PROGRAMAR

A REVISTA PORTUGUESA DE PROGRAMAÇÃO

Revista nº10 - Setembro de 2007 www.portugal-a-programar.org

INICIAÇÃO AO

ASSEMBLY x86

Aprenda os aspectos teóricos necessários para entender completamente o paradigma da programação nesta linguagem.

Projectos em MONO

Como criar aplicações em multiplataforma mesmo usando .Net

IPSEC

Saiba mais sobre este protocolo de segurança IP

e ainda...

PROGRAMAÇÃO COM GRAFOS

A resolução simples de muitos problemas complexos

índice

3 notícias
4 opinião
6 tema de capa
11 a programar
23 segurança
27 tutorial
31 gnu/linux
35 internet
36 bluescreen
37 comunidade
38 especial

equipa PROGRAMAR

administração
Rui Maia
David Pintassilgo

coordenador
Miguel Pais

coordenador adjunto
Joel Ramos

editor
Pedro Abreu

redacção
Sérgio Matias
João Matos
Sandro Pinto
Miguel Araújo
Fábio Correia
Pedro Teixeira
Filipe Jorge
Andreia Gaita

colaboradores
David Ferreira
Daniel Correia
José Oliveira
Sérgio Santos

contacto
revistaprogramar
@portugal-a-programar.org

website
www.revista-programar.info

A (des)informação

Quando um projecto web2.0 surge, e parte de utilizadores individuais e anónimos para que sejam eles os próprios agentes de produção de informação, obviamente nas áreas em que para tal se sentem mais à vontade, todos estranham e se inquirem como poderão os autores do projecto ter a ilusão que tal alguma vez resultará ou, impossibilidade absoluta, vir ainda a ser creditado como fonte de informação crível. Foram estas e muitas outras as incertezas que infernizaram, decerto, o começo de um dos primeiros, e agora mais bem-sucedidos, projectos da web.

Quase impossível era a tarefa, mas como não seria de esperar, os anónimos do mundo têm uma palavra a dizer, e tal como muitos outros projectos sociais que agora afloram por essa net fora e chateiam por não surpreenderem, a informação que antes seria repugnante ler, não estivéssemos a tomar como certa uma barbaridade escrita por alguém da cochichina, vira um dos mais importantes projectos da web, considerado agora fonte de informação quase tão crível como muitas das melhores enciclopédias do mercado, livremente fornecendo o que outrora muito se pagava para adquirir.

Nesta altura, a do sucesso, alguma reflexão é necessária. Alguém ainda se lembra quando torcia o nariz à credibilidade da informação online? Para onde foram os argumentos que sustentavam que nada ultrapassaria uma boa enciclopédia de centenas ou milhares de euros? Quem ainda se lembra que afinal, e se nos bem entender, uma página da tal enciclopédia que tem agora razão indiscutível no que diz, poderá muito bem ser editada a nosso proveito? Não é isso que o projecto pede e incentiva? A parte da credibilidade foi o grande problema no começo, mas é preciso manter em mente que esse problema não se ultrapassa, é constante. Cabe-nos a nós não tomarmos como certo o que pode estar intencionalmente errado. Atitudes deploráveis, é certo, de quem tira proveito de tal instância benévola para enganar tudo e todos, ou suplantar novas verdades, mas não me cabe a mim discutir agora esses assuntos.

Deploráveis são também as atitudes daqueles que, tendo acesso livre a informação, deixam de usar o lobo cerebral que produziria um trabalho genuíno, e que acima de tudo revelaria bom-senso, decidindo apresentar como deles informação que não é. Digamos que fica bem, claro... mas não por muito tempo.

I Fórum de Software Livre de Lisboa

Este fórum que conta com o apoio da FCCN, Google, SUN, PHP-Magazine, ANSOL, ASL-Angola, DRI, Vertical One e outras empresas do sector realizar-se-á nos dias 12 e 13 de Outubro na FCUL (Faculdade de Ciências da Universidade de Lisboa). O I FSL visa aprofundar a discussão sobre o uso e desenvolvimento de ferramentas de código aberto, não só como forma de aprofundar o conhecimento científico, mas também como ferramenta de trabalho e modelo de negócios.

O fórum é subordinado ao tema do software livre ou open source, assim conhecido pela liberdade que oferece aos profissionais, académicos e outros de estudar o seu funcionamento. Nos tempos que correm, muitas empresas mundiais geram ferramentas utilizando código fonte aberto, acabando por valorizar mais os profissionais e dar atenção aos serviços e não apenas a produtos ou licenças de software.

O modelo de desenvolvimento partilhado tem-se mostrado uma forma rápida e eficaz de corrigir "bugs", uma vez que ao lançarmos um software na Internet com acesso livre, rapidamente recebemos sugestões de correcções e melhorias, seja em relação à funcionalidade, segurança ou outro aspecto. Muitos países, como a Alemanha e o Brasil, já adoptaram software livre como política de redução de custos e valorização tecnológica interna, fortalecendo o mercado empresarial.

Ralf Braga, da coordenação geral do evento, acredita que o país deve investir em formação, devido à falta de mão-de-obra de profissionais em software livre. O sistema operativo GNU/Linux é o mais conhecido e divulgado mundialmente, fazendo muitos esquecerem-se de mencionar outros, como o OpenBSD, reconhecido como o sistema Unix-based mais seguro, não sendo identificadas falhas de segurança há mais de 9 anos, o FreeBSD, entre outros. Muitos destes softwares serão abordados no evento, assim como representantes das comunidades e empresas que prestam apoio aos programas. O software livre tem-se mostrado uma óptima opção para vários modelos de negócio, sendo utilizado em diversos casos de sucesso na Europa e no Mundo.

Venham partilhar experiências neste 1º Fórum. Ensinar, aprender, ouvir, falar, opinar... A vossa presença é fundamental! O lançamento oficial do evento aconteceu no passado dia 31 de Julho no site do evento: http://www.softwarelivre.com.pt

Comunidade Portuguesa de Wordpress

Com um mês de existência, a Comunidade Portuguesa de Wordpress veio colmatar um vazio existente em Portugal, uma vez que ainda não existia uma comunidade deste tipo. O número de utilizadores de Wordpress em Portugal é cada vez maior e podem agora contar com um local para partilhar experiências, tirar dúvidas e ajudar outros utilizadores em português.

Apesar da recente criação, esta comunidade conta já com um fórum (http://forum.wordpress-pt.com) e um blog (http://www.wordpress-pt.com) onde serão colocados os vários anúncios, as novidades e também ficheiros para download.

Ainda durante o mês de Agosto foi lançada pela equipa desta comunidade a versão do Wordpress 2.2.2, bem como a tradução de plugins, que resultaram da contribuição de alguns membros da comunidade, estando prevista a tradução de futuros lançamentos.

Portugal aprova standard OOXML

Como estava definido, a Comissão Técnica criada pelo Instituto de Informática para avaliação do processo de normalização do Open XML (OOXML) reuniu ontem para a votação final, decidindo a favor da aprovação da norma com 13 votos a favor e 7 contra. O processo, que esteve envolto em polémica desde o início, fica assim decidido a favor da nova norma utilizada nos documentos do Office.

Este não foi um processo pacífico, movimentando petições a nível nacional e internacional e tendo sido amplamente criticado em blogs e vários artigos de opinião por diversas razões, entre as quais se contam o facto da comunidade open source não considerar o OOXML um standard aberto e porque já existe um para documentos estruturados (o ODF proposto pela OASIS). Somam-se a estes argumentos o facto de terem ficado de fora da comissão técnica portuguesa duas empresas que defendem o formato ODF (a IBM e a Sun) e de o presidente eleito pela comissão ser funcionário da Microsoft Portugal.

Resta agora aguardar pela decisão final do ISO, que irá considerar os votos de outros países.

O Preço do Código

Quando se desenvolve uma actividade por conta própria ou se está a iniciar no mercado existem sempre dificuldades em orçamentar correctamente os produtos e serviços.

Pretendo com este artigo tentar esclarecer algumas dúvidas e tentar ajudar nesta quase sempre complicada tarefa.

Uma das melhores formas de orçamentar um serviço é definir um valor hora e multiplicar pelo número de horas previstas de execução.
Este valor deve ter em conta os seguintes factores de influência:

• A dificuldade e o conhecimento que se tem da linguagem;

• A experiência em termos de trabalhos do mesmo género;

• O tempo disponível para o desenvolvimento.

A somar a estes valores estão também:

• Deslocações (quando necessárias);

• Comunicações (telemóvel, correio, faxes e internet);

• Custos com o software de desenvolvimento;

• Custos de sub-contratação;

• Imprevistos (acreditem que acontecem mais do que seria desejável);

• E, claro, Impostos.

Outra forma de orçamentar pode passar pela definição de um preçário por tarefas. As tarefas regem-se pelas mesmas variantes, mas podem facilitar alguns tipos de orçamentação para trabalhos mais regulares.

Imagem por =O-range

Pode aplicar-se, por exemplo, em casos como desenvolvimento de designs, instalação de CMS, entre outros.

Assim sendo, por exemplo, no caso de um trabalho que consista na criação de um backoffice para um site pode-se orçamentar das seguintes maneiras:

Desenvolvimento de Web site com backoffice para gestão de conteúdos:
40h x 15€ = 600€

ou

Desenvolvimento de Web site com backoffice para gestão de conteúdos

• Design Personalizado: 150€
• Criação de Backoffice: 400€

Mas nem só de preços se faz um orçamento. Por vezes existem problemas com clientes que à partida não definiram correctamente o que desejavam, criando uma situação de deficiente orçamentação, ou uma má apresentação da mesma. Assim sendo, um orçamento também deve incluir:

• Especificação concreta e exaustiva dos serviços a prestar; no caso do software devem indicar todas as funcionalidades (funcionalidades adicionais são frequentemente um problema);

• Indicação de que todos os serviços não indicados deverão ser orçamentados à parte;

• Prazos de entrega, com as reservas para correcção de erros e atrasos da parte do cliente;

• Indicação de quem tem os direitos de autor sobre o resultado;

• Condições de Pagamento;

• Validade do orçamento - apesar de não parecer, revela-se útil em casos em que é necessário acelerar a decisão do cliente, ou evitar juntar muitos trabalhos ao mesmo tempo.

• Estes elementos visam evitar problemas de sub orçamentação que tornam o trabalho difícil e muitas das vezes pouco rentável, além de evitar que alguns clientes menos éticos se aproveitem de uma sub orçamentação ou uma orçamentação menos clara.

• Devem-se evitar orçamentos inflacionados só porque se acredita que o cliente pode pagar mais, uma vez que cada cliente é uma carta fechada, e além disso pode dificultar uma boa publicidade por parte do cliente.

• Se o leitor anda por estas andanças há pouco tempo, aconselho-o vivamente a criar um portfólio web ou em papel para poder mostrar os seus trabalhos e assim melhor justificar os valores pedidos e ter maior probabilidade de aprovação do orçamento.

Para finalizar recordo que estas dicas apenas tentam ajudar a orçamentar, e cabe a cada um o bom senso de definir o valor do seu trabalho. Aborrece-me ver pessoas que muitas vezes dificultam o trabalho de quem vive dos serviços que presta, criando uma ideia errada aos clientes de um baixo preço de alguns serviços, quando muitos sabem as dificuldades de quem legalmente desenvolve.

SOBRE O AUTOR

Tendo realizado o Curso Técnico de Informática III no Centro de Formação Profissional de Santarém e tendo-o terminado com a nota de 19 valores, Sérgio Matias ingressou depois numa empresa de Aplicações Multimédia, tendo-se lançado em 2006 como freelancer nas áreas de web design e programação.
Website: http://www.sergiomatias.net

Sérgio Matias

Iniciação ao Assembly x86: Aspectos teóricos

Introdução

Este tutorial pretende ensinar os procedimentos básicos de programação em linguagem Assembly para processadores x86 em ambientes GNU/Linux.

Para quem não está familiarizado, GNU/Linux é um sistema operativo modelado no UNIX. A parte GNU refere-se ao projecto GNU (GNU's Not Unix, http://www.gnu.org/), iniciado em 1983 por Richard Stallman, com o objectivo de criar um sistema operativo livre. Em 1991/1992, o projecto GNU já tinha desenvolvido a maior parte das aplicações essenciais para criar um sistema operativo livre, faltando o kernel (núcleo do sistema). Neste momento surge o Linux, um kernel baseado na arquitectura UNIX, desenvolvido por Linus Torvalds, um estudante finlandês. Com a integração dos dois projectos, surge o GNU/Linux, um sistema operativo livre e de código fonte aberto.

O kernel é o componente principal de um sistema operativo, responsável por gerir os recursos do computador e a comunicação entre o hardware e o software. Também funciona como uma camada de abstracção para os componentes/periféricos do computador (por exemplo: a memória, o processador e os dispositivos de I/O). Geralmente o sistema operativo disponibiliza estes recursos através de mecanismos de comunicação entre processos e chamadas de sistema (system calls).

No que toca às linguagens de programação, podemos considerar três categorias:

- 1. Código máquina
- 2. Linguagens de baixo nível
- 3. Linguagens de alto nível

A linguagem Assembly é uma linguagem de baixo nível constituída por um conjunto de mnemónicas e abreviações. Em comparação com código máquina (uma série de números em formato binário), Assembly torna as instruções mais fáceis de lembrar facilitando a vida ao programador.

O uso da linguagem Assembly já data da década de 1950, sendo nessa altura uma linguagem bastante popular. Actualmente, com a evolução das linguagens de alto nível, é usada maioritariamente no desenvolvimento de drivers, sistemas integrados e na área de "reverse engineering" (como a maior parte dos programas só estão disponíveis num executável binário ou código máquina, é muito mais fácil traduzi-los para linguagem Assembly do que para linguagens de alto nível - este processo designa-se por disassembly).

O código fonte de um programa em linguagem Assembly está directamente relacionado com a arquitectura específica do processador alvo - ao contrário das linguagens de alto nível, que são geralmente independentes da plataforma, bastando recompilar o código para o executar numa arquitectura diferente.

A linguagem Assembly é traduzida para código máquina através de um programa chamado assembler. Um assembler é diferente de um compilador na medida em que traduz as mnemónicas uma-a-uma para instruções em código máquina, enquanto um compilador traduz as instruções por blocos de código.

Antes de executar o código máquina gerado pelo assembler, temos de fazer a "linkagem" do executável. Este processo é realizado pelo linker, que basicamente substitui símbolos presentes no código do programa pelos locais concretos onde esses residem. Imaginem que é chamada uma função no código: o linker substitui essa referência pelo local em algum ficheiro onde o código da função se encontra (exemplo: função getline -> "módulo iosys - 123 bytes a partir do início").

Apresentados alguns pormenores desta linguagem, passamos à instalação das ferramentas necessárias:

1. Assembler

Existem muitos assemblers disponíveis, destacam-se: GAS (GNU Assembler - http://www.gnu.org/software/binutils/), TASM (Borland Turbo Assembler - http://en.wikipedia.org/wiki/Turbo_Assembler), MASM (Microsoft Macro Assembler - http://masm32.com/), FASM (Flat Assembler - http://flatassembler.net/) e NASM (Netwide Assembler - http://nasm.sourceforge.net/). Este último é multi-plataforma e o código fonte está disponível gratuitamente. O código fonte apresentado neste tutorial

foi desenvolvido para o NASM, logo recomendo que o usem. Atenção que a sintaxe pode ser diferente entre assemblers (existem 2 tipos genéricos de sintaxe: AT&T e Intel), logo um código para um determinado assembler pode não funcionar noutro.

2. Linker

No que toca a linkers, não existem tantas opções como na categoria dos assemblers. O linker que vai ser usado é o ld, que vem com o pacote binutils do projecto GNU (http://www.gnu.org/software/binutils/) . Outra alternativa é o alink (http://alink.sourceforge.net/).

3. Editor

Podem usar qualquer editor de texto. As escolhas mais populares em ambientes GNU/Linux são o vi/vim, emacs, e pico/ed/nano. Caso não se sintam à vontade a editar o código na consola (shell), também podem usar um editor de texto com interface gráfica, como o gedit, Geany, etc.

Caso o vosso sistema não tenha os pacotes instalados, procurem na documentação da vossa distribuição como o fazer.

Arquitectura do computador

Antes de começarmos a programar em Assembly, temos de aprender os conceitos básicos do funcionamento interno de um computador.

A arquitectura dos computadores modernos é baseada na arquitectura Von Neumann, seguindo o nome do seu criador. Esta arquitectura divide o computador em duas partes principais: o processador (CPU - Central Processing Unit) e a memória. Esta arquitectura é usada em todos os computadores modernos, incluindo os computadores pessoais, super computadores, mainframes, consolas de jogos e até mesmo telemóveis.

Estrutura da memória do computador

A memória do computador é o espaço onde estão armazenados todos os dados do computador. Este espaço tem um tamanho fixo e os dados podem ser acedidos através de endereços. Por exemplo, imaginem que têm 128MB de RAM no computador. Isto corresponde a 131072 kilobytes, ou 134217728 bytes. Neste caso, estão disponíveis 134217728 posições de armazenamento diferentes do tamanho de um byte. Não esquecer que o computador começa a contar no 0, logo os endereços de memória disponíveis neste caso começam no 0 e acabam em 134217727.

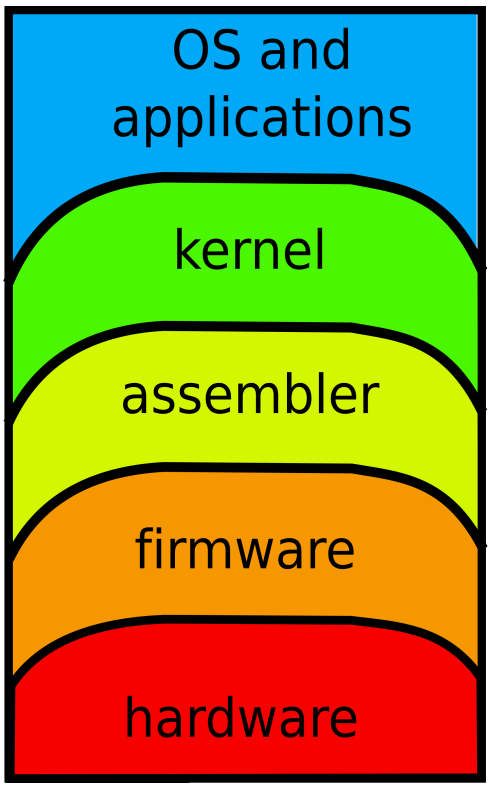
Processador

O processador é o componente do computador que interpreta e executa as instruções dos programas e processa os dados. Este é constituído por vários sub-sistemas, dos quais se destacam a ALU (Arithmetic Logic Unit) - responsável por todas as operações aritméticas (ex. adição e subtracção) e lógicas (ex. AND, XOR, OR); FPU (Floating Point Unit) - equivalente ao ALU mas para números decimais; os registos - zona de armazenamento ultra-rápida, utilizada pelo processador para acelerar a execução dos programas permitindo acesso aos valores utilizados mais frequentemente.

Existe um número limitado de operações, sendo o conjunto de todas essas operações e das suas variações designado por ISA (Instruction Set Architecture). Existem diferentes conjuntos de instruções mas consideram-se duas categorias: RISC (Reduced Instruction Set Architecture - ex. arquitectura MIPS) e CISC (Complex Instruction Set Architecture - ex. arquitectura x86).

Este tutorial vai abordar o conjunto de instruções base x86 (este surgiu pela primeira vez em 1978 no processador Intel 8086). Ao longo dos anos têm sido feitas extensões a este conjunto de instruções, tais como o MMX, 3DNow!, SSE, SSE2 e SSE3.

Todos os processadores com base na arquitectura Von Neumann funcionam com base num ciclo constituído por 3 passos essenciais: fetch, decode, execute.



No primeiro passo o processador obtém a próxima instrução a executar a partir da posição contida no registo PC, que armazena a posição actual da memória do programa; no segundo passo, o processador divide a instrução (em código máquina) em secções: uma com o opcode da operação a executar (Operation Code) e as outras com dados complementares para realizar a operação; no terceiro passo a operação é executada.

Outro componente do processador são os registos. Os registos são como variáveis ultra-rápidas embutidas no processador. É nos registos que são armazenados todos os dados necessários para efectuar os cálculos ou outras operações realizadas pelo processador.

O problema da família x86 de processadores é que existem poucos registos disponíveis, o que leva a que o programador tenha de gerir bem os registos necessários para a sua aplicação.

Os registos são normalmente de 32bits, mas também existem registos de 16bits e de 8bits. Por exemplo, tomando como base o registo AX de 16bits, podemos considerar o registo EAX (Extended AX) de 32bits, e os registos AH (AX Higher) e AL (AX Lower) que correspondem a dois registos de 8bits.

Registos gerais:
Tal como o nome indica estes são os registos usados a maior parte do tempo. A maioria das instruções tem como base estes registos.

• EAX, AX, AH, AL: "Accumulator register"
Normalmente usado para acesso I/O, aritmética, chamadas de sistema, etc...

• EBX, BX, BH, BL: "Base register"
É usado como um ponteiro base para o acesso à memória.

•ECX, CX, CH, CL: "Counter register"
É usado como contador de loops e para shifts.

• EDX, DX, DH, DL: "Data register"
Semelhante ao registo EAX.

Registos de segmento

• CS: "Code segment"
Armazena o segmento de código do programa.

• DS: "Data segment"
Armazena o segmento de dados do programa.

• ES FS GS:
Registos de segmentos adicionais para armazenamento de segmentos.

• SS: "Stack segment"
Armazena o segmento da stack do programa

Index e ponteiros

• EDI: "Destination index register"
Usado para cópia de strings e arrays de memória e para endereçamento de ponteiros em conjunto com o ESI.

• ESI: "Source index register"
Usado para cópia de strings e arrays de memória.

• EBP: "Stack Base pointer register"
Armazena o endereço da base da stack.

• ESP: "Stack pointer register"
Armazena o endereço do topo da stack.

• EIP: "Index Pointer"
Armazena o offset para a próxima instrução.

• Indicador
EFLAGS: Armazena o estado do processador

Modos de endereçamento de memória

1. Endereçamento por valor imediato (immediate address mode)
2. Endereçamento de registo (register address mode)
3. Endereçamento directo (direct addressing mode)
4. Endereçamento por index (indexed addressing mode)
5. Endereçamento indirecto (indirect adressing mode)
6. Endereçamento por ponteiro base (base pointer addressing mode)

No primeiro caso, atribuímos o valor directamente. Por exemplo, se quisermos inicializar um registo para 0, introduzimos directamente o valor 0, em vez de darmos um endereço para o processador ler o valor 0.

No modo de endereçamento de registo, a instrução contém o registo de onde deve obter o valor, em vez de uma localização na memória.

No modo de endereçamento directo, a instrução contém o endereço da memória que contém o valor. Por exemplo, podemos pedir ao processador para copiar um valor num determinado endereço da memória para o registo do processador.

No modo de endereçamento por index, a instrução contém um endereço de memória para aceder e um index, que funciona como um offset. Por exemplo, se utilizarmos o endereço 1390 e um index de 10, o valor lido vai ser o da

localização 1400. Nos processadores de arquitectura x86, ainda podemos especificar um multiplicador para o index. Isto permite aceder blocos de um determinado tamanho.

No modo de endereçamento indirecto, a instrução contém um registo que por sua vez contém um ponteiro para um endereço da memória onde a data deve ser obtida. Por exemplo, imaginemos que o registo eax está populado com o valor 10. Se estivéssemos a usar este modo de endereçamento indirecto, e pedíssemos o valor indirecto do registo eax, obteríamos o valor que estivesse na posição 10 da memória.

Finalmente, o modo de endereçamento por ponteiro base funciona de forma semelhante ao modo de endereçamento indirecto, mas é permitido especificar um index tal como no modo de endereçamento por index.

Chamadas ao sistema (system calls)

Quase todos os programas precisam de lidar com vários operações de entrada e saída de dados, controlo de pastas e ficheiros, obter detalhes do sistema, ou seja, interagir com o sistema operativo chamando as suas APIs (Application Programming Interface). Essas operações são efectuadas com recurso ao kernel, usando um mecanismo de chamadas ao sistema (system calls), através de um processo designado por interrupção.

Basicamente, quando o processador encontra uma instrução de interrupção, faz uma chamada ao kernel que executa a operação pedida. Acabada a operação, o kernel volta a ceder o controlo do processador ao programa, retornando ainda um código, possibilitando ao programa saber informação sobre o resultado da operação (exemplo: se um directório foi criado com sucesso, se os dados foram escritos correctamente num determinado ficheiro, etc...).

Este processo é efectuado com a instrução "int", estando o número do serviço no registo eax do processador. Dependendo de cada chamada, são necessários outros dados presentes noutros registos do processador, por exemplo, na chamada de saída (exit), que permite ao programa acabar a sua execução, o código de retorno para a consola é obtido no registo ebx.

Primeiro programa

Para começar vamos criar um programa que apenas retorna o código de saída para a consola de execução, de forma a demonstrar como se executam as chamadas ao sistema.

```
section .text      ; inicio da seccao de
texto
    global _start ; onde deve comecar a
execucao

_start:            ; label start - a
execucao comeca aqui
    mov eax, 1     ; move o valor 1 para
o registo eax
    mov ebx, 0     ; move o valor 0 para
o registo ebx
    int 0x80       ; chamada de sistema
para o kernel
```

Assembling e linking

O comando para assemblar o ficheiro de código fonte num ficheiro objecto é o seguinte: `nasm -f elf <codigo.asm>` .Se forem detectados alguns erros durante o processo, o NASM fará o output para consola dos erros e das linhas onde ocorreram.

O próximo passo é a linkagem, que pode ser feita com o seguinte comando: `ld -s -o <codigo> <codigo.o>`.Por fim executem o programa: `./<ficheiro>` . O programa deve ter terminado sem qualquer erro, para verem o código de saída com que o programa retornou: `echo $?`.

Nota: Por norma, usa-se a extensão ***.asm*** para código fonte em Assembly.

Agora que executámos o nosso primeiro programa em Assembly, vamos dissecá-lo e perceber como funciona.

Na linguagem Assembly, cada instrução está associada a uma linha distinta. A primeira linha do nosso programa inicia a secção de texto ("section .text").

Em Assembly podemos considerar três secções lógicas que dividem um programa: "text", onde se encontram as instruções que vão ser executadas pelo processador; "data", onde definimos constantes, como nomes de ficheiros e buffers - esta data não é modificada durante a execução; e "bbs", onde declaramos as variáveis e reservamos memória para os dados e estruturas que sejam precisos durante a execução do programa.

Seguidamente a instrução "global _start" diz ao assembler que o ponto de início de execução do programa é uma label chamada _start. Por convenção, usa-se "_start" em todos os programas desenvolvidos no ambiente GNU/Linux.

Na linha seguinte, é declarada uma label, de nome _start. Uma label é um conjunto de instruções. Quando se chama uma label para execução, as intruções desta são executadas sequencialmente pela ordem que aparecem no código.

Neste caso, a primeira instrução a ser realizada é a `mov eax, 1`. O que esta execução faz é mover o valor 1 para o registo eax do processador. Em todas as operações da sintaxe Intel, o primeiro operando corresponde ao local de destino, e o segundo ao valor inicial.

Nota: Na sintaxe AT&T, a ordem dos operandos é inversa.

Nas linhas seguintes movemos o valor o para o registo ebx do processador, e fazemos uma chamada ao sistema com a instrução int 0x80 (abreviatura de interrupt).

Como estamos a chamar o serviço exit do sistema (valor 1 no registo eax), o programa retorna à consola com o valor no registo ebx. Se experimentarem alterar este valor no código fonte, e voltarem a correr o programa, podem ver que o valor que o programa retorna para a consola é diferente.
Nota: Não utilizar valores superiores a 255 (o valor máximo de um unsigned byte) ou podem ocorrer problemas de overflow. Ao ultrapassar o valor máximo que o byte permite, o comportamento do sistema pode ser inesperado. No meu caso, ao utilizar 266, o valor retornado foi de 10 (266-255).

De seguida o clássico Hello World:

```
section .data
    msg     db      "Hello World!",0x0a
; string hello world
    len     equ     $-msg
; calcula o tamanho da string msg

section .text           ; inicio da seccao
de texto
    global _start       ; onde deve comecar
a execucao

_start:                 ; label start - a
execucao comeca aqui
    ; write
    mov ebx, 1          ; ficheiro de saida
- stdin
    mov ecx, msg        ; apontador para o
buffer
    mov edx, len        ; tamanho do buffer
    mov eax, 4          ; chamada write ao
sistema
    int 0x80

    ; exit
    mov eax, 1          ; move o valor 1
para o registo eax
    mov ebx, 0          ; move o valor 0
para o registo ebx
    int 0x80            ; chamada de sistema
para a kernel
```

Primeiro declaramos a string Hello World usando 'db' (declare byte). No fim da string usamos o caracter de representação hexadecimal oxoa, mais conhecido por \n ou mudança de linha.

Na linha seguinte atribuímos a 'len' o tamanho da string msg, usando para isso o 'equ'. Para obter o valor do tamanho da string subtraímos a posição actual `$` ao endereço inicial de msg.

No resto do programa usamos a chamada ao sistema write, para escrever a mensagem para a consola, e depois usamos o código do exemplo anterior para retornar do programa.

Esta foi a primeira parte deste artigo, saiu mais teórica do que o previsto. Assim ficam com as bases para aprender a maior parte dos conceitos mais complexos desta fabulosa linguagem. Estejam atentos às próximas edições para o próximo artigo que vai contar com muitos mais exemplos de código.

SOBRE O AUTOR

João Matos é um estudante de 18 anos e programador autodidacta interessado por tudo o que esteja relacionado com a área de informática e das novas tecnologias. Entrará este ano no curso de Engenharia Informática e de Computadores. Foi finalista nacional das Olimpíadas Nacionais de Informática 2007.

João Matos

Programação em Ambiente Gráfico - CircularGT 2ª Parte

Introdução

Bem, tal com referi na edição anterior, para este número da revista ficou reservada a parte mais prática da questão, isto é, enquanto outrora me preocupei em explicar como funcionava cada bloco de programação, agora tenciono analisar alguns programas exemplo, bem como explicar como se processa a comunicação entre o compilador e o robô, de forma a consolidar e fortificar os conhecimentos que o leitor já adquiriu.

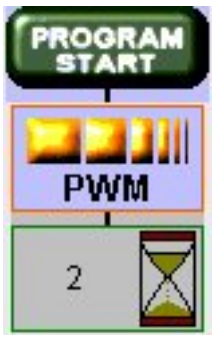
Programas exemplo

Na verdade, antes de partirmos, efectivamente, para os exemplos práticos seria interessante explicar as regras e estrutura dos programas. Com efeito, a primeira ideia a clarear é que o ciclo do programa será sempre de cima para baixo, isto é, começa pelo bloco imediatamente abaixo do bloco de configuração das entradas e saídas do microcontrolador ("PROGRAM START"), e caso não exista nenhuma comparação o bloco seguinte a ser executado é o bloco logo abaixo.

Todavia, caso exista uma comparação, existem duas hipóteses: se se verificar a condição imposta no bloco de comparação, o próximo bloco a ser executado é o bloco logo abaixo deste; se não se verificar a condição, neste caso, o bloco a ser executado é o bloco imediatamente à direita da comparação, que poderá ser uma nova comparação ou simplesmente não existir. No caso de não existir nenhum bloco à direita, é considerado como ciclo terminado e é iniciado um novo.

Explicada, basicamente, a sintaxe dos programas vamos, portanto, analisar o primeiro exemplo:

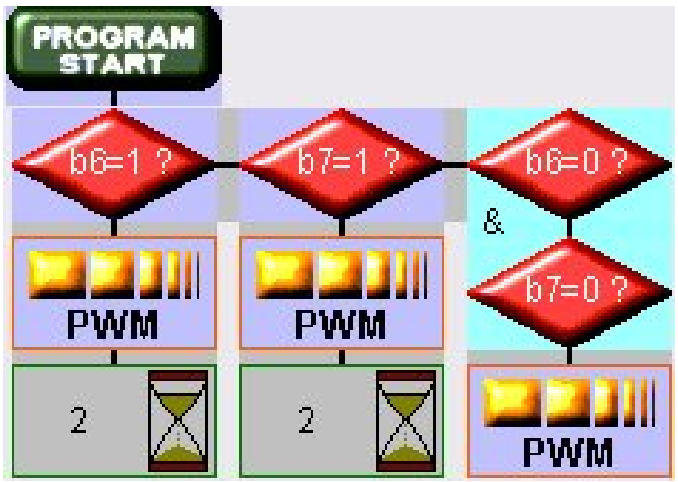
Este é, sem dúvida, dos programas mais simples desta linguagem. Como podemos observar, utilizamos um bloco PWM de regulação da velocidade dos motores seguido de um bloco temporizador.



Assim, lendo o programa de cima para baixo, vemos que o robô irá andar para a frente (conforme se pode ver na figura da direita), contudo como por baixo temos um

bloco temporizador ele não o irá fazer eternamente, mas apenas 0,5 segundos (0,5 porque como vimos no bloco temporizador o tempo encontra-se escalado em períodos de 0,25 – 0,25x2=0,5). É importante referir que há logo a passagem do bloco de PWM para o bloco de temporização pois não existe nenhum bloco de comparação, e, como tal, não se tem de verificar se uma determinada condição é verdadeira ou falsa. Passados os dois segundos, visto que não há mais nenhum bloco de instrução, o ciclo do programa é dado como finalizado, mas logo de seguida iniciar-se-á um novo, ou seja, os programas nesta linguagem estão em "loop" constante.

Visto o primeiro exemplo, está na altura de compreender algo um pouco mais elaborado:

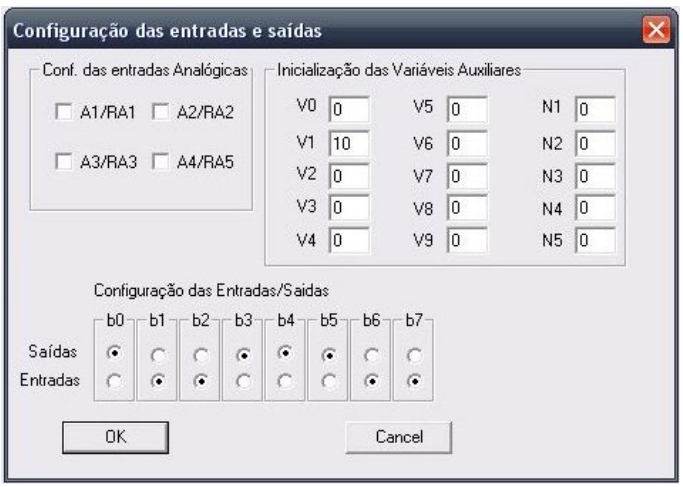


Antes de mais, convém referir que este é um bom exemplo para se demonstrar como se podem utilizar os blocos de comparação simples e com AND lógico, para além de se reforçar a compreensão dos blocos de movimentos pré-definidos e de regulação de velocidade. Feito o aparte, começarei então a explicar a estrutura do programa. Como podemos ver, o primeiro bloco imediatamente abaixo do

"PROGRAM START" é um bloco de comparação simples. Aqui o interruptor que se encontra ligado ao pino b6 vai ser testado, e caso este esteja pressionado, isto é, com o valor lógico 1, o robô fará, durante 0,5 segundos, uma rotação para a esquerda (visto que a PWM da roda esquerda está a "0" e a da direita a "255") após a qual vai voltar ao início do programa (pois está cumprido o ciclo) e fazer novamente o teste ao pino b6. Porventura, se este não estiver pressionado, vai, então, testar o bloco à sua direita. Assim, o interruptor que se encontra ligado ao pino b7 vai ser testado, e se estiver pressionado, o robô fará, ao longo de 0,5 segundos, uma rotação agora para a direita (motor esquerdo com o valor "255" e o direito com "0"), após a qual irá voltar para o início do programa. Todavia, se também o interruptor b7 não estivesse accionado, seria então testado o bloco ainda mais à direita. Neste caso, temos dois blocos "ligados" por um AND lógico, ou seja, para que se utilize as velocidades definidas no bloco PWM (que neste caso é que o dispositivo ande em frente – ambos os motores a "255"), é necessário que ambas, e não apenas uma, as condições se verifiquem. Por outras palavras, para que o robô execute o que se definiu no bloco de configuração da velocidade dos motores, não só é necessário que b6 não esteja pressionado, como também o interruptor b7.

Neste sentido, se ambas as condições se cumprirem, é executado o último bloco, e o programa volta novamente ao início. É de notar, que ao contrário dos anteriores, desta vez a seguir ao bloco de PWM não existe nenhum bloco de temporização. Isto tem, evidentemente, a sua lógica, pois se pensarmos que aos pinos b6 e b7 são ligados os sensores de contacto, deduzimos logo que visto estes não estarem activos, então o robô não foi contra nenhum objecto, pelo que pode prosseguir a sua acção. Concluindo, este é então um óptimo exemplo para se poderem testar os sensores do hardware.

Para finalizar, passarei a explicar um último exemplo, essencial para compreender os blocos de actuação, principalmente a funcionarem como contadores:



Antes de passar à explicação dos blocos em si, convém referir que a variável V1 foi colocada com o valor 10 (conforme se pode observar na figura), todavia podia ter sido outro valor entre 0 e 255. Passando agora à explicação do programa em si, vemos que o primeiro bloco é um bloco de comparação, o que significa que o robô irá realizar uma determinada tarefa enquanto V1 for menor que 13. Assim, como na configuração das entradas e saídas definimos que V1 é igual a 10, então a condição verifica-se, pelo que o motor irá andar 0,25 segundos (0,25 porque como vimos no bloco temporizador o tempo encontra-se escalado em períodos de 0,25 – 0,25x1=0,25) para a frente (neste caso a PWM dos dois motores está a 255), e de seguida incrementará a variável V1 em uma unidade. Neste sentido, ao incrementar a variável, V1 passa a ter o valor 11, pelo que a condição do primeiro bloco (V1<13) ainda se verifica. Ora se a condição do bloco ainda se verifica, então repetir-se-á novamente a primeira coluna do programa, isto é, o robô irá andar mais 0,25 segundos para a frente, incrementando novamente V1 em uma unidade. Neste momento, temos V1 com o valor 12, pelo que a condição ainda não é falsa. Visto que a condição não é falsa, então o robô andará novamente 0,25 segundos para a frente, incrementando no fim mais uma unidade.

Neste ponto, já temos que V1 é igual a 13, pelo que neste momento deixamos de ver a primeira condição válida. Se a primeira condição não é válida, então o dispositivo irá executar a instrução logo à sua direita. Aqui temos novamente um bloco de comparação simples, em que a condição é que V1 seja menor que 15. Como tínhamos que V1 era igual a 13, então isso é verdadeiro, o que levará o robô a realizar essa nova tarefa. Neste caso, agora o robô terá de andar não para a frente, mas rodar para a direita (roda esquerda a 255 e roda direita a 0) mais 0,25 segundos, aumentando, no fim, V1 para 14. Com V1 igual a 14 a condição continua ainda firme, repetindo-se a tarefa anterior, o que levará V1 a tomar o valor 15. Visto isto, a segunda condição também deixa de ser verdadeira, o que nos levará ao bloco imediatamente à direita. Neste bloco, é-nos indicada uma nova condição, sendo necessário que V1 seja menor que 16 para se efectuar uma rotação para a esquerda (roda esquerda a 0 e roda direita a 255) durante 0,5 segundos (pois o bloco temporizador indica dois períodos). Feita a rotação, V1 é novamente incrementado, passando a assumir o valor 16. Com efeito, como V1 é igual a 16 a condição da terceira coluna deixa de ser verdadeira, pelo que se vai passar agora para a comparação da quarta coluna. Como tal, V1 é efectivamente menor que 17 (tem o valor 16), pelo que se fará uma outra rotação para a direita, mas apenas de 0,25 segundos.

Realizada esta tarefa, como é definido é feito mais um incremento, pelo que agora V1 está a 17. Por fim, como V1 não cumpre a quarta condição, vai testar a condição da última coluna, que visto ser verdadeira levará o robô a andar

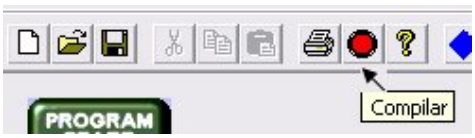
para trás (ambas os motores a 0) durante 0,75 segundos. Ora, agora em vez de um incremento, no final da coluna temos um bloco de atribuição de valores. Como o próprio nome indica, vamos atribuir um valor a variável V1, que sendo 10 fará com que se volte ao início do programa e se execute, novamente, todas as tarefas anteriormente mencionadas.

Em suma, este programa, através de incrementos (que também poderiam ser decrementos) e atribuição de valores às variáveis, faz com que o dispositivo realize uma sequência de movimentos, que bem elaborada, trabalhada e "alongada", poderá habilitar o robô a representar uma coreografia (falo por experiência própria!).

Comunicação compilador/robô

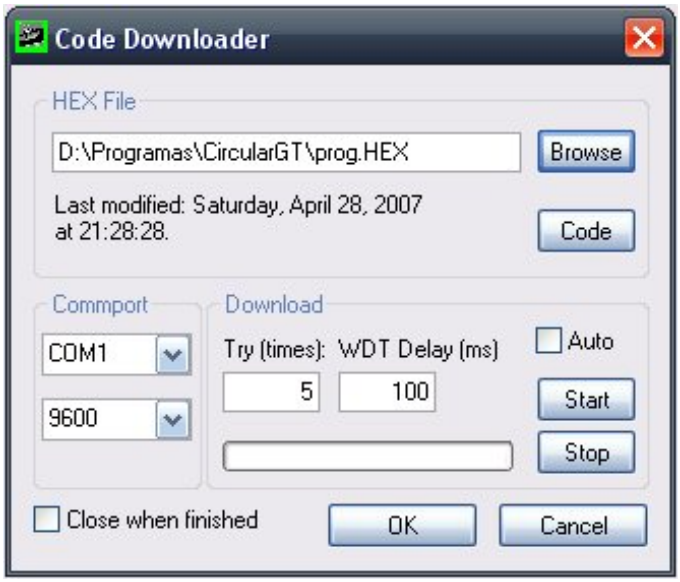
Neste momento, depois do estudo de todos os blocos e de alguns exemplos concretos, o básico e intermédio da programação já está deveras consolidado. Visto isso, torna-se agora importante perceber como é que é feita a comunicação entre o compilador e o robô, isto é, como se descarregam os programas na PIC do dispositivo.

Assim, para descarregar o programa devemos, em primeiro lugar, efectuar a sua compilação, ou seja, depois dele estar produzido devemos carregar no botão de compilação existente na Toolbar, conforme se pode observar na figura abaixo.



Depois de efectuada a compilação é, então, automaticamente aberta uma aplicação chamada "CD2M". Agora, torna-se importante a existência de um cabo RS232, essencial para estabelecer a comunicação entre a entrada série do PC e do robô.

Ligado o terminal macho ao robô e o terminal fêmea ao computador, na aplicação CD2M devemos especificar a porta série que estamos a utilizar "Com1", "Com2", "Com3" ou "Com4" no campo Commport e seleccionar a velocidade de comunicação de 9600. De seguida, utilizando-se a opção Browse abre-se o ficheiro Prog.hex que se encontra na directoria onde foi instalado o software de programação.
Finalmente, com o robô desligado (interruptor geral "OFF") liga-se o interruptor de Programação (põe-se "ON"), e seguidamente é que se liga o interruptor geral ("ON"). Carregamos em START e vemos então o programa a ser enviado para o microcontrolador. Quando aparecer a mensagem "Code has been downloaded" podemos desligar o interrutor geral e de programação (por esta ordem). Retira-se o cabo série, e accionando o interruptor geral vemos o dispositivo electrónico a fazer o que definimos na programação.



Conclusão

Agora que chegamos ao final do artigo, considero que os leitores estão já com um certo à-vontade em relação a esta linguagem em ambiente gráfico. Contudo, agora cabe a cada um aprofundar e "trabalhar" a informação adquirida de forma a tornar-se um "expert" na programação deste robô.

SOBRE O AUTOR

Apesar de apresentar especial paixão pelo futebol, Sandro Pinto é um declarado amante da informática, particularmente da componente electrónica e de hardware. Tendo concluído o curso tecnológico de electrotecnia e electrónica com média de 19,1, ingressará na universidade este ano na área de Engenharia Electrónica e de Computadores.

Sandro Pinto

Grafos
1ª Parte

Esta é a primeira parte de um artigo que pretende fazer com que o leitor compreenda e use grafos sem quaisquer problemas.

Contudo, dada a extensão do tema, fui forçado a dividi-lo em duas partes. Uma bastante mais cansativa e outra bastante mais interessante. Fui forçado a tal facto uma vez que a teoria é tão importante como a prática.

Deixo-vos contudo um pequeno aperitivo da segunda: Flood Fill. Na segunda parte vamos concentrar os nossos esforços em algoritmos cada vez mais complexos, de modo a cobrir o máximo possível deste mundo.

Introdução - O que é um grafo

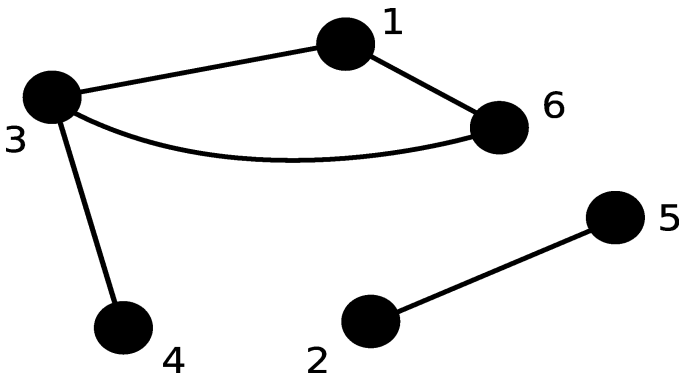
O leitor certamente que já ouviu falar em grafos. São amplamente usados em matemática, mas sobretudo em programação. Formalmente, um grafo é uma colecção de vértices (V) e uma colecção de arcos (E) constituídos por pares de vértices. É uma estrutura usada para representar um modelo em que existem relações entre os objectos de uma certa colecção. Pense nos vértices como "locais". O conjunto dos vértices é o conjunto de todos os locais possíveis. Nesta analogia, os arcos (ou arestas) representam caminhos entre estes locais. O conjunto E (vou usar o termo mais comum - "E" do inglês "edges") contém todas as ligações entre os locais.

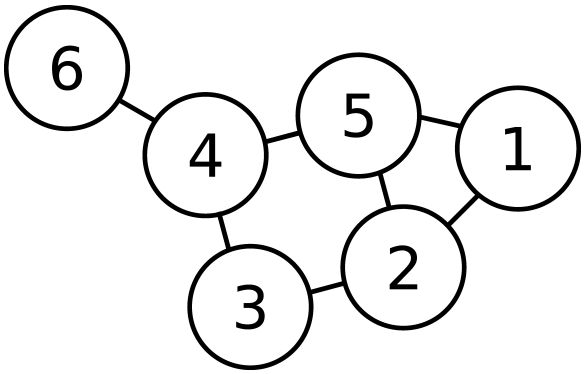
Utilizar grafos é de grande utilidade na representação de problemas da vida real.Podem ser cidades, e uma rede de estradas. Redes de computadores. Até mesmo os movimentos de um cavalo num tabuleiro de xadrez podem ser representados através de um grafo.

E depois de representá-los correctamente, o que podemos descobrir? O caminho mais curto entre duas cidades num mapa; dadas as coordenadas de n cidades, que estradas construir de modo que o número de quilómetros de estrada seja mínimo mas fiquem todas conectadas; dado um mapa de uma casa (em que paredes e chão são representados com caracteres diferentes) saber qual a divisão com maior área; etc.

As possibilidades são imensas, e ficarão admirados com a facilidade com que estes problemas são resolvidos.

Graficamente, um grafo é normalmente representado da seguinte forma:





Vértices são pontos ou círculos; Arcos são linhas entre eles.

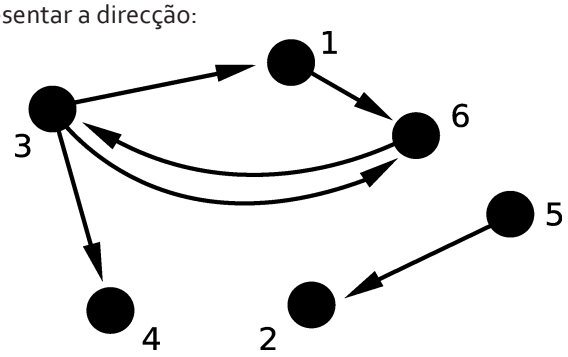
Usando o primeiro exemplo, V={1, 2, 3, 4, 5, 6} e E = {(1,3), (1,6), (2,5), (3,4), (3,6)}.
Cada vértice (também chamado "nó") é um membro do conjunto V. Cada arco é um membro do conjunto E.

Terminologia

Como é de esperar, tendo aplicações tão variadas, o grafo adapta-se às nossas necessidades. Assim, existem vários tipos de grafos. Aliados a isso, existem termos comummente usados para descrever um grafo, ou parte dele. Vou listar alguns (entre eles os mais comuns):

- **Vértice isolado** - Um vértice é considerado isolado se não possuir nenhuma ligação a outro vértice
- **Grafo trivial (ou ponto)** - Grafo sem arestas e um único nó.
- **Laço (ou loop/self-loop)** - Um arco é um laço se em ambas as extremidades estiver o mesmo vértice. (nenhum dos grafos apresentados possui laços)
- **Grafo simples** - Um grafo é simples se não contiver laços nem arcos repetidos em E.
- **Vértices adjacentes** - Dois vértices (u e v) são adjacentes se existir um arco que possui uma extremidade em u e outra em v. Os vizinhos de um vértice são todos os vértices adjacentes a ele.

- Grafo pesado (ou grafo de arcos pesados) - A cada aresta está associado um valor. Pode ser uma distância, um custo, seja o que for.
Uma definição similar existe para grafo de nós pesados.
- Grau de um vértice (ou valência) - O grau de um vértice é o número de arcos que lhe são incidentes. Um arco (u,v) é incidente tanto no vértice u como no vértice v.
Pode-se distinguir grau de entrada e grau de saída em grafos direccionados (ver à frente).
- Grafo direccionado - Cada arco tem um nó de origem e um nó de chegada. O exemplo típico é o das redes de estradas, uma vez que existem estradas só com um sentido seria o caos se um GPS não soubesse distingui-las.
Para os representar, são desenhados com setas para representar a direcção:



Da teoria para a prática

Certamente que tudo o que foi demonstrado até agora é muito bonito, mas o que interessa ao leitor é saber como transformar aqueles desenhos e aquelas setas em algo que o seu programa possa interpretar. Existem diversas formas de os armazenar, umas que ocupam mais espaço em memória mas mais eficientes, outras que ocupam menos espaço, são mais eficientes mas mais difíceis de programar. Vou somente falar das mais habituais.

- Matriz de adjacência
Esta é uma representação bastante fácil de programar, mas bastante pouco eficiente em termos de memória.
Consiste uma matriz "N x N" (onde N é um número de vértices) onde (i,j) indica se existe uma ligação do vértice i para o vértice j. Alternativamente, pode representar o tamanho dessa ligação.

Vantagens:
• Fácil de programar.
• Pode representar um grafo pesado sem comprometer a complexidade.
• Caso o grafo não seja pesado e seja possível existir mais do que uma ligação entre dois vértices, (i,j) pode representar o número de ligações entre eles.
• Verificar se dois vértices são adjacentes é muito rápido, tal como adicionar ou remover ligações.

Desvantagens:
• Elevado desperdício de memória (especialmente se o grafo for disperso).
• Debugging é difícil, uma vez que a matriz tende a ser grande.
• Listar todas as arestas incidentes num dado vértice é demorado (força-nos a percorrer todos os vértices). Como exemplo, o primeiro gráfico apresentado teria este aspecto:

	V1	V2	V3	V4	V5	V6
V1	0	0	1	0	0	1
V2	0	0	0	0	1	0
V3	1	0	0	1	0	1
V4	0	0	1	0	0	0
V5	0	1	0	0	0	0
V6	1	0	1	0	0	0

- Listas de adjacência
Nesta representação limitamos-nos a guardar a informação de todas as arestas incidentes num dado vértice.
Isto pode ser feito usando um vector de tamanho V (número de vértices), onde v[i] guardará a lista dos arcos ou vértices conectados ao vértice i.
A maneira de guardar estes vértices/arcos varia, dependendo da linguagem, podendo-se usar listas encadeadas ou mesmo transformar o vector numa matriz.

Vantagens:
• Listar todas as arestas incidentes num dado vértice é fácil (a operação mais frequente na maioria dos algoritmos).
• Baixo desperdício de memória.

Desvantagens:
• Dependendo da implementação, difícil de programar.
• Representar um grafo pesado implica uma matriz de estruturas ou mais um campo na lista encadeada.
• Para verificar se dois vértices são adjacentes necessitamos de percorrer todos os vértices adjacentes a um deles.
Mais uma vez, o primeiro gráfico fornecido poderia ser representado desta forma:

Vértice	Vértices Adjacentes
1	3, 6
2	5
3	6, 4, 1
4	3
5	2
6	3, 1

Resolvendo problemas - Algoritmos comuns

- Flood Fill
Geralmente estes problemas são muito simples de perceber, e mais ainda de implementar. Vejamos um exemplo:
"Num país distante, um terramoto destruiu uma grande parte das estradas que ligavam as suas cidades. Assim, certas zonas ficaram sem qualquer meio de contactar com outras. O nosso objectivo é saber quantas destas zonas é que existem. Uma zona é definida como sendo uma ou mais cidades, sem qualquer ligação ao exterior."

Sem grandes esforços, podemos chegar a um algoritmo simples:
• Precisamos de um vector que nos diga, num dado instante, se o elemento i foi ou não visitado (inicialmente todos os elementos começam a 0 (não visitado).
• Começando no primeiro elemento, marcar todos os elementos que lhe estão ligados como visitados. E todos os que estão ligados a esses (que ainda não foram marcados, obviamente) também. E por aí fora.
• Se quando chegarmos ao fim ainda houver elementos por marcar, então estamos perante a existência um novo centro urbano.

Um flood fill pode ser feito basicamente de 2 maneiras: em profundidade (dfs - depth-first search) ou em largura (bfs - breadth-first search) (simplificando, pois existem mais alternativas como o "dfs with iterative deepening" ou "breadth-first scanning").

– Breadth-first search
Podemos ver a pesquisa em largura como uma torneira aberta num chão de tijoleira: a primeira a ser molhada é a que se encontra no local onde está a torneira. Todas as que estão à volta serão as próximas, e por aí em diante, numa expansão a partir de um dado centro.
Este algoritmo normalmente não levanta muitos problemas a implementar, quer iterativa como recursivamente: temos o nosso vector de elementos visitados (e por visitar) e uma lista dos elementos que acabaram de ser visitados (recentemente). Para cada um destes últimos, adicionamos a uma nova lista os seus vizinhos ainda não visitados. Depois de termos feito isto com todos os elementos, passamos para a nova lista (e podemos esquecer a antiga).

– Depth-first search
O algoritmo de pesquisa em profundidade normalmente são mais complicados de compreender, porque dão trabalho a implementar iterativamente e muita gente não está familiarizada com a recursividade. Contudo, após esse problema estar ultrapassado, é ainda mais fácil de implementar do que o bfs.

Vamos ver o dfs como um rato à procura de um queijo perdido num labirinto: ele escolhe um caminho, e mal encontra um beco sem saída volta para trás e vira pelo primeiro caminho ainda não pesquisado. É exactamente este funcionamento que vamos implementar, desta vez em pseudo-código.

Partindo do vértice inicial, V.

```
função dfs(v)
  marcar v como visitado
  para todos os vértices i adjacentes a v
    se i não tiver sido visitado
      dfs(i)
```

– Problemas modelo
Street Race [*International Olympiads in Informatics '95*]
Dados: um grafo direccionado, um ponto inicial e um ponto final.
Encontrar todos os pontos "p" que um caminho do ponto inicial para o ponto final deve atravessar obrigatoriamente.

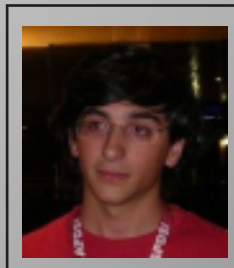
Análise: O algoritmo mais simples é remover cada ponto e verificar se o ponto final ainda é alcançado a partir do ponto inicial.

The Castle [*International Olympiads in Informatics '94*]
Dados: um mapa de um castelo (uma matriz) onde "#" representa uma parede e "." uma casa em branco.
Descobrir qual o tamanho da maior sala do castelo, após deitar abaixo uma das paredes.

Análise: Ao ser fornecida a matriz temos uma representação implícita do grafo, cada casa sendo um vértice. Não precisamos de a transformar numa matriz/lista de adjacência, podemos usá-la para saber quais os vértices adjacentes.
Mais uma vez, deitamos uma parede abaixo e verificamos o tamanho da maior sala.

Fontes: Wikipedia e USACO Training Program (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

Revista Linux

A Verdadeira Revista Portuguesa de Linux

Edição Bimestral em formato PDF

Download Gratuito

www.revista-linux.com

Serialização de Objectos em Java

Neste artigo iremos falar sobre serialização de objectos em Java. Mas afinal o que é a serialização de objectos? A serialização de objectos é o processo de conversão de um objecto a uma sequência de bytes, assim como o processo que permite passar a sequência de bytes para um objecto utilizável e válido.

A serialização em Java é bastante simples e a API disponível pela SUN para este efeito é muito directa e intuitiva. No decorrer do artigo iremos ver exemplos práticos de serialização binária, e serialização de XML embora esta não faça parte dos padrões de serialização do Java.

Vamos começar por criar uma classe simples que irá depois ser usada para serialização.

```
import java.io.Serializable;

public class Exemplo1 implements
Serializable {

    private int numero;
    private String nome;

    public Exemplo1(int numero, String
nome) {
        this.numero = numero;
        this.nome = nome;
    }

    public String getNome() {
        return nome;
    }

    public int getNumero() {
        return numero;
    }

    public String toString() {
        return new String("Numero =
"+this.numero+" | Nome = "+this.nome);
    }
}
```

[Exemplo1.java]

Como podemos ver, é uma classe bastante simples e nada fora do normal com duas variáveis "numero" e "nome" um construtor e os métodos GET. Para além disso tem também um Override do método toString que retorna uma String com os valores das variáveis. A única parte digamos fora do normal é mesmo a implementação do interface Serializable importado na linha 1 da nossa classe. Este interface vai permitir que os Objectos gerados por esta classe possam ser serializados e vice versa.

Temos agora uma classe de teste que irá serializar os objectos criados da classe Exemplo1.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

```
public class Teste1 {

    public static void main(String args
[]) {
        Exemplo1 e1 = new
Exemplo1(001,"White");
        Exemplo1 e2 = new
Exemplo1(002,"Magician");


System.out.println(e1.toString());

System.out.println(e2.toString());

        ObjectOutputStream out;
        ObjectInputStream in;

        try {
            out = new
ObjectOutputStream(new
FileOutputStream(System.getProperty("user
.dir")+File.separator+"Exemplo1.bin"));
            out.writeObject(e1);
            out.writeObject(e2);
            out.flush();
            out.close();
        } catch(Exception e) {
            e.printStackTrace();
        }

        Exemplo1 e3;
        Exemplo1 e4;

        try {
            in = new
ObjectInputStream(new
FileInputStream(System.getProperty("user.
dir")+File.separator+"Exemplo1.bin"));
            e3 = (Exemplo1)
in.readObject();
            e4 = (Exemplo1)
in.readObject();

            in.close();


System.out.println(e3.toString());

System.out.println(e4.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

[Teste1.java]

Esta classe é constituída apenas pelo método main visto que tem por objectivo demonstrar a serialização de objectos para um ficheiro binário. Começamos por criar alguns objectos, neste caso dois e o conteúdo desses objectos será impresso no ecrã com o auxílio do método toString implementado na classe Exemplo1. Inicializamos um ObjectOutputStream que irá criar um ficheiro binário de nome Exemplo1.bin na directoria actual. Este stream irá permitir serializar os objectos no ficheiro. Agora basta serializar os objectos, serialização essa realizada pelo método writeObject(Object object), que irá guardar o objecto dado como argumento no ficheiro de destino sob a forma binária.

Agora vamos fazer o processo inverso, passar os objectos serializados no ficheiro binário para objectos Java válidos. Este processo é tão simples como o seu inverso. Para isso iremos começar por criar dois novos objectos da classe Exemplo1 sem os inicializar. Em seguida inicializamos um ObjectInputStream que irá ler o ficheiro Exemplo1.bin criado anteriormente. Este stream contém o método readObject() que lê um objecto serializado num ficheiro. Como podemos ver, os objectos retornados pelo método readObject() são guardados nas variáveis e3 e e4 criadas anteriormente. Podemos ver também que nestas mesmas linhas é feito um cast para (Exemplo1). O cast deve ser sempre feito porque, embora os objectos serializados sejam do tipo Exemplo1 o método readObject() retorna o tipo genérico Object, que depois deve ser convertido para o tipo original desse objecto.

Por fim vamos imprimir os objectos e3 e e4 tal como fizemos anteriormente para o e1 e e2 e, se todo o processo correr normalmente, os valores de e1 serão iguais aos de e3 e o mesmo acontece com e2 e e4, o output. Neste caso será algo como o que podemos ver em seguida.

```
Numero = 1 | Nome = White
Numero = 2 | Nome = Magician
Numero = 1 | Nome = White
Numero = 2 | Nome = Magician
```

Vamos agora ver uma outra forma de serialização, embora use o mesmo mecanismo que o exemplo anterior. Neste exemplo os objectos serão serializados para um ByteArray. Esta técnica pode ser bastante útil para envio de grandes quantidades de objectos pela rede.

Para isso vamos usar novamente a classe Exemplo1 e uma nova classe de teste semelhante à classe Teste1 usada anteriormente.

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

```
public class Teste2 {
    public static void main(String args
[]) {
        Exemplo1 e1 = new
Exemplo1(001,"White");

        Exemplo1 e2 = new
Exemplo1(002,"Magician");


System.out.println(e1.toString());
System.out.println(e2.toString());

        ByteArrayOutputStream buffer =
new ByteArrayOutputStream();
        ObjectOutputStream out;
        ObjectInputStream in;

        try {
            out = new
ObjectOutputStream(buffer);
            out.writeObject(e1);
            out.writeObject(e2);
            out.flush();
            out.close();
        } catch(Exception e) {
            e.printStackTrace();
        }

        Exemplo1 e3;
        Exemplo1 e4;

        try {
            in = new
ObjectInputStream(new
ByteArrayInputStream(buffer.toByteArray()
));
            e3 = (Exemplo1)
in.readObject();
            e4 = (Exemplo1)
in.readObject();

            in.close();


System.out.println(e3.toString());
System.out.println(e4.toString());

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

[Teste.java]

À semelhança do exemplo anterior nesta classe são criados dois objectos da classe Exemplo1 e em seguida os seus valores são impressos.

As diferenças principais começam agora. Criamos um ByteArrayOutputStream que será usado como buffer para guardar os objectos serializados, e à semelhança do exemplo anterior, inicializamos o ObjectOutputStream, mas ao contrário do primeiro exemplo, em que damos como argumento um FileOutputStream usado para escrever no ficheiro, neste caso damos como argumento o ByteArray criado. Assim cada objecto serializado será guardado no ByteArray e não num ficheiro. Depois, são serializados dois objectos Exemplo1 para o ByteArray que será usado mais tarde.

O ByteArrayOutputStream que criamos pode ser usado para várias coisas recorrendo ao método toByteArray() que retorna uma array de bytes com todos os dados do nosso ByteArrayOutputStream. As utilidades deste array são inúmeras, mas neste caso vamos apenas usá-lo para construir um ObjectInputStream, e assim iremos conseguir recuperar os nossos objectos serializados.

Por fim vamos imprimir os valores dos nossos novos objectos e iremos ver que os valores coincidem tal como no exemplo anterior.

```
Numero = 1 | Nome = White
Numero = 2 | Nome = Magician
Numero = 1 | Nome = White
Numero = 2 | Nome = Magician
```

Para terminar este artigo sobre serialização de objectos em Java iremos fazer o processo usando XML ao invés do ficheiro binário como vimos anteriormente. Comecemos por criar uma classe Exemplo2.java.

```
public class Exemplo2 {
    private int numero;
    private String nome;

    public Exemplo2 () {
    }

    public Exemplo2(int numero, String
nome) {
        this.numero = numero;
        this.nome = nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void setNumero(int numero) {
```

```
        this.numero = numero;
    }

    public String getNome() {
        return nome;
    }

    public int getNumero() {
        return numero;
    }

    public String toString() {
        return new String("Numero =
"+this.numero+" | Nome = "+this.nome);
    }
}
```

[Exemplo2.java]

Para começar vamos ver algumas diferenças de implementação entre as classes Exemplo2 e Exemplo1. Neste novo caso a classe não implementa a interface Serializable, como acontecia nos exemplos anteriores, e para além disso este processo requer alguns cuidados: a classe deve conter os métodos set e get para todas as variáveis do objecto que queremos serializar, em contraste com os exemplos anteriores, em que apenas implementávamos os métodos que queríamos. Deve também ser implementado o construtor vazio. Atenção que a serialização para XML apenas consegue realizar-se com visibilidade public, logo os métodos get e set, bem como os construtor, devem ser public, caso contrário os valores não serão serializados.

Vejamos agora a classe que irá proceder à serialização.

```
import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class SerialXML {

    public static void main(String args
[]) {
        Exemplo2 e1 = new
Exemplo2(001,"White");
        Exemplo2 e2 = new
Exemplo2(002,"Magician");


System.out.println(e1.toString());
System.out.println(e2.toString());
        try {
            XMLEncoder encoder = new
XMLEncoder( new BufferedOutputStream(
new
FileOutputStream(System.getProperty("user
.dir")+File.separator+"Exemplo2.xml")));
            encoder.writeObject(e1);
            encoder.writeObject(e2);
            encoder.flush();
            encoder.close();
        } catch(Exception e) {
            e.printStackTrace();
        }

        Exemplo2 e3;
        Exemplo2 e4;

        try {
            XMLDecoder decoder = new
XMLDecoder(new BufferedInputStream( new
FileInputStream(System.getProperty("user.
dir")+File.separator+"Exemplo2.xml")));
            e3 =
(Exemplo2)decoder.readObject();
            e4 =
(Exemplo2)decoder.readObject();
            decoder.close();


System.out.println(e3.toString());

System.out.println(e4.toString());
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

[SerialXML.java]

Para este processo precisamos das classes java.beans.XMLEncoder para gravar os objectos e java.beans.XMLDecoder para os recuperar do arquivo XML.

Tal como nos exemplos anteriores, esta classe apenas contém o método main e começa por criar dois objectos da classe Exemplo2 e em seguida imprimir os seus valores.

Vamos agora passar ao processo de serialização. Para isso começamos por criar um objecto XMLEncoder que vai permitir serializar os objectos num ficheiro XML.À semelhança dos exemplos anteriores, iremos usar o método writeObject. Depois destes passos já teremos um ficheiro Exemplo2.xml com o seguinte conteúdo:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0"
class="java.beans.XMLDecoder">
 <object class="Exemplo2">
  <void property="nome">
   <string>White</string>
  </void>
  <void property="numero">
   <int>1</int>
  </void>
 </object>
 <object class="Exemplo2">
  <void property="nome">
   <string>Magician</string>
  </void>
  <void property="numero">
   <int>2</int>
  </void>
 </object>
</java>
```

Como se pode ver, o ficheiro contém os objectos serializados, a apresentação dos dados é bastante bem estruturada e simples de ler e até mesmo editar, aliás esta é uma das vantagens da serialização face à serialização binária.

Temos agora o processo inverso, passar de XML para objectos. vamos criar um objecto XMLDecoder e vamos usar o método readObject disponibilizado por este objecto.

Por fim e tal como nos exemplos anteriores teremos o seguinte output.

```
Numero = 1 | Nome = White
Numero = 2 | Nome = Magician
Numero = 1 | Nome = White
Numero = 2 | Nome = Magician
```

Para finalizar a serialização de objectos basta apenas acrescentar que na serialização binária é possível definir determinadas variáveis ou constantes de forma a que, no momento da serialização, estas não sejam serializadas. Para isso basta usar a palavra reservada transient. A sua utilização é bastante simples, basta colocá-la antes do tipo e nome da variável, ou seja, algo semelhante a private transient String password; assim a variável password não será serializada.

Podem ainda consultar as APIs utilizadas neste artigo em:

- http://java.sun.com/javase/6/docs/api/java/io/Serializable.html
- http://java.sun.com/javase/6/docs/api/java/io/ObjectOutputStream.html
- http://java.sun.com/javase/6/docs/api/java/io/ObjectInputStream.html
- http://java.sun.com/javase/6/docs/api/java/beans/XMLEncoder.html
- http://java.sun.com/javase/6/docs/api/java/beans/XMLDecoder.html

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

IPSec – Protocolo de Segurança IP

IPSec significa Internet Protocol Security. Em português corrente pode-se traduzir para protocolo de segurança IP. Este protocolo visa ser o método padrão para o fornecimento de privacidade, integridade e autenticidade das informações transferidas através de redes IP.

O IPSec permite a construção de túneis seguros sobre redes (internet ou intranet). Tudo o que passa através da rede é cifrado pelo gateway IPSec e decifrado pelo outro extremo da comunicação. Disto resulta uma Rede Privada Virtual (VPN).

O IPSec pode ser usado em qualquer máquina que disponha de rede IP (router, PC, ...), e assenta em dois protocolos:
• ESP (Encapsulating Security Payload), que disponibiliza encriptação e autenticação;
• IKE (Internet Key Exchange), responsável pela troca de chaves para o ESP incluindo a sua ligação.

Os protocolos IPSec foram desenvolvidos pelo IETF (Internet Engineering Task Force) e são implementados tanto em IPv4 como IPv6. De referir ainda que quase todos os fornecedores de soluções de firewall ou software de segurança suportam o IPSec.

O IPSec pode ser usado para proteger os dados que circulam entre dois computadores, como por exemplo, um servidor de aplicações e um servidor de base de dados. O IPSec torna-se completamente transparente para as aplicações porque os serviços de criptografia, integridade e autenticação são implementados no nível de transporte. As aplicações continuam a comunicar-se umas com as outras, da maneira habitual, usando as portas TCP e UDP.

Vantagens do uso do IPSec

A principal vantagem do IPSec é que, ao nível da camada de rede, pode proteger qualquer tipo de tráfego sobre IP. Isto não acontece por exemplo com o SSH, PGP (email), HTTPS, entre outros.

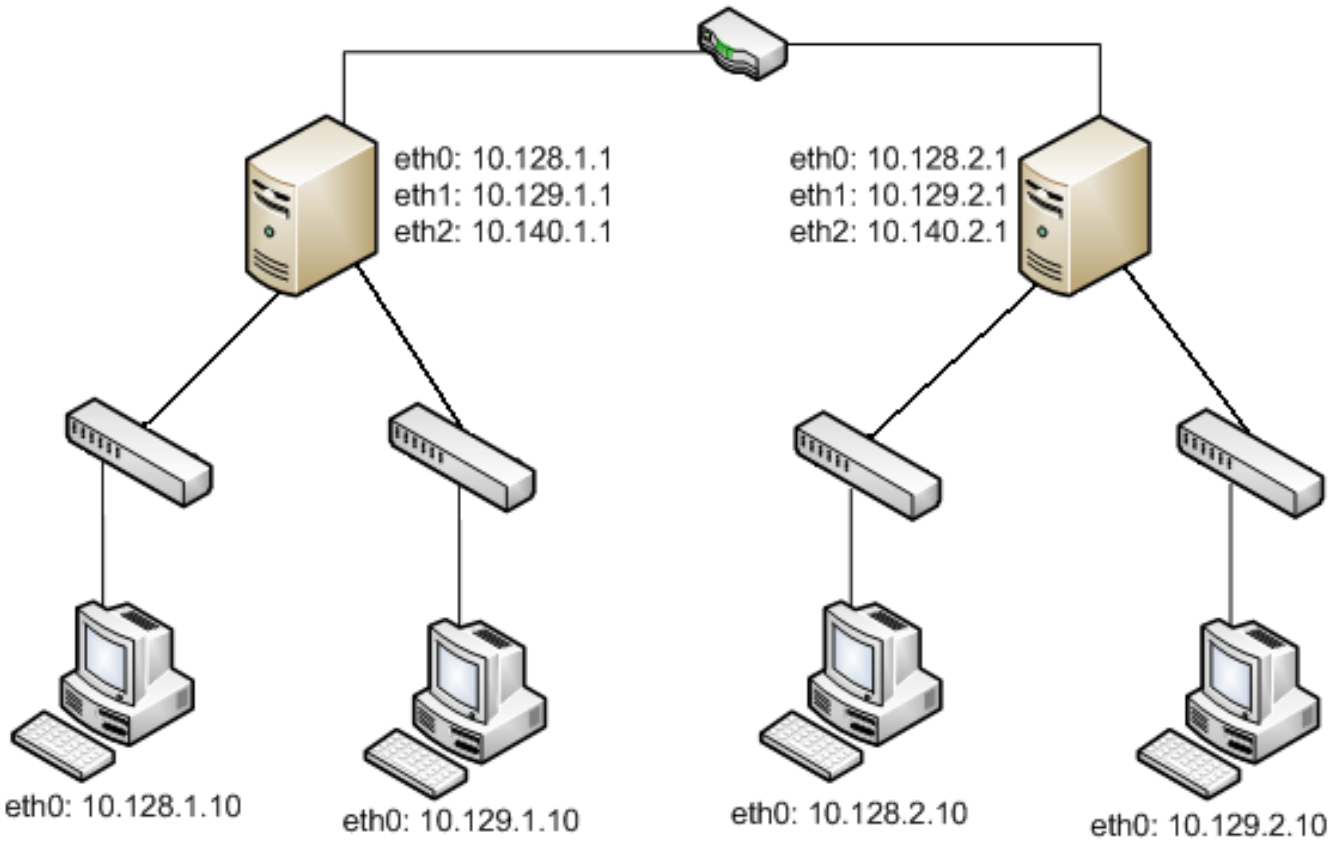
Como o IPSec funciona ao nível da camada de rede, pode ser usado para todo o tipo de tráfego, como por exemplo:
• Formar túneis entre uma máquina com IP dinâmico e um gateway remoto ("Road Warrior");
• Interligar de forma segura, cifrando os dados, vários locais classificados como inseguros, como é o caso da Internet;
• Economizar, causada pela diminuição do número de linhas alugadas para acesso à Internet e de comunicação directa entre as empresas e as suas filiais;
• Tornar as comunicações privadas, pois usa os mecanismos mais testados da área da criptografia;
• Flexibilizar, pois pode residir em servidores, clientes móveis ou firewalls.

Usando o IPSec, pode-se:
• Manter as mensagens confidenciais, cifrando todos os dados transmitidos entre dois computadores;
• Proporcionar integridade às mensagens transmitidas entre dois computadores (sem cifrar os dados);
•Proporcionar autenticação mútua entre dois computadores. Por exemplo, pode-se ajudar a proteger um servidor de base de dados estabelecendo um canal que permita enviar solicitações só a partir de um determinado computador cliente (por exemplo, um servidor de aplicações ou Web);
• Restringir quais os computadores que podem comunicar entre si. Pode-se também restringir a comunicação com protocolos IP e portas TCP/UDP específicas.

Vejamos um exemplo prático que demonstra a implementação do IPSec rede a rede. Serão usados routers Linux com três interfaces de rede.
Supõe-se que existe a rede que está esquematizada na página seguinte.

Agora parte-se do princípio que é necessário cifrar os dados entre as redes 10.129.1.0/24 e a 10.129.2.0/24, ou seja, se o terminal 10.129.2.10 enviar um pacote para o 10.129.1.10, este terá que ir cifrado. Parte-se do princípio também que, o administrador de rede que vai implementar o IPSec só tem acesso físico aos terminais com os IPs 10.128.2.10, 10.129.2.10 e ao respectivo router Linux, por isso vai ter de efectuar as todas as configurações do outro lado do esquema da rede recorrendo ao acesso remoto (ssh).

É necessária a instalação do software openswan nos routers linux. Pode-se fazer o download no seguinte site: http://www.openswan.org/. Nas distribuições mais conhecidas, pode-se instalar o software com o gestor de pacotes, se estiver nos repositórios:



[root@revistaprogramar ~]*# yum install openswan*
[root@revistaprogramar ~]*# aptitude install openswan*

Posteriormente vai-se inicializar o serviço:

```
[root@revistaprogramar ~]# service ipsec start
```

E verificar se ficou bem instalado:

```
[root@revistaprogramar ~]# ipsec verify
```

Deve aparecer um output semelhante a este:

```
Checking your system to see if IPsec got installed and started correctly
Version check and ipsec on-path                 [OK]
Checking for KLIPS support in kernel            [OK]
Checking for RSA private key (/etc/ipsec.secrets)  [OK]
Checking that pluto is running                  [OK]
```

Estes comandos têm de ser efectuados nos dois routers Linux.

Agora, supondo que o administrador está no router linux a que tem acesso, é necessário obter a chave pública do IPSec tanto desse router como no outro. Para isso recorre-se ao seguinte comando:

[root@revistaprogramar ~]*# ipsec showhostkey --right*

Deverá aparecer a respectiva chave:

```
# RSA 2192 bits   hp   Mon Jul 28 17:43:21 2007
rightrsasigkey=0sAQOF3Xy0JBTrFCtrL6cICHhrAl/ZAGKbnfvuXrDUMCuVuGDReQ3YqkRnbZF+zSkvzC0GErTgurLi8h47TSlHK4vaVUNIzJUKDQv0sD7l4e8znxmOcXy1cms3KPCybvVynV9XLGEgpnQz6OZiRYZ6OcyD179jZyGf0sf46mClMB0G8j9/Zr0x4x5Omb9N0x/S75Dt4d3xeIm6/znk3sBo9RmHmXJMC6f+XkPDXGT/ciIuzJfdVwb5kABmUBSAEtzwdjn1yzaDKKZWFXJwD9n1VSiOEMiNIWIaFsqG2ituu3CvN4UnvryIcMDrJJkT/PsDmLgr2cpxG3IHtj9bnsOVDgbRQ403hkEybt50KJ1fwzVnM4dn
```

Se não for gerada nenhuma chave, será necessário criar uma, executando o seguinte comando:

```
[root@revistaprogramar ~] ipsec newhostkey –output /etc/ipsec.secrets
```

Agora é necessário ligar-se por ssh ao outro router Linux, e repetir o mesmo procedimento, apenas trocando a palavra right por left:

[root@revistaprogramar ~]# *ssh 10.140.1.1*
[root@10.140.1.1 ~]# *ipsec newhostkey --output /etc/ipsec.secrets*
[root@10.140.1.1 ~]# *ipsec showhostkey --left*

 # RSA 2192 bits hp Mon Jul 28 17:47:21 2007
leftrsasigkey=0sAQOJX+Z6JJkEWyPgV/4aIZwz4Krpv+1crqs1z2ekxuVUXY958DeP+jVW4OmVsk7GV7I/1LdVYA4AUm31eXVp4UK500JIxnTfq3S+WTyT6KcoMW47Ri1BbZ+CJCCuBiEKC7JHRJkSYpYFdPdDcCOulXaJBeWReKovSZrYpm1ctZ9IIg76yC13yIaK9j6PR/IADPIesCiOd2lMi/uXG57kyeyxw04ahvd5b7MR0oZ9OTGzyBPnHqcX89vNuASX0t90v3AjA16LB4TcsheE+b8e2Rqx+EBal09CPkRF6L7opipWl84lo2O3VgyMCGKkUMFd8i2vT79NornhZz3RWO3sV+DheJS93ki4sB82iGH6fBzZBRdf

[root@10.140.1.1 ~]# *exit*

Voltando ao sistema local, será necessário editar o ficheiro /etc/ipsec.conf de forma a reflectir os dados do gateway. O ficheiro deverá ser configurado da seguinte maneira:

```
# /etc/ipsec.conf - Openswan IPsec
configuration file
#
# Manual:     ipsec.conf.5
#
# Please place your own config files in
/etc/ipsec.d/ ending in .conf

version    2.0    # conforms to second
version of ipsec.conf specification

# basic configuration
config setup
# Debug-logging controls:  "none" for
(almost) none, "all" for lots.
   # klipsdebug=none
   # plutodebug="control parsing"
   nat_traversal=yes

include /etc/ipsec.d/*.conf
```

```
conn local1-to-local2

left=10.140.1.1

leftsubnet=10.129.1.1/24

leftid=10.129.3.10

leftrsasigkey=0sAQOJX+Z6JJkEWyPgV/4aIZwz4K
rpv+1crqs1z2ekxuVUXY958DeP+jVW4OmVsk7GV7I/
1LdVYA4AUm31eXVp4UK500JIxnTfq3S+WTyT6KcoMW
47Ri1BbZ+CJCCuBiEKC7JHRJkSYpYFdPdDcCOulXaJ
BeWReKovSZrYpm1ctZ9IIg76yC13yIaK9j6PR/IADP
IesCiOd2lMi/uXG57kyeyxw04ahvd5b7MR0oZ9OTGz
yBPnHqcX89vNuASX0t90v3AjA16LB4TcsheE+b8e2R
qx+EBal09CPkRF6L7opipWl84lo2O3VgyMCGKkUMFd
8i2vT79NornhZz3RWO3sV+DheJS93ki4sB82iGH6fB
zZBRdf

leftnexthop=%defaultroute

right=10.140.2.1

rightsubnet=10.129.2.1/24
rightid=10.129.4.10

rightrsasigkey=0sAQOF3Xy0JBTrFCtrL6cICHhrA
l/ZAGKbnfvuXrDUMCuVuGDReQ3YqkRnbZF+zSkvzC0
GErTgurLi8h47TSlHK4vaVUNIzJUKDQv0sD7l4e8zn
xmOcXy1cms3KPCybvVynV9XLGEgpnQz6OZiRYZ6Ocy
D179jZyGf0sf46mClMB0G8j9/Zr0x4x5Omb9N0x/S7
5Dt4d3xeIm6/znk3sBo9RmHmXJMC6f+XkPDXGT/ciI
uzJfdVwb5kABmUBSAEtzwdjn1yzaDKKZWFXJwD9n1V
SiOEMiNIWIaFsqG2ituu3CvN4UnvryIcMDrJJkT/Ps
DmLgr2cpxG3IHtj9bnsOVDgbRQ403hkEybt50KJ1fw
zVnM4dn

rightnexthop=%defaultroute
auto=add
```

"left" e "right" representam os routers Linux que têm o software instalado. "leftsubnet" e "rightsubnet" as máquinas que vão ser protegidas.
Depois de guardar as alterações, é necessário que o outro router Linux tenha os mesmos dados no ficheiro de configuração. Desta forma envia-se o ficheiro por ssh:

[root@revistaprogramar ~]# *scp /etc/ipsec.conf 10.140.1.1:/etc/ipsec.conf*

Agora é necessário iniciar a ligação nos dois gateways (routers Linux):

[root@revistaprogramar ~]# *ipsec auto --up local1-to-local2*
[root@10.140.1.1 ~]# *ipsec auto --up local1-to-local2*

Para verificar se os dados estão a passar cifrados, pode-se mandar um simples ping do terminal 10.129.2.10 para o 10.129.1.10 e usar um sniffer como o tcpdump num dos gateways:

[root@revistaprogramar ~]*# tcpdump --i eth2*

Os dados estão a passar cifrados se aparecer um output semelhante a este:

```
listening on eth2, link-type EN10MB (Ethernet), capture size
96 bytes
18:03:24.348436 IP 10.140.2.1 > 10.140.1.1:
ESP(spi=0xd7d78753,seq=0xc4e), length 132
12:03:24.348436 IP 10.129.2.10 > 10.129.1.10: ICMP echo
request, id 62983, seq 1698, length 64
12:03:24.348671 IP 10.140.4.1 > 10.140.3.1:
ESP(spi=0x56f73530,seq=0xc50), length 132
```

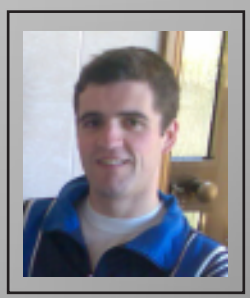
Por fim resta referir que para activar o IPSec no boot do computador tem de se trocar a última linha do ficheiro /etc/ipsec.conf de

auto=add
para
auto=start

e copiar o respectivo ficheiro para o outro gateway.

Note-se que os dados passam em claro do terminal a ser protegido até ao respectivo gateway, só sendo cifrados aí e enviados até ao outro gateway. Os dados são decifrados e encaminhados até ao respectivo destino novamente em claro.

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 3º 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

Criar um chat em Flash com Smartfox Server

Smartfox Server é uma plataforma criada em Java que permite o desenvolvimento rápido de aplicações multi-utilizador em Actionscript. É actualmente usada para a criação de comunidades como o Habbo Hotel e alguns jogos multiplayer como o Zwok.

Vou ensinar a criar uma sala de chat que contém os princípios básicos e essenciais desta plataforma.
Antes de começar a programar é necessário instalar o servidor Smartfox que pode ser baixado no site oficial:

http://www.smartfoxserver.com

Existem várias versões, desde a versão lite até à versão pro. Para seguir este tutorial basta a versão lite mas recomendo a versão Pro que contém um painel para administrar as salas e utilizadores.

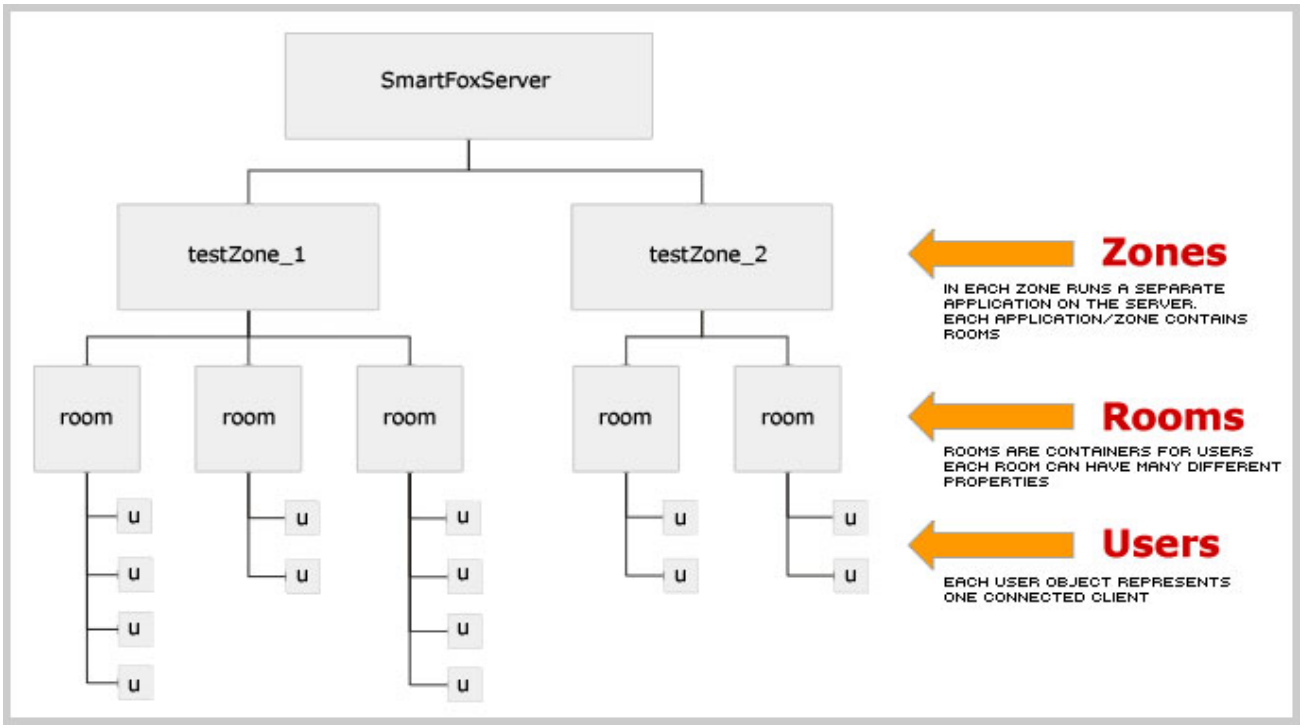
Depois de instalar o servidor é necessário instalar o API, basta ir à pasta de instalação e depois procurar pela pasta Flash API/Actionscript 2.0 e fazer duplo clique no ficheiro SmartFoxClient_AS2.mxp para instalar (Atenção que as pastas variam conforme a versão do servidor instalado). Agora executamos o servidor para começarmos a programar e testar a aplicação.
Já no flash criei a seguinte estrutura na timeline:

```
import it.gotoandplay.smartfoxserver.*;

var smartFox:SmartFoxClient = new SmartFoxClient();

init();
function init(){
    stop();
    smartFox.connect("127.0.0.1", 9339);
    smartFox.onConnection = handleConnection;
    status_txt.text = "a ligar...";
}
function handleConnection(success){
    if(success){
        status_txt.text = "ligado";
        gotoAndStop("login");
    } else {
        status_txt.text = "ligação falhou";
    }
}
```

Está dividida em 3 secções: ligação ao servidor, login e por fim a sala de chat.

Na primeira frame apenas coloquei uma caixa de texto dinâmico com instance "status_txt", e o seguinte código:

Começamos por importar as classes do Smartfox Server e criar o objecto da classe SmartFoxClient. A função init faz o pedido de conexão ao servidor e define a função handleConnection como ouvinte para a resposta da conexão, que verifica se teve sucesso ou falhou. No caso de sucesso avança para a página de login, se falhou mostra uma mensagem nesse sentido.



Na frame "login" coloquei um input text com o instance "user_txt" onde vai ser digitado o user name, coloquei também um botão com o instance "login_btn". Antes de passar ao código quero explicar que o Smartfox Server é composto por zonas que por sua vez estão divididas por salas, como podem ver no esquema acima que retirei da documentação.

O código para esta frame é o seguinte:

```
var userName:String;

initLogin();
function initLogin() {
    Selection.setFocus("user_txt");
    login_btn.onRelease = login;
}
function login() {
    smartFox.login("simpleChat",
user_txt.text);
    smartFox.onLogin = onLogin;
    status_txt.text = "a verificar
login...";
    login_btn.enabled = false;
    login_btn._alpha = 60;
}

(...)
```

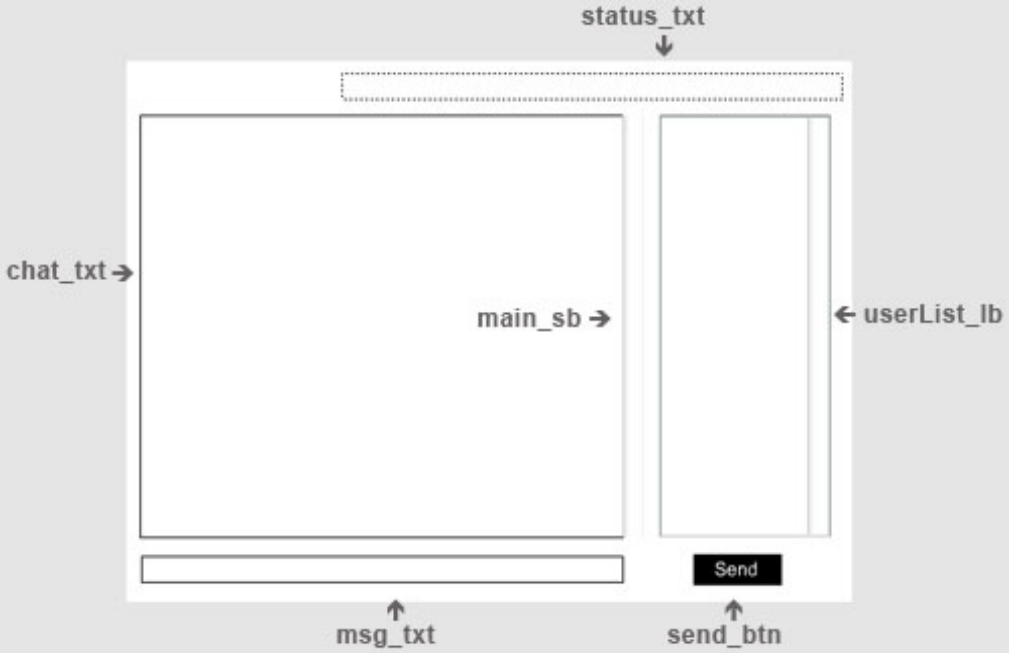
A função login que é executada ao clicarmos no botão de login, faz o pedido de login e define a função onLogin como ouvinte do evento com o mesmo nome.

```
(...)

function onLogin(resObj:Object) {
    if (resObj.success) {
        userName = resObj.name;
        status_txt.text = "Bem-Vindo " +
userName + "!";
        gotoAndStop("sala");

    } else {
        status_txt.text = resObj.error;
        login_btn.enabled = true;
        login_btn._alpha = 100;
    }
}
```

Ao receber a resposta do login é então executada a função onLogin que verifica se teve sucesso, se for o caso é guardado o nosso user name, mostra a mensagem de boas vindas e avança para a frame onde se encontra a sala de chat. No caso de falhar, é mostrada a mensagem de erro enviada pelo servidor.

Ao clicar no botão login poderia colocar uma validação do user name, mas o Smartfox ao receber um user name vazio cria um do género guest_x. Para usar password pode-se verificar através de um server-side script e só depois fazer o pedido de login ao Smartfox.

Na frame "sala" vamos então criar a sala de chat, que consiste na seguinte estrutura bastante típica das salas de chat, como se pode observar na página seguinte:



E agora a parte de código mais extensa:

```
import it.gotoandplay.smartfoxserver.*;

initSala();
function initSala() {
    chat_txt.html = true;
    smartFox.onRoomListUpdate =
handleRoomListUpdate;
    smartFox.onJoinRoom =
handleJoinRoom;
    smartFox.onJoinRoomError =
handleJoinRoomError;
    smartFox.onUserEnterRoom =
handleUserEnterRoom;
    smartFox.onUserLeaveRoom =
handleUserLeaveRoom;
    smartFox.onPublicMessage =
handlePublicMessage;
    send_btn.onRelease = sendMsg;
}

function
handleRoomListUpdate(roomList:Object) {
    smartFox.autoJoin();
}

(...)
```

A função initSala define as funções ouvintes dos vários eventos. O primeiro evento depois de fazer login é receber a lista de salas existentes na nossa zona. Ao receber este evento a função handleRoomListUpdate corre o comando para entrarmos na sala predefinida da zona.

```
(...)

function handleJoinRoom(roomObj:Room) {
    var userList:Object =
roomObj.getUserList();
    for (var i in userList) {
        var user = userList[i];
        userList_lb.addItem(user.getName(
), user.getId());
    }
    userList_lb.sortItemsBy("label",
"ASC");
    chat_txt.htmlText = "Bem-vindo ao "
+ roomObj.getName();
    status_txt.text = userName;
    main_sb.scrollPosition =
chat_txt.maxscroll;
    chat_txt.scroll =
chat_txt.maxscroll;
}
function
handleJoinRoomError(errorMsg:String) {
    status_txt.text = errorMsg;
}

(...)
```

Ao entrarmos na sala com sucesso, é executada a função handleJoinRoom que recebe um objecto com os dados da sala e começa por ir buscar e apresentar a lista de utilizadores dessa sala. No caso de falhar a entrada na sala a função handleJoinRoomError mostra o erro enviado pelo servidor.

```
(...)

function
handleUserEnterRoom(roomId:Number,
userObj:Object) {
    userList_lb.addItem(userObj.getName()
, userObj.getId());
    userList_lb.sortItemsBy("label",
"ASC");
    chat_txt.htmlText +=
userObj.getName() + " entrou na sala";
    main_sb.scrollPosition =
chat_txt.maxscroll;
    chat_txt.scroll =
chat_txt.maxscroll;
}
function
handleUserLeaveRoom(roomId:Number,
userId:Number) {
    for (i = 0; i <
userList_lb.getLength(); i++) {
        var item:Object =
userList_lb.getItemAt(i);
        if (item.data == userId) {
            var userName:String =
item.label;
            userList_lb.removeItemAt(i);
            break;
        }
    }
    userList_lb.sortItemsBy("label",
"ASC");
    chat_txt.htmlText += "" + userName +
" saiu da sala";
    main_sb.scrollPosition =
chat_txt.maxscroll;
    chat_txt.scroll =
chat_txt.maxscroll;
}

(...)
```

Esta duas funções são executadas quando entra ou sai um utilizador da nossa sala. No primeiro caso, a função handleUserEnterRoom adiciona o novo utilizador à lista e acrescenta na janela de chat uma mensagem de entrada desse novo utilizador. No segundo caso, a função handleUserLeaveRoom percorre a lista de utilizadores até encontrar o utilizador com o id equivalente ao que abandonou a sala, remove-o da lista e acrescenta na janela de chat uma mensagem indicando a saída do utilizador.

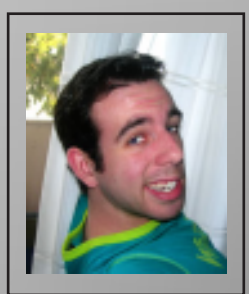
```
(...)

function handlePublicMessage(msg:String,
sender:User) {
    chat_txt.htmlText += "[ " +
sender.getName() + " ] " + msg;
    main_sb.scrollPosition =
chat_txt.maxscroll;
    chat_txt.scroll =
chat_txt.maxscroll;
}
function sendMsg() {
    if (msg_txt.text != "") {
        smartFox.sendPublicMessage(msg_tx
t.text);
        msg_txt.text = "";
    }
}
```

Quando alguém envia uma mensagem, é executada a função handlePublicMessage que recebe o user e a respectiva mensagem e só acrescenta na janela de chat a mensagem com a identificação do utilizador.

Finalmente, a função sendMsg é executada quando enviamos uma mensagem. Apenas envia a nossa mensagem para o Smartfox e esvazia a nossa input text. Ao enviar a nossa mensagem vai receber o mesmo evento que recebe quando alguém manda uma mensagem e assim acrescentar na janela de chat a nossa mensagem.
Este tutorial fica por aqui, tentei explicar os conceitos básicos para criar aplicações com o Smartfox. Se pretenderem aprofundar mais, o servidor vem com excelente documentação e vários tutoriais. Espero que tenham gostado e vos tenha ajudado.

Fontes: Documentação do Smartfox

SOBRE O AUTOR



Tendo iniciado a programação na componente Web, Filipe Jorge tornou-se especialista na tecnologia Flash. No entanto, e como autodidacta que é, tem extendido os seus conhecimentos a outras linguagens de programação web. Trabalha na empresa portuguesa Djomba e tem no currículo o curso profissional de técnico de multimédia.
Website: http://www.filipejorge.com

Filipe Jorge

Aplicações .NET em Linux com o Mono

Já muito se falou sobre aplicações multi-plataforma, e sobre a frase "Write once, run anywhere". Não há muitas pessoas que associem esta ideia à plataforma .NET, que é vista como exclusiva para Windows, mas na realidade a especificação .NET é completamente aberta, e foi desenhada para ser independente de plataforma, permitindo ao programador usar a linguagem e plataforma de desenvolvimento que prefere, sem que isso limite onde a sua aplicação irá correr.

Sendo uma especificação aberta, e embora a Microsoft não tenha implementado o .NET para outras plataformas (à parte algumas experiências, como o Rotor), nada impede que alguém pegue nas especificações e implemente um sistema compatível .NET - e houve vários que o fizeram, nomeadamente o Projecto Mono(1).

O que é o Mono

O Mono é uma implementação em código aberto de duas especificações ECMA (334 - C#, 335 - CLI), compatível com o MS.NET, e englobando uma máquina virtual, a linguagem C# e bibliotecas básicas, seguindo o standard ECMA. Além disso, o Mono contém também uma biblioteca de compatibilidade com outros componentes .NET, como por exemplo o ASP.NET, ADO.NET, ou Windows Forms, incluindo ainda um compilador de VB.NET. O Mono está disponível em Linux, FreeBSD, UNIX, MacOS, Solaris e Windows.

O projecto Mono foi iniciado na Ximian (agora Novell) por Miguel de Icaza em 2001, e a primeira versão pública foi lançada em 2004. A versão mais recente à altura da escrita deste artigo é a 1.2.4, suportando já .NET 2.0 e parcialmente 3.0. A versão 1.2.5 deverá sair em meados de Agosto, contendo já uma implementação compatível com o Silverlight (implementação light para web do WPF, a nova API do .NET 3.0 para GUIs).

Instalação

Para correr aplicações .NET em plataformas não-Windows, é necessário instalar o Mono - na página do projecto estão disponíveis pacotes para vários sistemas (2), e links para outros sites que distribuem pacotes não oficiais. Está também disponível uma imagem VMWare que inclui o openSuse e uma instalação completa de todos os componentes do Mono, para que o possa testar sem ter que o instalar no seu sistema, se assim o preferir.

No mínimo, irá precisar de instalar o pacote mono-core, que contém a máquina virtual e tudo o que é necessário para correr aplicações .NET. Se a aplicação que quer correr foi feita em Windows Forms, precisa também de instalar o pacote mono-winforms e o pacote libgdiplus. Para ASP.NET, precisa do pacote mono-web e do mod_mono para o apache - em alternativa pode usar o xsp, um pequeno servidor de web com suporte para ASP.NET. Para aceder a bases de dados, instale o pacote mono-data. Se a sua aplicação é ASP.NET e está feita em VB.NET, irá precisar também do pacote mono-basic, que contém este compilador.

Certifique-se que o Mono está correctamente instalado abrindo uma linha de comandos e escrevendo "mono". Deverá obter uma mensagem de ajuda.

Portar uma aplicação .NET para Mono

Idealmente, todas as aplicações .NET usariam somente as bibliotecas públicas, que o Mono suporta a 100%, e por isso todas correriam na perfeição em qualquer plataforma. Mas, obviamente, as bibliotecas disponíveis no MS.NET são muito mais extensas, suportando todo o tipo de funcionalidades, muito para além das especificações, e embora o projecto Mono se esforce para manter a máxima compatibilidade com o .NET Framework da Microsoft, há sempre bugs a corrigir e mais funcionalidades a acrescentar e melhorar. Nem todas as aplicações .NET correm a 100% no Mono, dependendo muito do nível de suporte das funcionalidades usadas.

Outro problema que se coloca é o de nem todas as aplicações .NET usarem exclusivamente managed code (nome que se dá ao código feito numa linguagem .NET, seja ela qual for) - muitas aplicações recorrem a bibliotecas externas feitas em C, C++ ou outras linguagens unmanaged através de um mecanismo intitulado P/Invoke. Muitas bibliotecas comerciais de geração de PDFs e GUIs, por

exemplo, recorrem ao acesso directo a funções de Windows através de chamadas a bibliotecas como a shell32.dll, e como é óbvio, portar aplicações que usam directamente funções nativas de Win32 para Linux ou MacOS é uma tarefa complicada.

Como então determinar se uma aplicação que temos pré-compilada é 100% compatível com o Mono? Para isto recorremos a um pequeno utilitário chamado MoMA(3), que corre tanto em Linux com o Mono ou em Windows com o MS.NET, e que analisa um executável ou DLL .NET e determina o nível de compatibilidade com o Mono, gerando um relatório de tudo o que pode não ser suportado a 100%, bem como de todos os P/Invoke a DLLs de Windows.Para correr o MoMA no Linux, basta puxar o pacote disponibilizado no site, ir a uma linha de comandos e escrever "mono MoMA.exe" na directoria para onde colocou o conteúdo do pacote. Embora possa parecer estranho correr uma aplicação com extensão .exe no Linux, a verdade é que a compatibilidade total entre o que é produzido no MS.NET e o Mono exige que mesmo os nomes de ficheiros tenham extensão - .exe para executável, .dll para bibliotecas.

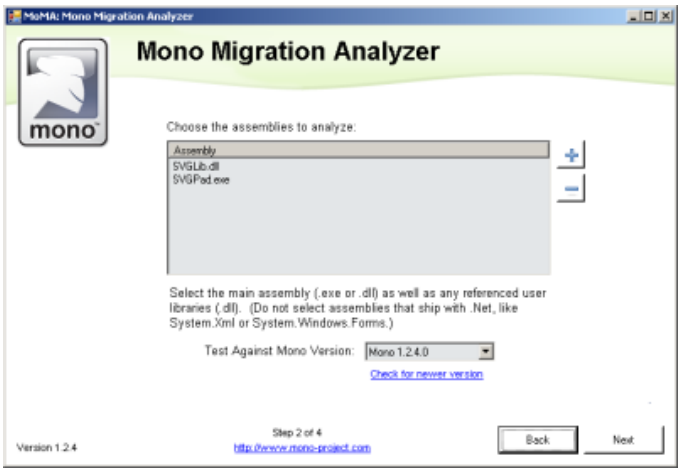
Após seleccionar a aplicação que quer analisar (inclua o executável e todas as DLLs que possam vir juntamente com este), o MoMA irá produzir um relatório indicando potenciais problemas. Mesmo que a aplicação não seja 100% compatível, é possível que corra por o código incompatível não ser usado senão em situações específicas. Caso tenha problemas de compatibilidade, verifique o relatório com atenção - pode dar-se o caso das incompatibilidades não afectarem a operação da aplicação (uma incompatibilidade pode ser tudo desde funcionalidades em falta até diferenças de píxeis no desenho de uma caixa).

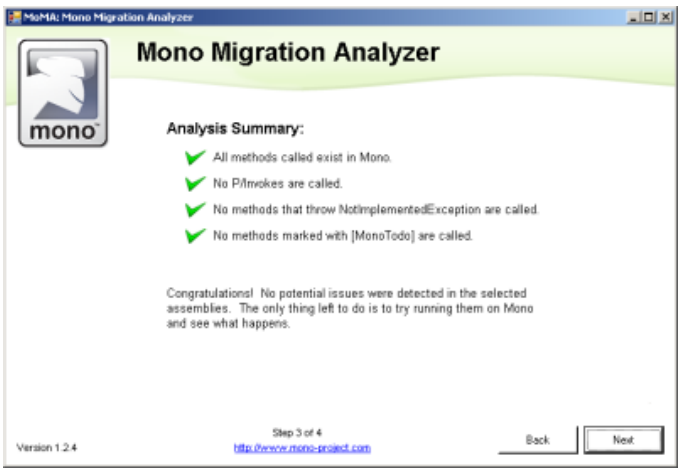
Exemplo prático

Para ver como tudo funciona, nada melhor que um exemplo prático. Uma boa fonte de aplicações .NET é o site Code Project (4), quem tem vindo a acumular muito código fonte, bibliotecas e aplicações para testar as mais variadas funcionalidades do .NET (entre outros frameworks). Para o nosso exemplo prático, escolhi a aplicação SVGPad (5), um editor de ficheiros svg em C# que usa alguns dos controlos e funcionalidades mais comuns do .NET.

Na página do SVGPad está disponível o código fonte e a aplicação pré-compilada. Visto que o objectivo é correr aplicações .NET sem recompilar, este exemplo usa a aplicação pré-compilada obtida na página (6), que consiste num executável - SVGPad.exe - e numa biblioteca – SVGLib.dll.

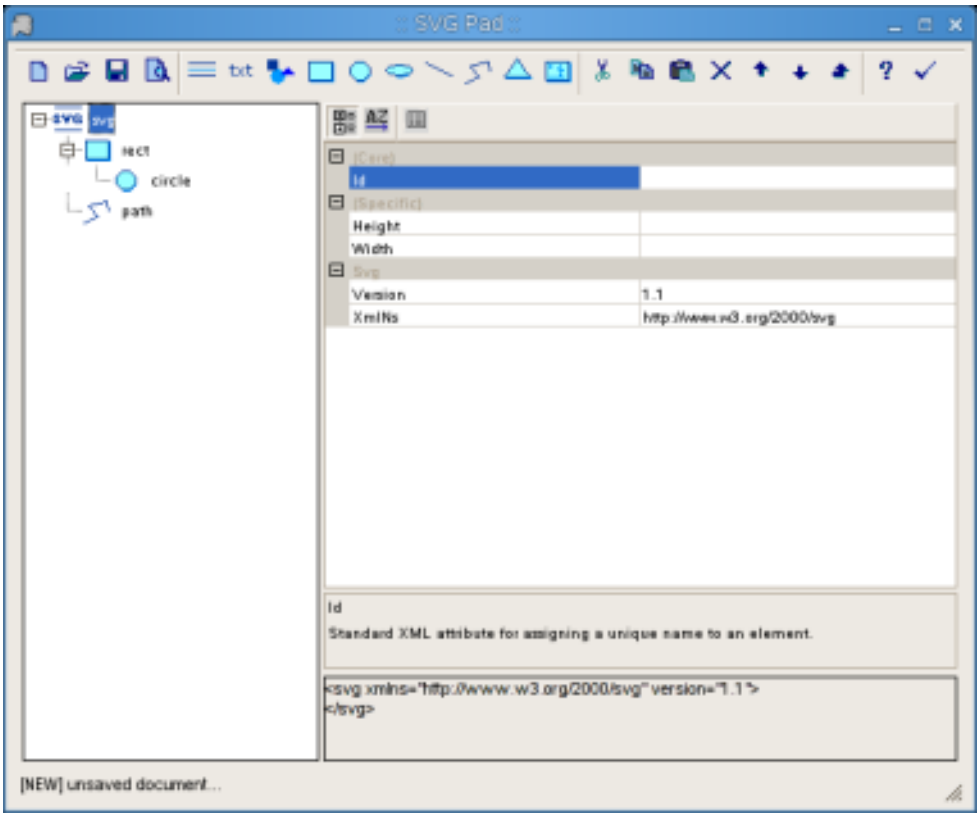
O primeiro passo após desempacotar a aplicação será correr o MoMA, para verificar se a aplicação usa alguma funcionalidade que ainda não seja totalmente suportada pelo Mono, ou se usa alguma biblioteca específica de Windows.





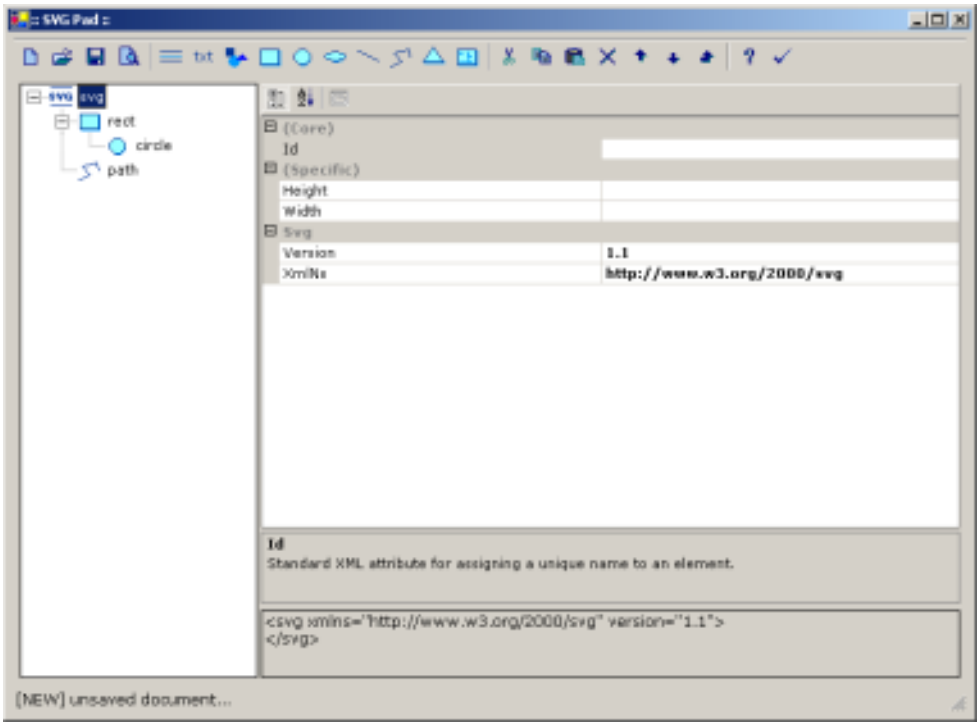
A análise do MoMA indica que esta aplicação é completamente suportada pelo Mono (v.1.2.4), pelo que podemos corrê-la directamente em Linux.

Para correr o svgpad com o Mono, basta abrir uma linha de comandos (Gnome Terminal, X Terminal, Konsole ou outra), ir à directoria onde está o svgpad, e escrever "mono SVGPad.exe".



SVGPad no Linux

SVGPad no Windows



À parte algumas diferenças mínimas de cor e arredondamento das margens que têm origem nos diferentes temas aplicados no Windows e no Linux, a aplicação corre perfeitamente nas duas plataformas sem recompilação.

Executar aplicações com o Mono

Há várias maneiras de evitar abrir uma linha de comandos para correr uma aplicação .NET com o Mono. A maneira mais simples é de criar um shell script para correr a aplicação. Crie um novo ficheiro chamado SVGPad (sem extensão) na mesma directoria onde tem a aplicação, e coloque o conteúdo abaixo indicado. Grave-o e altere as permissões para executável ("chmod a+x SVGPad"). Agora já pode executar a aplicação clicando no SVGPad.

[SVGPad]
```
#!/bin/sh
mono SVGPad.exe
```

Uma outra maneira de facilitar a execução de aplicações .NET com o Mono é usando o mkbundle. Este utilitário, exclusivo para Linux, pega na aplicação e em todas as suas dependências e cria um executável que contém dentro de si (opcionalmente comprimidos) todos os ficheiros necessários para correr a aplicação.

[mkbundle]
```
mkbundle -z SVGPad.exe SVGLib.dll -o svgpad
```

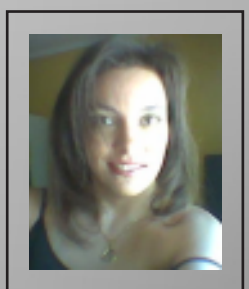
A opção -z comprime os ficheiros, para que o executável final não seja muito grande. Neste caso, o resultado é um executável "svgpad" que contém dentro de si o exe e a dll originais, e que pode ser invocado como qualquer aplicação de Linux. Este ficheiro pode ser distribuído e corrido em qualquer Linux que tenha o Mono instalado.

Recursos disponíveis

O Wiki do Projecto Mono é um bom ponto de partida para obter mais informações sobre o projecto em si, outras alternativas para correr aplicações .NET em Linux e Mac (7), como portar aplicações (8), como depurar aplicações de Windows.Forms no Visual Studio para o Mono (9), como depurar aplicações com o Mono (10), e muito mais. A equipa e a comunidade do Mono estão online todos os dias no irc (11), e há várias mailing lists de suporte activas (12) para esclarecer todas as dúvidas.

1) http://www.mono-project.com
2) http://www.mono-project.com/Downloads
3) http://www.mono-project.com/MoMA
4) http://www.codeproject.com
5) http://www.codeproject.com/csharp/svgpad.asp
6) http://www.codeproject.com/csharp/SvgPad/VGPad_bin.zip
7) http://www.mono-project.com/Guide:Running_Mono_Applications
8) http://www.mono-project.com/Guide:_Porting_Winforms_Applications
9) http://www.mono-project.com/Guide:_Debugging_With_MWF
10) http://www.mono-project.com/Debugging
11) irc.gnome.org, canal #mono
12) http://www.mono-project.com/Mailing_Lists

SOBRE A AUTORA



Depois de ser programadora, analista, dev leader e consultora, Andreia Gaita é, hoje em dia, engenheira a tempo inteiro na equipa do Projecto Mono da Novell, divertindo-se a produzir código em projectos fascinantes. Nos seus tempos livres, para relaxar, frequenta o curso de informática na FCUL.

Andreia Gaita

PTSec

www.ptsec.net

A PTSec é uma comunidade nacional de segurança informática, a única activa nesta área, por sinal um pouco "esquecida" em Portugal. Esta comunidade aborda diversas áreas da segurança, nomeadamente protecção e invasão de sistemas. Depois de alguns "obstáculos" iniciais, esta comunidade começa a "jogar para valer" no panorama nacional de comunidades on-line, esperando alcançar um lugar nos favoritos do seu browser.

edu2.0

www.edu20.org

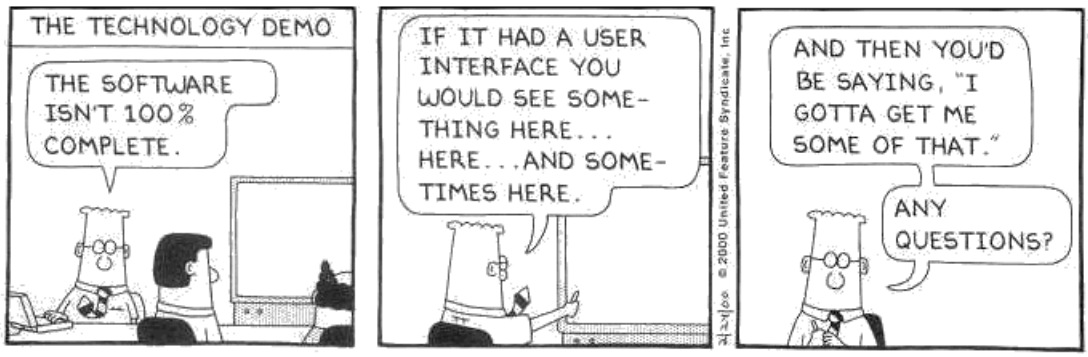
A web2.0 não pára de surpreender. Desta vez, foi no mundo da educação que as suas potencialidades se mostraram imensas, através do site edu2.0. O criador define-o como um site de educação da próxima geração que torna o ensino e a aprendizagem mais agradáveis e eficientes, mas por detrás desta pequena definição esconde-se um enorme mundo de partilha gratuita, onde qualquer um pode ensinar ou aprender com os outros, participar activamente na comunidade ou até contribuir para alargar a já enorme rede de recursos de diversos tipos, desde aulas a cursos, livros, experiências, exercícios. Esperam-se ainda novas funcionalidades, como vídeo-conferência e suporte para telemóveis.

ScienceHack

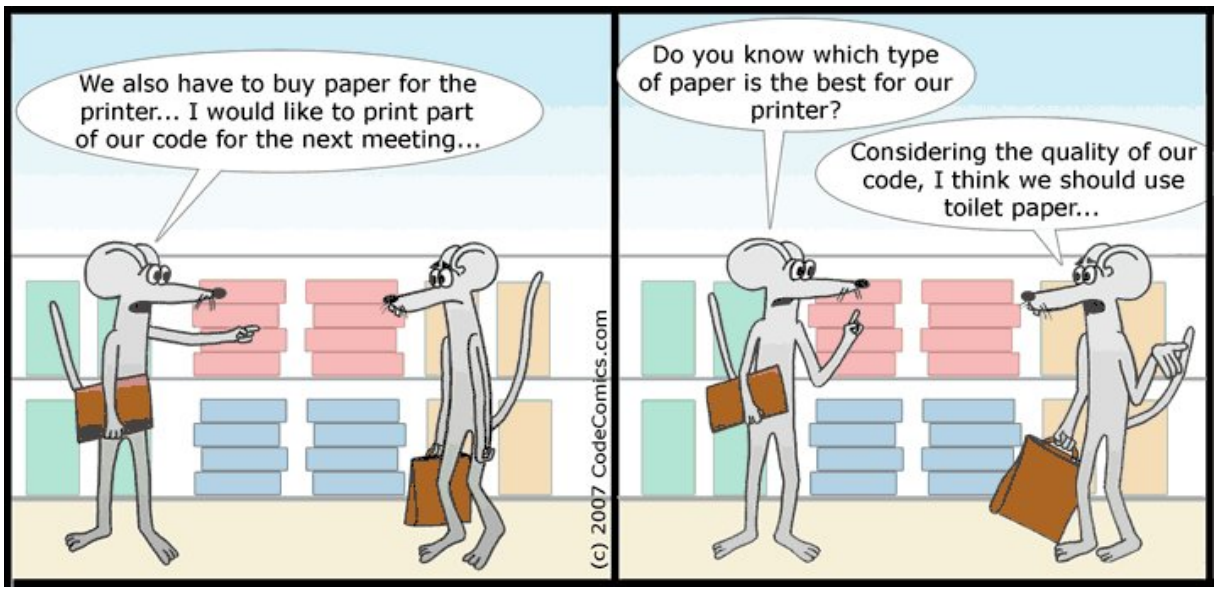
www.sciencehack.com

ScienceHack é um repositório de vídeos sobre ciência, numa tentativa de juntar todos os vídeos deste tipo que se encontram na Internet. Agrupados por 13 secções, de Matemática a Psicologia, passando por Robótica, Física, Ecologia, os vídeos podem ser submetidos por qualquer utilizador. Mas, para evitar o problema da credibilidade a que muitos projectos deste tipo estão sujeitos, todos os vídeos são analisados previamente por cientistas, garantindo a sua fiabilidade. O site disponibiliza também um motor de busca, que permite a todos os interessados encontrar vídeos de forma rápida.

Uma Beta muito particular



Código de qualidade... recompensado.



Às vezes é preciso aprender uma 25ª linguagem!

Em Desenvolvimento

• Estão em desenvolvimento vários MODs e novas características para o fórum que permitirão facilitar a navegação dos utilizadores e o trabalho dos moderadores. Entre eles destacam-se o Dropdown para Blogs (um sistema que permitirá colocar no index do fórum os blogs das várias equipas do P@P e, quem sabe, dos vários projectos que se associem).

• O Wiki P@P está a ser revolucionado em todos os aspectos (visual, script adoptado, MODs, funcionalidades e objectivos) pelo que esperamos que antes do lançamento da próxima edição haja já uma versão BETA do nosso Wiki v2.0. A plataforma basear-se-á no modelo DokuWiki que serve de melhor forma os interesses da comunidade, serão migrados os artigos do wiki antigo para o novo wiki e está a ser repensado o novo design e estrutura do mesmo.

Concluído

• Foram adicionados recentemente novos MODs que facilitam a comunicação entre os utilizadores e os moderadores. Deve aceder ao fórum para os experimentar.

• Foi concluída a instalação do Pastebin do P@P disponível em http://paste.portugal-a-programar.org. Apesar de já estar disponível para o utilizador trabalhar com a ferramenta, o Pastebin sofrerá em breve alterações a nível de design e serão adicionadas funcionalidades. Para aqueles

que não conhecem o sistema, devo dizer que ficarão com a possibilidade de colocar pequenos excertos de código, logs de IRC, enfim, pequenas notas e apresentá-los sob a forma de link para que todos possam visualizar de forma mais fácil. Este sistema não vem substituir o syntax highlight no fórum, visto que serve essencialmente para alojar código e notas para mostrar durante conversas privadas.

• Foi concluída a instalação da ferramenta TaskFreak!, uma plataforma que serve para orientação e organização das equipas da comunidade. Esta plataforma vem facilitar muitos métodos e preenche algumas lacunas que existiam no que toca à organização de cada uma das equipas.

Projectos em Destaque

System Empires

O System Empires é um jogo web-based que decorre nos planetas do nosso sistema solar, cada um com as suas características e dividido em centenas de regiões. Os jogadores começam com uma colónia na Terra e alguma economia, tentando depois desenvolver o seu império e lutar pela supremacia sob constantes ameaças de outros jogadores e de cometas.

O desenvolvimento do System Empires iniciou-se em Fevereiro de 2006, por uma equipa de três estudantes de informática. A versão pública será lançada a 22 de Setembro deste ano. Foi criada uma mailing list aberta a todos para notícias sobre o jogo.

www.systemempires.com

20 meses, **10** edições!

É com bastante orgulho que cortamos esta pequena meta. Dez edições parece pouco, mas são vinte meses de árduo trabalho, são dez pequenas, mas importantes, metas cortadas a cada dois meses. Cada edição é um desafio, cada desafio é um risco, mas têmo-los superado.

Aqui fica o nosso pequeno agradecimento a todos os que colaboraram directa ou indirectamente para a sobrevivência desta revista. O agradecimento é igual para todos, sem destaques, pois todos são ou foram uma pedra sem a qual o edifício desta revista se desmoronaria.

Andreia Gaita
Bruno Matos
Bruno Vaz
Carlos Silva
Celso Ramos
Daniel Correia
David Costa
David Ferreira
David Pintassilgo
Diogo Alves
Evandro Oliveira
Fernando Martins
Fabiano Ferreira
Fábio Correia
Fábio Monteiro
Fábio Pedrosa
Filipe Jorge
Gil Sousa
Guilherme Rodrigues
Hugo Violante
Joel Calado
Joel Ramos
José Domingos
José Oliveira
João Carvalho
João Matos
João Pereira
João Simões
Leandro Belo
Luis Macedo
Luis Madureira
Luís Rente
Marcelo Martins
Marco Silva
Miguel Araújo
Miguel Pais
Miguel Wahnon
Márcio Lima
Nuno Correia
Patric Figueiredo
Pedro Abreu
Pedro Lobo
Pedro Santos
Pedro Silva
Pedro Sousa
Pedro Teixeira
Pedro Verrume
Ricardo Amaral
Ricardo Antunes
Ricardo Ribeiro
Ricardo Rocha
Rui Gonçalves
Rui Maia
Sandro Pinto
Sérgio Lopes
Sérgio Matias
Sérgio Santos
Tiago Palhota
Tiago Salgado
Werichnilson Ataliba


PROGRAMAR
ASSEMBLY
Projectos em MONO
mono
IPSEC


PROGRAMAR
Começar a Programar
Setups em Visual Studio
C e o Makefile
Python
Entre neste mundo de desafios constantes com a melhor ajuda


PROGRAMAR
Visual Basic
Introdução ao PHP
Visual Basic.NET
XGL


PROGRAMAR
Scripts NASL
Mobilizing Web Applications
Projecto PDA
LaTeX


PROGRAMAR
Microcontroladores
JavaScript
Orion's Belt
Opera Mini


PROGRAMAR
Bases de Dados
Programação em ASP
RFID Applications
weSpendMoney
Mounting File Systems


PROGRAMAR
Programação em Scheme
Iniciação ao Python
Exploits
Gravação no Desktop


PROGRAMAR
Programação em WML
Iniciação ao C
ASPECTJ
Redes CAN


PROGRAMAR
Windows Vista para programadores
Programação em LUA
Protocolos SPI
Lua
Ubuntu 7.04
Feisty Fawn
Segurança em Sistemas Distribuídos


PROGRAMAR
AJAX & PHP
RSS Feed em PHP + MySQL
Templates em PHP
A WEB 2.0 EM ANÁLISE
Compilar e detectar erros em C

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