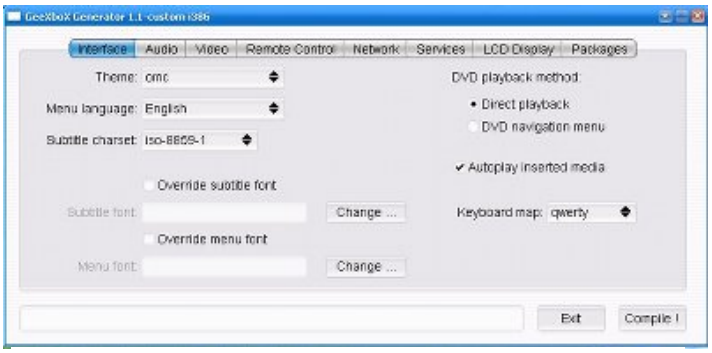
Usando o GeexBox, podemos ver DVDs e HD-DVDs, filmes em vários formatos como DivX, XviD, H.264, MPEG ou WMV, com ou sem legendas em .srt, ouvir música em AAC, FLAC, MP3 ou WMA, assistir TV ou ouvir rádio pela Internet ou usando uma placa de captura de TV/rádio. Com o teclado ou um comando, podemos também definir delays entre o vídeo, o som e as legendas, controlar a velocidade de reprodução, entre outras coisas.

Gerar um ISO personalizado

Um dos grandes trunfos do GeexBox é o seu gerador de ISOs personalizados. Usando-o podemos configurar as características da rede a que nos vamos ligar no GeexBox, seja por Ethernet ou por WiFi, que pacotes ou drivers queremos incluir, se queremos correr um servidor de FTP para acedermos ao disco da máquina remotamente ou instalar a interface web. O download do gerador está



Ecrã inicial do GeexBox



Gerador de ISOs

disponível na seguinte página:

http://geexbox.org/en/generator.html

No gerador, poderá configurar o seu sistema de som (se é 5.1, 2.0, ...) na aba Audio. Se tiver telecomando deverá escolher o modelo apropriado na aba Remote Control. Na aba Network, poderá configurar manualmente a sua ligação à rede, seja por Ethernet ou WiFi.

Uma coisa que me chamou bastante à atenção pela primeira vez que usei este configurador, foi o facto de podermos activar um web server na aba Services que nos permite controlar a nossa máquina com o GeexBox a partir de um browser. Embora esta interface web esteja pouco desenvolvida, a nós, programadores, abre-nos as portas para expandirmos o seu potencial

Sobre o gerador, pouco mais tenho a dizer para além de que aconselho seleccionar o codec Windows Media 9 na aba Packages, que o Keyboard map é o QWERTY, que o charset das legendas é, regra geral, ISO-8859-15 e que em relação às Windows Shares, nada deverá ser necessário mudar.

O que fazer depois de gerar o ISO

Depois de gerar o ISO, tudo o que tem de fazer é gravá-lo para um CD para bootar o GeexBox como LiveCD. Depois de

o LiveCD bootar, terá um media center pronto a testar!
Na opção Controls, é apresentada uma pequena lista com as teclas de controlo do GeexBox. Sabendo isto, tudo o que tem de fazer é ir à opção Open e abrir um dos seus filmes. Deixo aqui uma imagem de um filme com legendas externas num ficheiro .srt a ser reproduzido no GeexBox.

Adorei o GeexBox, mas é chato estar sempre a pôr e a tirar o CD...

Não se preocupe, aquando do boot do GeexBox, há um ecrã onde pode especificar as opções de boot. Nesse ecrã, tudo o que tem de fazer é escrever install e seguir o instalador. A única parte menos óbvia da instalação é o particionamento do disco, onde basta criar uma partição a ocupar todo o disco rígido e formatá-la como ext2 ou ext3.

Para instruções mais detalhadas, veja a documentação oficial aqui: http://www.geexbox.org/wiki/index.php/Installation

Ler mais:
http://geexbox.org/
http://geexbox.org/en/generator.html
http://geexbox.org/wiki/index.php
http://www.portugal-a-proogramar.org/forum/index.php/topic,4600.0.html

SOBRE O AUTOR

David Ferreira é um jovem apaixonado pelo mundo da informática. Deu os seus primeiros passos na programação aos 12 anos através da linguagem PHP, tendo hoje conhecimentos de XHTML, JavaScript, Python, redes, e outros. É um entusiasmado pelo mundo do hardware e software open-source, que o levou a explorar os sistemas GNU/Linux, com a distribuição Kurumin e depois Ubuntu.

David Ferreira

Hi5 Developer Center

O Hi5 nunca disponibilizou quaisquer meios para 3rd party developers desenvolverem aplicações que usassem o hi5, obrigando-os a "manobras" mais feias para o poder fazer. No entanto, isso está a mudar com o uso de microformats e disponibilização de uma API SOAP completa, alguns feeds e REST end-points.

http://www.hi5networks.com/developer/

Google Code

O Google Code é nada mais nada menos que a homepage indicada para qualquer developer. Tem um repositório de projectos desenvolvidos pelo Google e oferece alojamento a projectos open-source. Para além disso o developer que pretender alojar o seu projecto no Google Code tem também direito a um wiki para o mesmo, assim como uma ferramenta de Bug Reports e um sistema de controle de versões, tudo ferramentas que podem sempre fazer falta a qualquer projecto.

http://code.google.com/

Twitter

O twitter é uma rede social, assim como um servidor para microblogging. É possível enviar actualizações através de SMS, email ou IM. Estas mensagens ficarão disponíveis na página do nosso perfil, assim como na página de quem tem o nosso perfil assinado. Depois do envio de um SMS (pago) para confirmação de número, é possível receber actualizações dos perÿs que temos assinados através de SMS a custo zero..

http://www.twitter.com/

CodePlex

O CodePlex é um site da Microsoft que oferece alojamento grátis a projectos open-source. Uma atitude de louvar, e mais um site a concorrer com o Google Code e com a SourceForge.net

http://www.codeplex.com/

A vantagem dos programas de IM

Até o compilador se queixa...

Perigos da blogosfera

Mini Radio Player

O Mini Radio Player é um programa que reúne numa base de dados várias Web streams de televisão e de rádio com uma interface agradável e com variadas opções de visualização e personalização.

O programa encontra-se neste momento na versão 2, que introduziu uma nova organização e nova informação para os canais (RTDB2), um novo visual, possibilidade de usar as Media Keys do teclado, suporte para outros Idiomas e pesquisa de canais simples.

As funções principais são as seguintes:

RTDB2

Para além do nome do canal e do endereço de streaming, esta função pode englobar o site oficial do canal, o logótipo, o país de origem, uma descrição do canal e a função de Canal QPassa! (este permite ver que programação está a decorrer neste momento)

Favoritos

Permite guardar os seus canais favoritos numa secção à parte.

Conteúdo

Permite "bloquear" qualquer canal ou categoria ou país por uma palavra-chave ou por uma mensagem de aviso.

RapidChan!

Uma função prática que permite escolher o canal quando o programa se encontra minimizado no system tray, sem ter de o maximizar.

Media Player Systray Controls

Permite colocar os controlos play, stop, sem som, mais som, menos som no system tray quando o programa se encontra minimizado.

Para além destas destacam-se ainda: Mini TV, Detached Screen, Auto Updates, pesquisa simples, janela de Informação Extra e opção 'O que estou a ouvir/ver' para o Windows Live Messenger.

No futuro serão (re)adicionadas as funções de temas, MultiChannel, AutoTasks e Fullscreen Mode.

http://miniradioplayer.no.sapo.pt/

Queres participar na Revista PROGRAMAR? Queres integrar este projecto, escrever artigos e ajudar a tornar esta revista num marco da programação nacional?

Vai a

www.revista-programar.info

para mais informações em como participar,
ou então contacta-nos por

revistaprogramar
@portugal-a-programar.org

Precisamos do apoio de todos para tornar este projecto ainda maior...

contamos com a tua ajuda!

Equipa PROGRAMAR

Um projecto Portugal-a-Programar.org