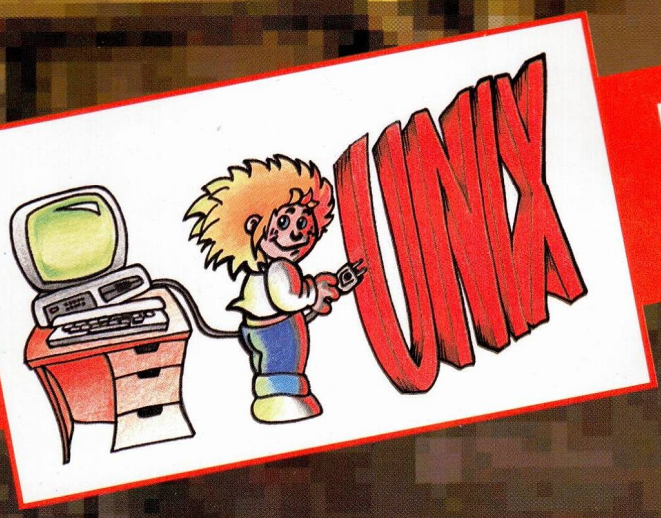
N.º 20 DEZEMBRO DE 1992 PREÇO: 550$00

Spooler

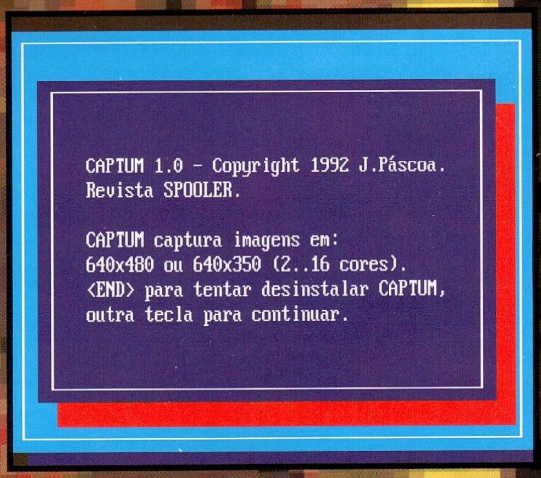


MAGAZINE DOS UTILIZADORES PC

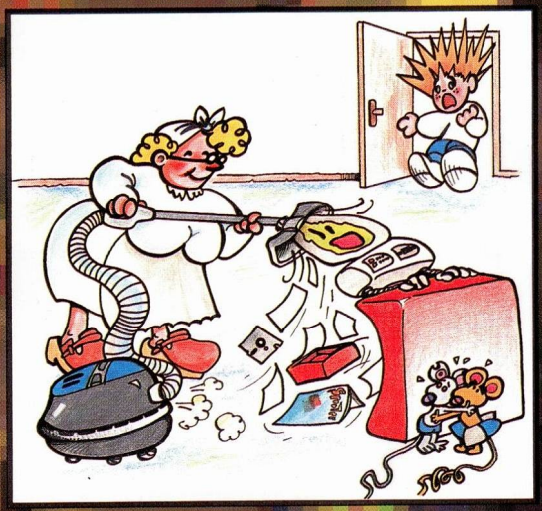


LIGANDO AO UNIX

CAPTUM!!!



- SABIA QUE...
- ABCMEDIC
- SYSINFO
- PROGRAMAR EM:
  - C
  - CLIPPER
  - ASSEMBLY
- INFORMÁTICA INDUSTRIAL



LIMPA

A disqueta gratuita que acompanha cada exemplar da revista, é parte integrante da mesma e não pode ser comercializada separadamente.

A SOLUÇÃO GRÁFICA COMPLETA!

CorelDRAW 3.0



CorelDRAW 3.0 é um produto revolucionário que responde a todas as suas necessidades gráficas apenas com UM *package*.

Quatro módulos independentes encontram-se dentro da mesma caixa: CorelDRAW, CorelCHART, Corel PHOTO-PAINT e CorelSHOW.

Juntos, representam a solução gráfica para os anos 90. CorelDRAW é o único conjunto de programas que trabalha como você: inteligentemente.

Combinando estes programas ou usando-os individualmente, você pode criar as mais extraordinárias imagens, impressos e apresentações.

NOVO

COREL DRAW!

O programa n.º 1 do mundo para ilustrações. Agora mais poderoso.

NOVO

COREL CHART

Versátil e poderoso utilitário para produzir gráficos de dados e pictogramas

NOVO

COREL PHOTO-PAINT

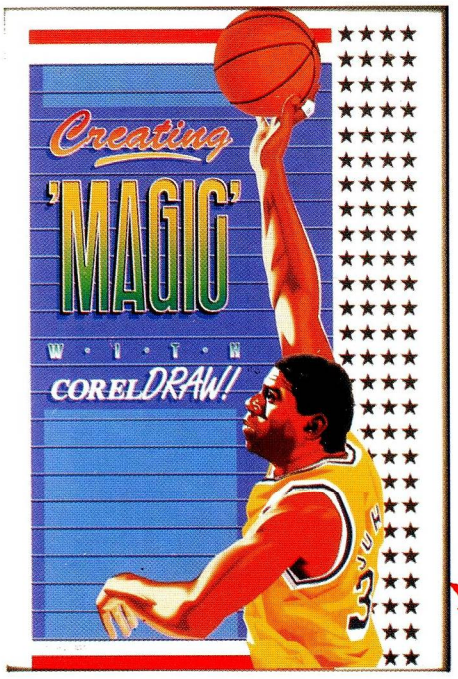
***Tratamento de fotos, e produção de imagens* bitmap.**

NOVO

COREL SHOW

A maneira mais fácil de criar uma apresentação.





E AINDA...

- **CorelMOSAIC:** Gestor de ficheiros de utilização simples e fácil. Importação, exportação e impressão de ficheiros, pré-visionamento e criação de catálogos.
- **CorelTRACE:** Um poderoso utilitário para auto-delineamento que converte imagens a cores e P/B em imagens vectoriais.
- **Corel SCREEN CAPTURE:** A maneira rápida e simples de capturar imagens do ecrã.
- **Bónus CD-ROM:** CorelDRAW 3.0 *software*, + 250 Fontes *True Type* e *Type 1*, 14 000 imagens *clipart*, exemplos para CorelCHART, Corel PHOTO-PAINT e CorelSHOW.

CorelDRAW é para todos – dos principiantes aos experientes!

laserdata, lda.

Rua Nina Marques Pereira, 24 - Loja C – 1500 LISBOA
Telefs.: 716-1001/2/3/4 – Fax: 716-3363

Grafiteria pub

Nº 20 – Dezembro de 1992

Spooler
MAGAZINE DOS UTILIZADORES PC



2 Informática em Notícia

4 O Conteúdo da Disqueta

6 Correio dos Leitores

7 Sysinfo

10 Informática Industrial

12 ABCMedic

14 Sabia que...

16 Ligando ao Unix

20 Limpa

22 Programação em C (IV)

28 Captum

30 Programar em Clipper (IV)

32 Tradutor (II)

35 Programar em Assembly (IV)

39 Fichas D. O. S.

O PROGRAMA «SÁBADO INFORMÁTICO»

Já aqui falámos do programa radiofónico «Sábado Informático» transmitido pela Antena 2 da RDP todos os sábados, cerca das 12 horas. Trata-se de um excelente programa da autoria de Gil Montalverne que, simpaticamente, nunca esquece a **Spooler**. Voltamos a referí-lo por dois motivos: em primeiro lugar, pela qualidade e actualização do noticiário técnico, bem como pelo carácter didático que tanto prezamos; depois, por considerarmos ser importante divulgar, junto dos nossos leitores, um meio de comunicação na área da informática normalmente ignorado pela grande maioria dos ouvintes das «rádios». É que a Antena 2 da RDP, pela natureza dos seus programas essencialmente culturais e pela música que transmite - a chamada «clássica» – não será das mais populares. Mas porquê não quebrarmos a rotina e deixarmos entrar, de quando em vez, um pouco da «outra música» em nossas casas? Para que os nossos leitores fiquem devidamente informados, segue-se uma lista dos transmissores da Antena 2 e respectivas frequências em FM (Mhz):

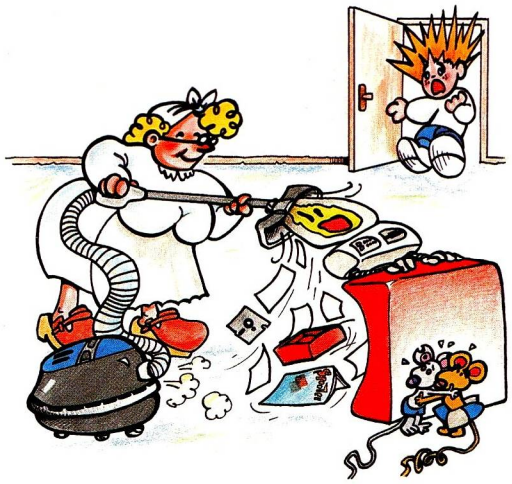
Valença:	99.0	Montejunto:	88.7
Braga:	91.3	Lisboa:	94.4
Bornes:	91.1	Santiago do Cacém:	90.7
Guarda:	88.4	Portalegre:	92.9
Porto:	92.5	Monchique:	91.5
Mendro:	91.1	Faro:	93.4
Lousã:	89.3		

E, no sábado seguinte a terem recebido a **Spooler**, não deixem de ligar os vossos rádios para uma destas frequências: ouvirão as novidades mais recentes do mundo da informática e, muito provavelmente, a apreciação do nosso amigo Gil Montalverne sobre o último número da vossa revista preferida.

A INFORPOR-92

Só visto! Para quem ainda tinha dúvidas, a afluência de assinantes, amigos, leitores e simples curiosos ao modesto *stand* da **Spooler** durante os quatro dias do certame, foi a consagração da nossa popularidade e a recompensa pelo esforço da equipa que produz a revista. Os nossos agradecimentos a todos quantos nos visitaram e apoiaram. E, como não podia deixar de ser, nesta quadra, aqui vos deixamos os nossos votos de feliz Natal e esperanças para um «93» onde a informática desempenhe um papel cada vez mais construtivo.

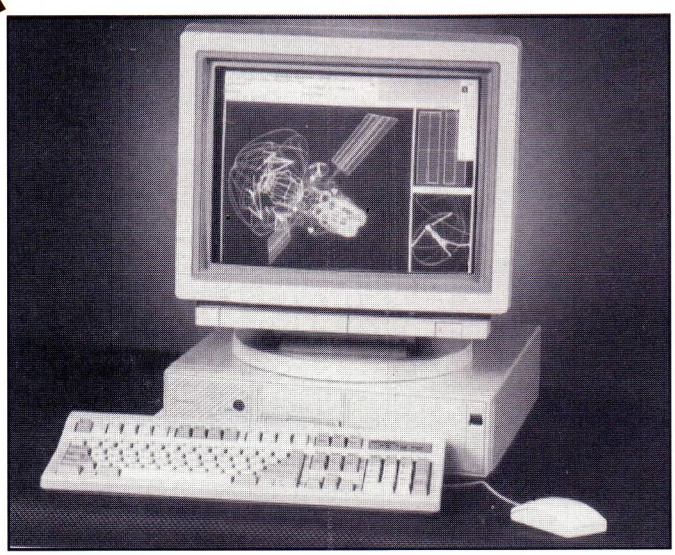
Renato Casquilho



Spooler

- **Propriedade:** Graficria, Publicidade e Artes Gráficas, Lda.
- **Director de Edição:** Franco Gomes ■ **Director**: Renato Casquilho
- **Redacção:** Franco Gomes, Paulo Bernardoe Renato Casquilho.
- **Colaboradores:** Carlos Ladeira, Cunha Brandão, José Orlando Pereira, José Páscoa, Paulo Bernardo, Vítor Monteiro.
- **Ilustrações:** Leonor Gomes
- **Relações Públicas e Publicidade:** Fátima Ribeiro
- **Assinaturas e encomendas:** Paula Martinez
- **Redacção, Publicidade e Serviços Administrativos**: Rua Duarte Pacheco Pereira, 69-A – Damaia – 2700 AMADORA – Telfs.: 475 22 89/474 60 11/41
- **Produção Gráfica:** Graficria, Publicidade e Artes Gráficas, Lda.
- **Impressão:** Empresa Litográfica do Sul, S.A. ■ **Distribuição:** Electroliber
- **Revista bimestral** ■ **Preço:** 550$00 ■ **SRIP N.º** 213900
- **Tiragem:** 12500 exemplares ■ **Dep. Legal N.º:** 28587/89

INFORMÁTICA EM NOTÍCIA



INTERGRAPH PC

A Intergraph Portugal anuncia a disponibilização no mercado nacional do Intergraph PC, um sistema informático baseado num processador Intel 486 a 33 MHz, que é fornecido com o MicroStation PC4.03, MS-DOS 5.0 e Microsoft Windows, devidamente instalados, permitindo ler directamente os ficheiros do tipo .DWG.
Configurado para trabalhar em redes de dados, o INtergraph PC está equipado com adaptador Ethernet, bem como com software TCP/IP, NFS e NOvell Netware.
Para utilizadores de MicroStation, a Intergraph possibilita a funcionabilidade gráfica do Intergraph PC, com monitores SVGA de 17", a cores, simples ou duplos.
Para além das suas características para funcionar autonomamente, o Intergraph PC é um excelente «gateway» para plataformas UNIX que «correm» software Intergraph.

NOVA LINHA DE COMPUTADORES PHILIPS «EVOLUTION LINE»



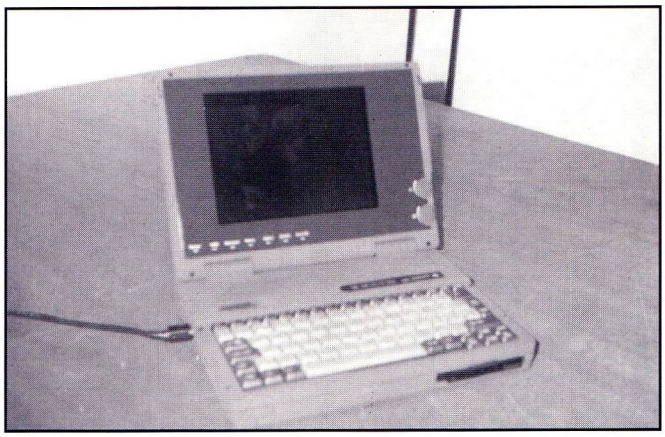
A Philips Portuguesa, S. A. apresenta a sua nova linha de Computadores Pessoais *Evolution Line*.
Como principal inovação, estes sistemas oferecem a possibilidade de trocar a placa do processador, transformando um sistema AT 286 ou 386 SX num sistema 486 SX ou um sistema 386 DX num 486 DX.
Esta característica deve-se ao desenho modular único apresentado pela Philips nos seus computadores pessoais.
O computador está internamente dividido em duas partes: uma, a da placa do processador e a outra onde se encontram os restantes componentes.
Este conceito vem permitir que o utilizador, particular ou empresa, possa manter o seu sistema actualizado tecnologicamente durante um período considerável, bastando, para o efeito, uma simples troca da placa do processador, *upgrade* de memória ou do disco rígido.

ENCONTRO TOPIS INTERNACIONAL



Com a finalidade de dar a conhecer a sua nova linha de produtos e a sua estratégia de vendas a TOPIS INTERNACIONAL, tal como já tinha acontecido no ano passado em Mira, promoveu a realização do seu Encontro Anual onde reuniu, para além de clientes, amigos e representantes da Comunicação Social, os seus corpos directivos, membros da delegação no Norte, a sua equipa de vendedores e todos os seus revendedores.
Conforne salientou a S.ª D. Isabel Neves Rosa, irmã do fundador da empresa e actual sócia-gerente, a TOPIS INTERNACIONAL tem estado atenta ao desenvolvimento operado na área do mercado dos microcomputadores e às exigências que tornaram os fabricantes mais conscientes de avanços rápidos, competitivos e tecnicamente qualificados.
É neste panorama – prosseguiu a gestora – que se observa a consolidação da marca TOPIS no mercado nacional, totalizando presentemente 34 000 unidades instaladas, o que representa, apenas durante os seis primeiros meses de 1992, um crescimento de 60% em relação a igual período do ano anterior.
A TOPIS INTERNACIONAL resolveu definir uma estratégia de qualidade dos equipamentos e de assistência ao cliente final que veio traduzir-se nesses números bastante significativos. Não é por acaso que essa estratégia se apoia em marcas conceituadas tais como a INTEL, a NOVELL e a SAMSUNG, tendo estado presentes no Encontro representantes desses verdadeiros *leaders* de mercado que se deslocaram especialmente a Portugal para apresentar as suas mais recentes inovações tecnológicas. Foi por exemplo o caso da Gilles Granier que apresentou o próximo processador da Intel, previsto para Fevereiro de 1993, até ao momento conhecido apenas por PS e que vai produzir uma verdadeira revolução naquela área.
A TOPIS apresentou as suas novas impressoras SAMSUNG de 9 e 24 agulhas, assim como as laser, detentoras de uma tecnologia apurada que lhes garante qualidade e duração indiscutíveis. A qualidade e a assistência ao cliente são aliás a tecla rnais difundida pela TOPIS que anunciou, pela voz do seu Director Técnico, Eng.º António Sampaio, as novas condições de venda de equipamentos, que incluem um contrato de manutenção a preço extraordinariamente acessível e cuja primeira anuidade, que se segue à tradicional garantia de um ano, é gratuita. Isto só é possível, salientou, porque a TOPIS tem absoluta confiança na qualidade técnica dos equipamentos que vende.
Foram igualmente apresentados os novos NOTEBOOK TOPIS/HALIKAN, o NBA 386SX/25 MHz e o NBC 386SL/25 MHz.
Quanto aos restantes computadores TOPIS, nas versões Slim-Line, Mini-Tower e Tower, equipados com os processadores 386SX/25, 386DX/25 e 33, 486SX/25, 486DX/33, DX2 e DX50, representam a continuidade numa linha que tem sido distinguida pela sobriedade, fiabilidade tecnológica e eficiência.
A TOPIS confirmou, mais uma vez, a sua promessa de garantir ao cliente final, para além da qualidade, uma assistência constante e eficiente.

«NOTEBOOK» COM MONITOR POLICROMÁTICO M-TEC

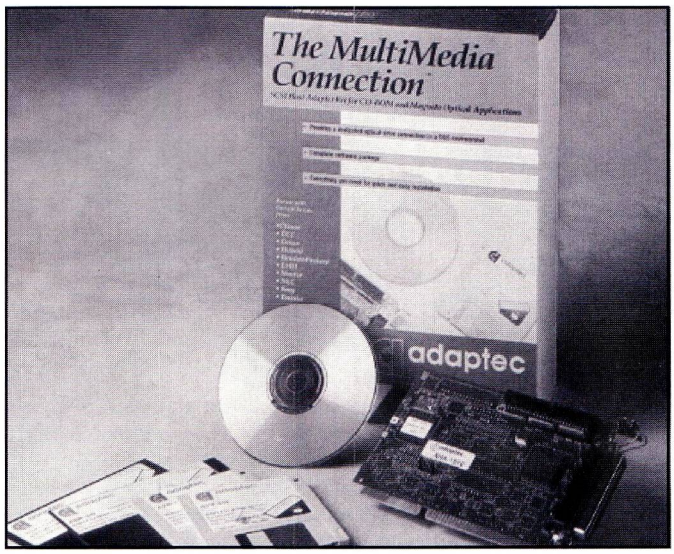


Fabricado em Taiwan foi recente anunciado, pela empresa M-TEC Expresso Lda., este «notebook» com monitor policromático.
Equipado com processador Intel 386SX-25, *drive* de disquetas de 3,5" - 1,44 Mb, 4 Mb de RAM expansíveis a 8 Mb, teclado português, ecrã de cristais líquidos retroiluminado, resolução de 640x480, portas série, paralela, para rato, para monitor e teclado externos.
O disco rígido é Seagate 80 Mb-15 ms.
Possui autonomia de 4 horas com bateria removível. Inclui bolsa de transporte.

DEFESA DO SOFTWARE LEGAL

A Associação Portuguesa de Sotware – cujos corpos gerentes, recentemente eleitos, estão a desenvolver uma actividade intensa – anuncia uma nova iniciativa, com a qual pretende afirmar positivamente o Software legal comercializado em Portugal.
Consciente de que o combate à pirataria do software não se pode fazer só através de acções directas de penalização dos infractores – como já aconteceu e continuará a acontecer, sempre que se veriquem situações que o justifiquem – ASSOFT tem vindo a tomar um conjunto de iniciativas de carácter informativo e formativo.
A afirmação do software legal, como produto de qualidade e isento de infracções, é uma das iniciativas da ASSOFT que – a pedido dos seus associados – emite selos de «software legal», a serem colocados nos produtos comercializados, como garantia da sua legalidade.

«ADAPTEC MULTIMEDIA CNNECTION»



A empresa Diode tem já disponível o *kit* «Adaptec Multimedia Connection» que inclui todo o *hardware* e *software* necessários para a ligação de periféricos CD-ROM ou magneto-ópticos via BUS SCSI.
O «Multimedia Connection» permite executar as mais recentes aplicações que combinam áudio, vídeo e texto. A par da mais elevada compatibilidade, comum a todos os *kits* da Adaptec, comprovada nos mais recentes modelos de CD-ROM e disco magneto-óptico com *interface* SCSI ou SCSI-2.
O *kit* completo inclui adaptador AT-SCSI AHA-1510, módulo de *software* ASW-410 para CD-ROM em DOS, ASPI (Advanced SCSI Programming Interface) ASW-1210 com suporte para disco magneto-óptico, manual de utilização e instalação e acessório de montagem.

COMPUTADOR DELL 486DX-33

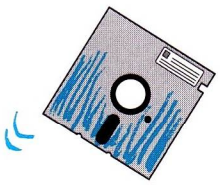
Preocupada em proporcionar aos utilizadores soluções informáticas de qualidade a baixos preços, a Década, anunciou o computador DELL 486DX a 33 MHz, disco rígido de 120 Mb, 4 Mb de RAM e controlador gráfico VGA - 256 cores, 1024x768 – 1 Mb RAM vídeo.
436 160$00 é o preço com monitor VGA monocromático; 472 120$00 com monitor policromático. Os valores apresentados já incluem 16% de IVA.

Graficria
Publicidade e Artes Gráficas, Lda.
Rua Duarte Pacheco Pereira, 69-A
Damaia-de-Baixo – 2700 AMADORA
Telefs.: 474 60 11/474 60 41 – Fax: 475 22 89

DIVULGASOFT, LDA.
FREEWARE – SHAREWARE – PUBLIC DOMAIN
Rua dos Barbosas, N.º 151 – Loja 22 – 4700 BRAGA
Telef.: 053 - 619244 Fax: 053 - 76159
A maior casa de SHAREWARE do País.
Tire vantagens da nossa enorme oferta.
Descubra o «MUNDO» do SHAREWARE
Preço por disquete com programa de SHAREWARE
Formato 3,5" = 600$00
Formato 5,25" = 500$00
Peça já a nossa disquete/catálogo
por apenas 250$00 + portes



O CONTEÚDO DA DISQUETA



Tal como anunciámos na Spooler Nº 19, os ficheiros presentes na disqueta deste número e nos que se seguem, deixaram de ser «autodescompactáveis». Isto significa que os nossos amigos leitores terão um pouco mais de trabalho. Mas também beneficiam com o facto dos programas ficarem com menor dimensão e, consequentemente, haver mais espaço disponível na disqueta. O utilitário que vai servir para descompactar os programas encontra-se na disqueta sob o nome de «LHARC.EXE». Este sim, é autodescompactável, pelo que basta copiá-lo para um directório do disco e digitar o seu nome para se obter o utilitário LHA.EXE. Depois, é simples: para descompactar qualquer um dos ficheiros com extensão .LZH, digite: LHA E <nome>. Por exemplo: LHA E ABCMEDIC. Para conhecer todas as opções do programa, digite só LHA: aparecerá um ecrã de instruções.







ABCMEDIC

Continuamos, nesta rubrica, a dar a conhecer aos nossos leitores os programas concorrentes ao Passatempo «Programas Didáticos», cuja classificação ficou situada entre os dez primeiros lugares. Cabe a vez ao ABCMEDIC, um interessante programa sobre medicina e cuidados de saúde.



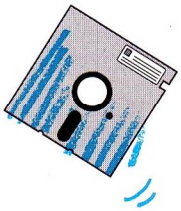
AGENDA2

O programa AGENDA2 é a passagem fiel do programa escrito em dBASE para Clipper. O objectivo é mostrar ao leitor as diferenças entre as duas linguagens. Por motivos de espaço na disqueta, não é possivel enviar o programa executável, pelo que é necessário proceder à respectiva compilação e ligação. Para isso, leia atentamente o artigo «Programar em Clipper – Parte IV» do nosso colaborador Victor Monteiro.



CALC4

No seu artigo «Programar em C – Parte IV», Carlos Ladeira oferece-nos esta nova versão da calculadora onde foram introduzidas algumas potencialidades. Entre elas destacam-se a inserção de variáveis (A a Z), funções de cálculo matemático e a possibilidade de gravar o conteúdo das variáveis num ficheiro, para que este possa ser utilizado na próxima execução do programa.



CAPTUM

O «pacote» de programas CAPTUM, da autoria de José Páscoa, é um conjunto de ferramentas que lhe permitirão capturar imagens em alta resolução EGA (640x350) ou VGA (640x480) até 16 cores e incorporá-las muito facilmente nos seus programas produzidos em Turbo C ou Turbo Pascal.



FUNCDATA

Trata-se do módulo FUNCDATA.PRG que, por lapso, não incluímos nas fontes do programa FERY publicado no anterior número da **Spooler**.

LHARC

O famoso compactador/descompactador *Shareware* de Haruyasu Yoshizaki, versão 2.13, que distribuimos gratuitamente aos nossos leitores.

LIMPA

«Uma das omissões de todas as versões do sistema operativo MS-DOS (incluindo a 5.0), é a capacidade de, com um único comando, remover directórios e/ou subdirectórios, sempre que estes não estejam vazios». É com estas palavras que Carlos Ladeira introduz este interessante utilitário, cujo desenvolvimento se encontra no respectivo artigo.

LINKER

José Nascimento Pereira presenteia-nos com uma «emulação» das capacidades do UNIX. Tal como nos diz no seu artigo: «O que vos proponho desta vez segue esta linha de acção, mas vai um pouco mais longe. A ideia é acrescentar uma nova capacidade ao sistema de ficheiros do DOS, que é de grande utilidade no Unix: **ligações simbólicas**, ou como são vulgarmente conhecidas, *links*.».

RELOGIO

«Trata-se do programa que serviu de base a todas as explicações que demos até aqui.» Este discurso é de José Páscoa, no seu artigo «Programar em *Assembly* – Parte IV». E acrescenta: «Obterá um simpático programa que faz uma quantidade realmente fabulosa de trabalho em apenas 137 bytes de código máquina. É esta uma das maravilhas do *Assembly*».

SYSINFO

Como nos diz o Hugo Ferreira: «Trata-se de mais um utilitário a juntar à longa lista dos ditos que a **Spooler** vem oferecendo aos seus leitores. Para além das informações facultadas sobre o nosso sistema, a «leitura» da *fonte* será uma «fonte» de preciosos conhecimentos na área do Pascal.

FUJITSU

Periféricos

Impressionado com as Matriciais Fujitsu



Pergunte pelas Impressoras Fujitsu

FUJITSU

Liberdade de Impressão

- Ampla gama em 24 agulhas cobrindo as necessidades mínimas e máximas.
- Velocidade de 180 a 505 caracteres/segundo. Alta fiabilidade com MTBF* até 8.000 horas.
- Impressão de formulários e impressos até 7 vias.
- Programação desde PC por menús em português.

(*) Tempo médio entre avarias

Porque na FUJITSU há uma impressora por cada nível de necessidade.
Porque a sua tecnologia lhe permite conseguir um produto silencioso, rápido e versátil. Tudo o que possa imaginar encontrará na gama FUJITSU.

Comprove-o no seu distribuidor e ficará interessado.

■ DIODE
Rua Rodrigo da Fonseca, n.º 72 - 4.º Dto. — 1200 LISBOA
Tels.: 386 41 57 / 65 - 386 31 58/64 • Fax: 386 09 87

DIODE
Grupo Getronics

Correio dos Leitores

Por Renato Casquilho

Gostaria de trabalhar com outro editor que não o do dBASE...A dúvida está em como fazer para testar o programa ou seja, qual o comando que devo fazer para obter o mesmo resultado que com «DO <nomeprograma>. Posso chamar com outro editor um programa feito com o editor do dBASE (Modify Command)?...
Tenho o programa Thedraw...como integrar um trabalho feito com este programa nos meus programas dBASE?

***Maria João** – Lisboa*

Os ficheiros .PRG criados pelo editor do dBASE (com o Modify Command) são ficheiros de texto ASCII. Como tal, podem ser criados com qualquer editor que grave em ASCII puro (pode também usar um processador de texto, mas deverá gravar em ASCII e com a extensão .PRG). Uma forma elegante de chamar outro editor consiste na utilização do ficheiro CONFIG.DB que deverá estar no mesmo directório do dBASE. Este ficheiro pode ser criado ou alterado em qualquer editor ASCII (ou com o próprio Modify Command) e pode conter múltiplos comandos destinados à configuração que pretendermos dar ao dBASE. Por exemplo, para substituir o editor do dBASE por outro, sempre que se chama Modify Command, deverá incluir a linha **TEDIT=E2** (se E2 for o nome do editor que escolheu). Também tem outra alternativa que nos parece mais aconselhável, a qual permite manter o editor do dBASE e chamar outro pela pressão de uma tecla de função: acrescenta a linha **f3="run e2;"**. E pronto, sempre que premir F2 corre automaticamente o seu editor. Como os programas são gravados com a extensão .PGR, sempre que fizer o comando **DO <programa>**, ele arranca como qualquer outro.
Quanto aos desenhos ou ecrãs construídos pelo Thedraw, nada mais simples. Grave o seu desenho no formato .COM. Coloque esse ficheiro no directório do dBASE. No seu programa, inclua a instrução **RUN <ficheiro>** (em que ficheiro será o nome com que o gravou no Thedraw). E pronto, o ecrã (ou desenho) aparecerá imediatamente. Sobre estes assuntos e outros não menos interessantes, sugerimos a leitura do livro «dBASE III Plus – Programação Avançada» que poderá obter através da «Montra Spooler».

... sei que entre o dBASE e o Lotus 123 se faz «transladação» de bases de dados, de um para o outro. Gostaria de saber se estas são possíveis entre o QPRO 3.0 e o dBASE III Plus e vice-versa...

***António Manuel** – Lisboa*

Como não possuímos e nunca trabalhámos com o 123 e o QPRO, pedimos ao nosso colaborador José Orlando Pereira para encontrar uma solução para este nosso leitor. Aqui vai a sugestão:
«O QPRO, além dos ficheiros normais, pode ler vários tipos de ficheiros de texto através da opção **Tools/Import...**, pelo que o problema consiste em transformar os .DBF em ASCII puro. Por passos, na linha de comando do dBASE e com a base de dados aberta:
a) Crie um ficheiro de relatório .FRM com o comando **Create From <ficheiro.frm>**. Neste ficheiro de relatório devem constar apenas as colunas de dados, sem cabeçalhos nem sumários. Deve-se pôr também o tamanho de página a 500 linhas (o máximo) e suprimir as margens.
b) Executar o comando **Report Form <ficheiro.frm> TO FILE <ficheiro.txt>**.
c) Sair para o DOS e com um editor de texto remover de **<ficheiro.frm>** todas as linhas que não correspondam a um registo. Este passo não é absolutamente necessário, mas simplifica.
d) Do QPRO usar **Tools/Import/ASCII Text** e especificar <ficheiro.txt>.
e) Escolher o primeiro registo e usar **Tools/Parse/Create**.
f) Fazer **Tools/Parse/Input** e escolher a *range* apropriada, incluindo a linha recém-criada pela alínea anterior. Atenção que é só a primeira linha que conta!
g) Fazer **Tools/Parse/Output** e escolher o destino da *range* transformada.
h) Finalmente, executar **Tools/Parse/Go** e pronto, já está!.
No sentido inverso, isto é, da folha de cálculo para a base de dados, não posso ajudar. Sublinho, apenas, que com o dBASE IV é bastante simples, em qualquer dos sentidos, pelo que o leitor talvez deva pensar numa actualização».

...o pedido é para alargar o programa COMPRIME com o compressor ARJ e com as opções que este nos pode dar...
... o meu computador é um Schneider VGA AT System 40...gostaria de saber quanto me custaria cada MB para aumentar a memória...preço da placa...etc....

***Alcino Alves** – Ermesinde*

O pedido deste nosso leitor pareceu-nos merecer uma resposta que pudesse ser extensiva aos muitos que se mostraram interessados no utilitário COMPRIME publicado no número 18 da **Spooler**. Por isso, pedimos ao Pedro Miguel, seu autor, que nos preparasse uma resposta:
«Sugeriram-me que o programa Comprime utilizasse o compressor ARJ (programa que eu não conhecia na altura). Por isso aqui vos apresento as modificações necessárias para colocar o programa Comprime a trabalhar também com o ARJ.
No ficheiro COM.BAT temos de adicionar na linha 46 (quadro do Menu) a linha: **echo. | 3 | ARJ - usa o compactador Arj |** de maneira a ficar igual às outras opções. E mudar a opção FIM para 4.
Na linha 54 temos de mudar o nº **51** para **52**. A seguir a esta temos de adicionar uma linha com o seguinte: **if errorlevel 51 goto arj1**
Na linha 65 adicionar uma linha com o seguinte: **:arj1 set prg=ARJ**
goto menu
Na linha 73 adicionar uma linha com: **if %prg%==ARJ echo * ARJ *** de maneira que fique centrado.
Na linha 102 adicionar uma linha com: **if %prg%==ARJ arj f -je %1**
Na linha 121 adiciona-se uma linha com: **if %prg%==ARJ arj e %1**
Mudar a linha 176 para: **if %prg%==ZIP pkzip -p -r %1 >nul** e adicionar a linha: **if %prg%==ARJ arj a -r -jm %1 >nul**
Mudar a linha 196 para: **if %prg% == ZIP pkzip -ex %1 >nul**
e adicionar a linha: **if %prg% == ARJ arj a -jm %1 >nul**
Mudar a linha 216 para: **if %prg% == ZIP pkzip -es %1 >nul** e Adicionar a linha: **if %prg% == ARJ arj a -m2 %1 >nul**
Na linha 238 adicionar uma linha com: **if %prg% == ARJ arj l -jp %1.arj**
E, por fim, apaga-se a linha 49. Feitas estas modificações o programa ARJ estará pronto a funcionar. Na opção de Autodescompactação não é preciso criar um ficheiro .arj para depois o passar para EXE, pois o programa ARJ vai comprimir a directoria e tornar automaticamente o ficheiro auto descompactável.»
Quanto aos preços que nos pede, a nossa resposta terá de ser a mesma que temos dado para questões semelhantes: dirija-se à empresa que lhe vendeu a máquina ou ao seu representante.

... estando a preparar um utilitário em Pascal que recorria a algumas interrupções do DOS, deparei com um problema: algumas interrupções retornavam valores bos bits de bytes ou «words» (8 ou 16 bits).... como consigo ter acesso aos bits desses bytes? Como saber o conteúdo dos bits?

***Pedro Andrade** – Barreiro*

A resposta a este nosso leitor foi dada pelo Rui Anastácio, um dos nossos colaboradores na área do Pascal. Por a considerarmos de interesse geral, aqui a reproduzimos:
«Se convertermos um byte ou *word* para binário, podemos verificar facilmente quais os bits ligados ou desligados. Por exemplo: o valor 58 escrito em binário é 000111010. Facilmente verificamos que os bits 1,3,4 e 5 (contando da direita para a esquerda e começando em 0) estão ligados (1's) e os restantes bits estão desligados (0's). Dependendo da posição do bit dentro do byte, este terá um valor que corresponde a uma potência de 2. Assim o bit 0 terá o valor de 2^0=1 (^ corresponde à potência), o bit 1 o valor 2^1=2, o bit 2 o valor 2^2=4, etc. Podemos construir um *array* contendo os valores dos bits o que nos facilita um pouco a vida:
CONST bit: Array[0..7] of byte = (1, 2, 4, 8, 16, 32, 64, 128);
O truque para a manipulação dos bits está nas operações AND e OR.
Vamos imaginar que temos a variavel B (byte) que tem um valor qualquer.
Para verificar se um bit está ligado basta fazer um AND de B com o valor do bit. Se o resultado for igual a 0 então o bit está desligado; senão está ligado.
IF (B AND bit[6])<>0 { verifica se o bit 6 do byte B está ligado } **THEN** { o bit ESTÁ ligado } **ELSE** { o bit NÃO está ligado }
Para desligar fazemos um AND de B com o valor máximo do byte (255) menos o valor dos bits que queremos desligar. Assim:
B:=(B AND (255-bit[6])); { Desliga o bit 6 } **B:=(B AND (255-bit[6]-bit[7]);** { Desliga os bits 6 e 7 }
Para ligar um bit fazemos um OR com o valor do bit a ligar.
B:=(B OR bit[6]); { Liga o bit 6 }
B:=(B OR (bit[6]+bit[7]); { Liga os bits 6 e 7 }
Por vezes temos de extrair certos bits de um byte. Por exemplo: uma determinada interrupção devolve nos bits 3 e 4 o nº de *drives* do sistema. Neste caso teremos que deslocar os bits para a direita e depois desligar os restantes que não interessam. Se o byte fosse por exemplo 180 (10110100 em binário) teriamos: **B:=B SHR 3;** { SHIFT RIGHT - deslocação dos bits para direita } { B=00010110 - 3 0's inseridos do lado esquerdo } **B:=B AND (bit[0]+bit[1]);** { desliga todos menos os 2 primeiros } { 00010110 -> B com os 2 bits que interessam à direita AND 00000011 -> **bit[0]+bit[1]** – 00000010 = 2. E assim teríamos o valor 2 em B que corresponderia ao nº de *drives* do sistema».

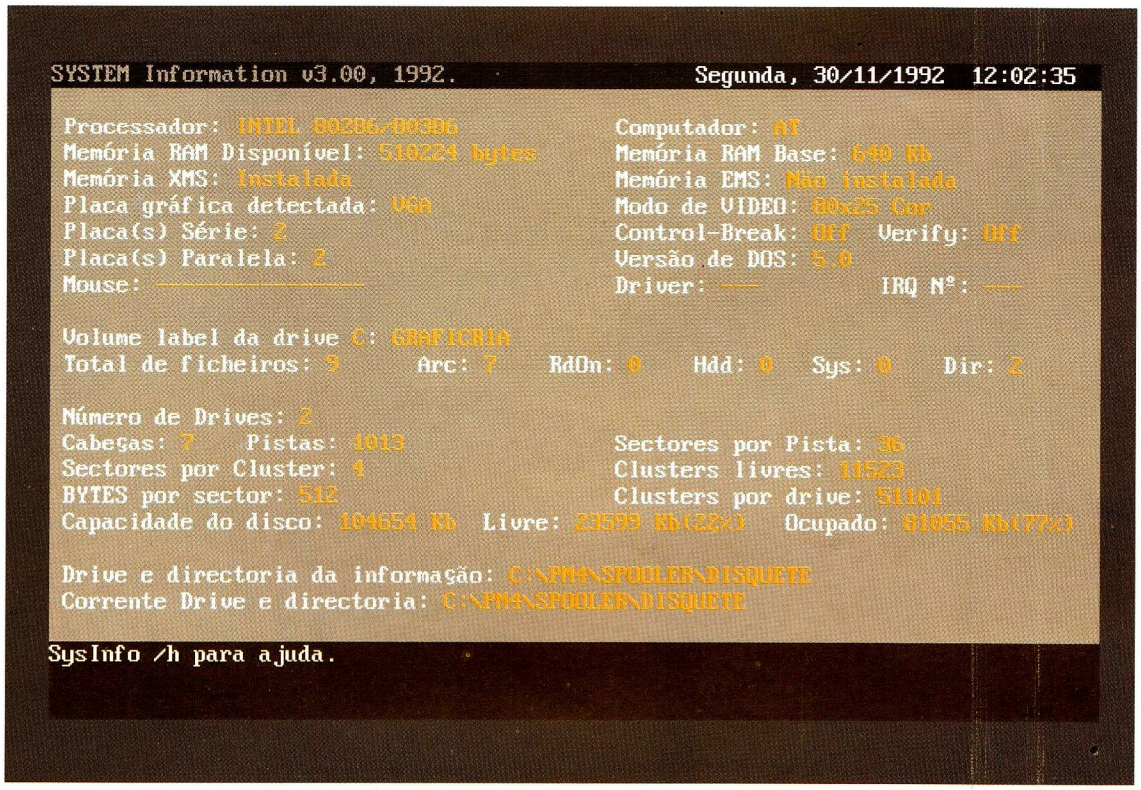
...na Spooler Nº 19 vem um programa ANIMA...quando sigo as instruções do autor aparece-me a seguinte mensagem: «Erro fatal: Inicialização de gráficos» Serão capazes de me dizer o que isto significa e o que fazer?

***José M. Lopes Pires** – Amadora*

É o próprio Rui Anastácio, autor do programa, que responde:
«O erro deve-se ao facto do programa não encontrar o ficheiro que contém a informação sobre a placa gráfica existente. O ANIMA vai procurar no directório corrente um dos ficheiros com extensão .BGI. Se não o encontrar sai com um erro de inicialização. Para solucionar o problema basta copiar os ficheiros com extensão .BGI para o directório onde se encontra o ANIMA. Se o leitor tiver uma placa VGA ou EGA, deverá copiar o ficheiro EGAVGA.BGI.»

SYSINFO

COLABORAÇÃO DO LEITOR



Como nos diz o Hugo Ferreira, trata-se mais um utilitário a juntar à longa lista dos ditos que a Spooler vem oferecendo aos seus leitores. Para além das informações facultadas sobre o nosso sistema, a «leitura» da fonte será uma «fonte» de preciosos conhecimentos na área do Pascal.

SYSINFO é mais um programa a acrescentar ao grupo dos «utilitarios», publicado pela **Spooler**. Não é um programa original, nem nada parecido com isso. Baseie-me em muitos «Demos» e Listagens, para fazer um programa que eu tentei que tivesse qualidade e agora vos apresento.

SYSINFO edita 40 informações sobre o sistema, desde o número de portas série e paralelas, até ao número de *clusters* no disco.

Para conseguir todas estas funções, usei ordens convencionais do Pascal e tambem tive de recorrer às tão faladas interrupções.

Bem, mas vamos deixar a conversa e passarmos ao que interessa. De seguida explico, uma por uma, cada função do programa:

Dia e Hora:
Aqui é editado o dia e hora correntes do computador. São usadas as funções GetDate e GetTime para o efeito. Utiliza-se a função LeadingZero para tranformar a hora numa cadeia e caso o número seja só um carácter, acrescenta-lhe um zero à esquerda.

Processador:
Determina o tipo de processador através de um valor Hexadecimal na memória.

Computador:
Através do mesmo endereço na memória determina o tipo de computador (PC/XT/AT).

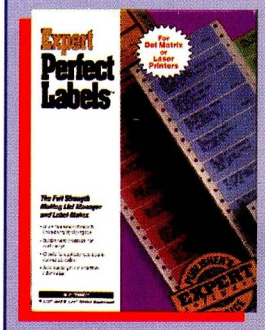
Memória RAM disponível:
Retorna o tamanho da maior área disponivel na memória.

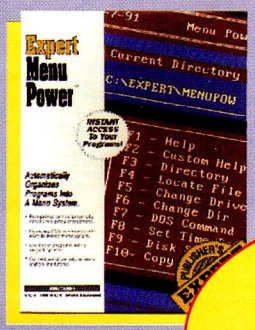
Memória Base:
Este é o primeiro exemplo da utilização de interrupções. Através de uma chamada à interrupção 12h, retorna a dimensão da memória real através do registo AX, pelo que podemos ter acesso à memória base do computador.

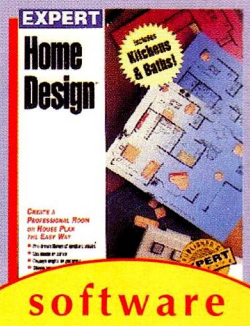
Memória XMS:
Retorna «instalada» se a memória XMS estiver instalada ou «não instalada» em caso contrário. Utiliza-se aqui a int. 2Fh (interrupções múltiplas do DOS), e a função 43. Caso o retorno AX tiver o valor 80h, a e**X**tended **M**emory **S**ystem está instalada.

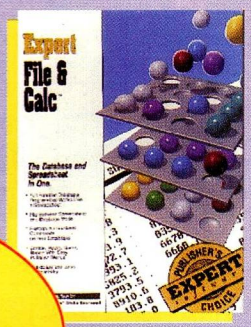
Memória EMS:
Retorna tal como no exemplo anterior o estado desta memória, (Instalada/Não instalada). Utilizamos uma função booleana para verificar se a memória está ou não instalada. A função após chamar a int. 21h, função 35h (retorno do endereço da interrupção (neste caso 67h –

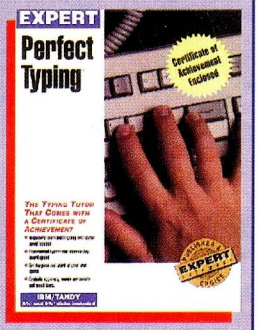
“SYSINFO edita 40 informações sobre o sistema, desde o número de portas série e paralelas, até ao número de clusters no disco”











software
EXPERT
P.V.P. 3.500$00
IVA Incluido

software
EXPERT
P.V.P. 4.100$00
IVA Incluido



IOMEGA
P.V.P. 75.400$00
IVA Incluido





TDK MF-2HD
CAIXA DE 10
P.V.P. 2.400$00 IVA Incluido



JVC - CAIXA DE 10
P.V.P. 2.300$00 IVA Incluido

SERIE
Sociedade de Exportação, Representação e Importação de Electrónica, Lda.

Rua Nunes dos Santos, 21 - 2º, sala 30 • São Pedro do Estoril • 2765 ESTORIL
Telefs.: 468 28 28 • 467 28 94 • Fax. 468 25 85

À VENDA NA LOJA SPOOLER

controlador de memória expandida LIM), vai ler os sete caracteres seguintes ao endereço da int., e caso os caracteres sejam iguais à assinatura da memória, então a memória expandida encontra-se instalada.

Placa Vídeo Detectada:
Retorna a placa vídeo detectada pelo programa. Utiliza-se a função DetectGraph para o efeito.

Modo de Vídeo:
Retorna o modo de vídeo em que o programa está a trabalhar. É utilizada a int. 11h (retorna a informação interna do *Hardware*).
Através do valor de AL tem-se acesso ao modo de vídeo exitente.

Placa(s) série:
Retorna o número de placa(s) série do computador. É utilizada também a int. 11h, mas é através do valor de AH que se detecta o número de portas.

Placa(s) paralela(s)
A sua existência é determinada exactamente da mesma maneira, assim como o Número de *drives*.

Control Break:
Retorna o estado de Control-Break.

Verify:
Retorna o estado de Verify.

Versão de DOS:
Retorna o número da versão do DOS. Utiliza-se a função DosVersion.
O primeiro byte é o valor mais baixo da versão e o segundo o mais alto: por exemplo, DOS 5.0, o primeiro byte será 0, e o segundo 5.
Cabe ao programador inverter os números para conseguir a versão do DOS correcta.

Mouse, Driver, IRQ:
Retorna o tipo de mouse utilizado, a versão do *driver* e o IRQ. Esta função tem que ser activada na linha de comando com o parâmetro /M. Resolvi deixar a utilização desta função por conta do utilizador do programa, pois alguns utilizadores poderão ter *drivers* do mouse já bastante antigos e, neste caso, o programa poderá produzir um erro.
Se o teste ao rato fosse obrigatório, seria bastante mau que o programa desse erro, cada vez que fosse chamado. Para obter estas 3 funções utilizei a interrupção 33h (interrupção de rotinas do mouse), função 24h (Para mais informações ler os artigos da **Spooler** «História e programação dos roedores Informáticos», Nºs 15, 16, 17).

Volume Label:
Retorna o *volume label* da *drive*. Esta função nem sempre resulta a 100%, mas é a única maneira que eu conheço de ter acesso ao dito.
Utilizei a função FindFirst(VolumeID).

As seis funções que se seguem de contagem de ficheiros foram possíveis com a utilização de FindFirst e FindNext.

Número de *drives*:
Tal como para saber o número de portas da máquina, para saber o número de drives também utilizei a int. 11h.

Cabeças:
Pode-se aceder a esta informação através da int. 13h (Serviços de processamento de disco no ROM BIOS). O valor encontra-se no registo DH.

Pistas:
Este valor é retornado após se descodificar o registo CX.

Sectores por pista:
Outro valor da mesma interrupção (13h), retornado após descodificar novamente CX.

Sectores por Cluster:
Aqui utilizei a int. 21h (Funções múltiplas do DOS), função 36h (retorna o espaço em disco); quando chamada, esta função vai retornar o espaço livre em disco, só que não será em bytes como estamos habituados a ver nos programas ou no Diskfree ou DiskSize do Pascal: o valor livre de um disco será retornado em *clusters*, cabendo ao programador codificar os valores retornados a fim de obter o valor livre em bytes. A interrupção retorna os valores da seginte maneira:

AX: sectores por *cluster*
BX: *Clusters* livres
CX: Bytes por sector
DX: *Clusters* por *drive*

Para obter em bytes com estes valores o espaço livre no disco, tem que se fazer os seguintes cálculos:

```
RegBX:= BX; (RegBX tem que ser do LONGINT)
EspLivre = (AX * CX) * RegBX;
```

Para obter o tamanho do disco:

```
RegDX:= BX; (RegBX tem que ser do LONGINT)
TamDisco = (AX * CX) * RegDX
```

Caso se queira estes valores apenas para informação, tal como no Sysinfo, basta utilizar os valores tal como estão nos registos sem qualquer alteração.

Capacidade do disco:
Retorna o tamanho total em KBytes do disco a ser examinado. Função do Pascal DiskSize.

Livre:
Retorna o espaço livre em KBytes do disco a ser examinado. Função do Pascal DiskFree.

Ocupado:
Retorna o espaço ocupado por programas no disco, em KBytes. Resultado do cálculo DiskSize-DiskFree.

***Drive* e directório da informação:**
Retorna a *drive* e directório a que a informação dada corresponde. Função do Pascal GetDir.

***Drive* e directório correntes:**
Retorna a *drive* e o directório de onde o programa foi chamado.
Função do Pascal GetDir.

Outros dados:
O programa tem incluída uma rotina que permite ver o modo de vídeo em que trabalha (COR/MONO) e uma outra que permite sair com uma mensagem de erro caso os parâmetros inseridos não sejam correctos.

Hugo Ferreira

INFORMÁTICA INDUSTRIAL

SISTEMAS DE SUPERVISÃO, CONTROLO E AQUISIÇÃO DE DADOS

Iremos apresentar, numa série de artigos, uma visão do mundo da Informática Industrial, focalizando nos sistemas de *software* de aplicação para Supervisão, Controlo e Aquisição de Dados de Processo (tipo SCADA).

Será feita uma caracterização funcional destes produtos de *software* baseados em microcomputadores, será abordado o ambiente de *software/hardware* que pode garantir uma operação em tempo real da aplicação e será descrita a engenharia de *software* associada ao desenvolvimento de um sistema tipo SCADA.

Concluiremos esta série de artigos apresentando rotinas de *software* que ilustrem a implementação de algumas tarefas críticas deste tipo de aplicações.

Ficamos, neste primeiro artigo, com uma introdução geral aos sistemas tipo SCADA, deixando para o próximo uma caracterização mais detalhada das funções destes sistemas.

INTRODUÇÃO

Os Sistemas de Informação Industriais podem ser apresentados em quatro níveis funcionais de acordo com a seguinte hierarquia: Gestão da Empresa, Gestão da Produção, Supervisão e Operação directa do processo produtivo (figura 1).

Estes sistemas têm-se desenvolvido por uma informatização baseada em sistemas digitais com uma estrutura distribuida que possa incluir equipamento analógico pré-existente, e que ofereça a cada nível hierárquico uma integração das funções relevantes. A utilização, cada vez mais generalizada, de sistemas digitais tem como motivação a redução de custos por aumento da automatização e integração de funções, com ganhos de flexibilidade e de qualidade, e permitindo a inclusão de novas funções em sistemas de informação, abrindo novas perspectivas à gestão industrial.

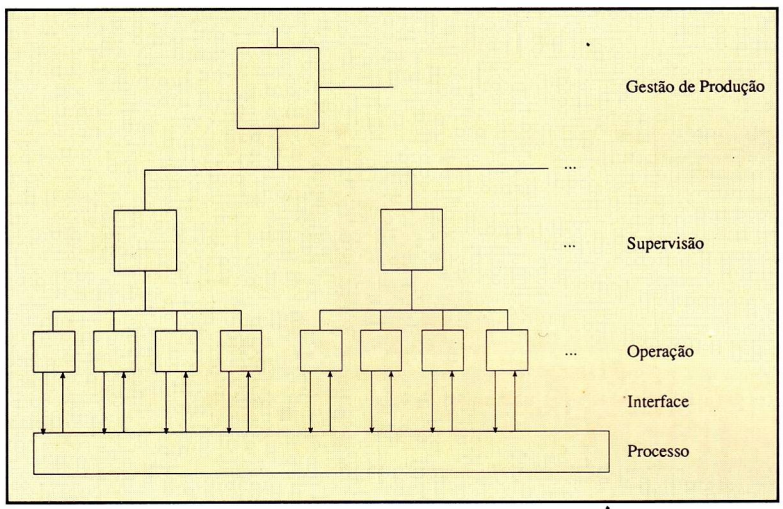


Figura 1 – Estrutura típica para um Sistema de Informação Industrial

Aos níveis hierárquicos considerados podem corresponder tipicamente os seguintes sistemas digitais: *mainframe*, minicomputador, microcomputador e controladores. Um sistema de informação industrial pode incluir como principais componentes: sensores que asseguram a interface de medida entre o sistema e o processo; actuadores que convertem instruções de comando em acções efectivas a nível do processo; interfaces de *Input/Output* que asseguram a ligação entre o sistema informático e os sensores e actuadores; equipamento de comunicações que interligam os vários subsistemas envolvidos; equipamento informático que inclui os computadores utilizados nos diferentes níveis hierárquicos; o *software* de sistema, de desenvolvimento e de aplicação que, conforme o nível considerado, terá de oferecer características específicas

Os sistemas de *software* de aplicação para Supervisão, Controlo e Aquisição de Dados (tipo SCADA - *Supervision, Control and Data Aquisition)* cobrem funções aos níveis da supervisão de processo e da operação directa de equipamento.

OS SISTEMAS SCADA

As funções implementadas por este tipo de *software* podem incluir: Monitorização do processo e registo da sua evolução; Controlo automático e Operação manual; Análise, Optimização e Supervisão do processo; Comunicações e Integração no sistema de gestão da produção.

Este tipo de *software* de aplicação caracteriza-se fundamentalmente pela necessidade de traduzir o comportamento real de um processo e de possibilitar intervenções na evolução desse processo, de acordo com restrições de tempo que caracterizam a sua dinâmica, de forma a manter o comportamento em conformidade com as especificações existentes.

O *software* tem que ser capaz de gerir, em tempo real, tarefas simultâneas que têm que ser tratadas em tempos pré-definidos e sincronizadas de acordo com prioridades.

O funcionamento em tempo real do *software* de aplicação depende também de todo o sistema de *software/hardware* que o suporta, incluindo o sistema operativo, a linguagem de programação, as características da base de dados eventualmente usada, o sistema de comunicações e a arquitectura de *hardware*.

Uma outra característica dos sistemas SCADA, que representa uma restrição forte, é a exigência de fiabilidade que os processos industriais pressupõem. A fiabilidade, aqui, significa uma baixa probabilidade de paragem do sistema, já que, em geral, pode ser aceitável alguma degradação no seu funcionamento.

Pode assim ser necessário prever no projecto de *software* formas alternativas de funcionamento (equipamento de emergência, operação manual) e procedimentos de recuperação de erros ou falhas.

Por outro lado, é necessário ter em conta que a aplicação é para ser usada em ambientes industriais, sujeita a inter-

venção de técnicos e operadores, e que as funções que executam podem ser críticas pelas consequências humanas, económicas e ecológicas associadas, áreas cada vez mais regulamentadas e fiscalizadas.

Pelas suas características, o *software* SCADA é muito dependente do sistema envolvente e, pela complexidade que pode atingir, pressupõe um processo de desenvolvimento rigoroso.

CARACTERÍSTICAS E FUNÇÕES

Tipicamente, os produtos comerciais deverão ser configuráveis, interactivamente, para um dado processo e sistema de *hardware* e incluir então dois subsistemas bem distintos, um correspondente a desenvolvimento e outro a exploração.

O subsistema de desenvolvimento oferece todas as facilidades para configurar o funcionamento da aplicação em tempo real. Deverá ser independente, não sendo necessário existir instalado após completada a configuração, libertando recursos e tornando mais económica uma utilização multiposto. As configurações poderão ser especificadas em programas modulares e independentes, de preferência normalizados ou de uso generalizado (ex: editor gráfico,*package* estatístico, gerador de relatórios, etc.), originando uma base de dados de configurações a serem usadas na fase de exploração.

Num sistema aberto deverá ser oferecido um ambiente para acesso à base de dados e para produzir programas de ligação *(drivers)* para comunicação com equipamento de aquisição e controlo. As configurações devem prever a possibilidade de integrar programas de utilizador para execução de diferentes funções.

As configurações da aplicação ao processo devem incluir os seguintes aspectos: interface com o operador, aquisição e controlo, alarmes, análises, relatórios, *hardware* instalado.

O subsistema de exploração implementa as funções do SCADA durante a operação em tempo real, de acordo com o especificado nas várias configurações pré-existentes.

As funções típicas de um sistema SCADA incluem:

- aquisição de dados do processo, com eventual arquivo em disco;
- visualização de quadros sinópticos representativos das várias partes do processo;
- análise dos dados recolhidos, automática ou manual;
- monitorização e tratamento de situações de alarme;
- operação manual do processo;
- controlo digital directo;
- optimização do processo;
- supervisão do processo: interacção com os controladores locais;
- comunicação com equipamento local no processo e com computadores de níveis hierárquicos iguais ou superiores;
- emissão de relatórios sobre a aplicação e o proces -so;
- segurança e protecção na utilização da aplicação;
- ajudas e informações ao utilizador;
- exportação de dados de exploração para tratamento *off-line* em aplicações de uso comum.

Eduardo Luis Cardoso

INVISTA NA TECNOLOGIA QUE IMPRESSIONA

Sim, a tecnologia FUJITSU impressiona. As novas impressoras DL900 e DL1100 de 24 agulhas pensam em grande. Imprimem em A3 terminando de vez com as limitações que a maioria das impressoras A4 tradicionais. De facto são ligeiramente maiores do que o papel onde imprimem. Não podem ser culpadas de lhe roubar espaço de secretária. Apesar do tamanho possuem a maior memória existente em impressoras de secretária. 24 KBytes mais precisamente podendo armazenar 11 páginas típicas de dados. Graças ao programa DLMENU em português, programá-las do seu PC é extremamente simples e cómodo. Com tudo isto, é uma pena se gasta dinheiro em impressoras menos avançadas. Principalmente se pode ter a sua FUJITSU DL900 por 72 000$00 a DL1100 por 92 000$00 e por 115 000$00 pode ter a DL1100 cor (preços sem I.V.A.). Consulte o seu revendedor autorizado fujitsu e conheça a tecnologia que impressiona.



GARANTIA 2 ANOS

FUJITSU

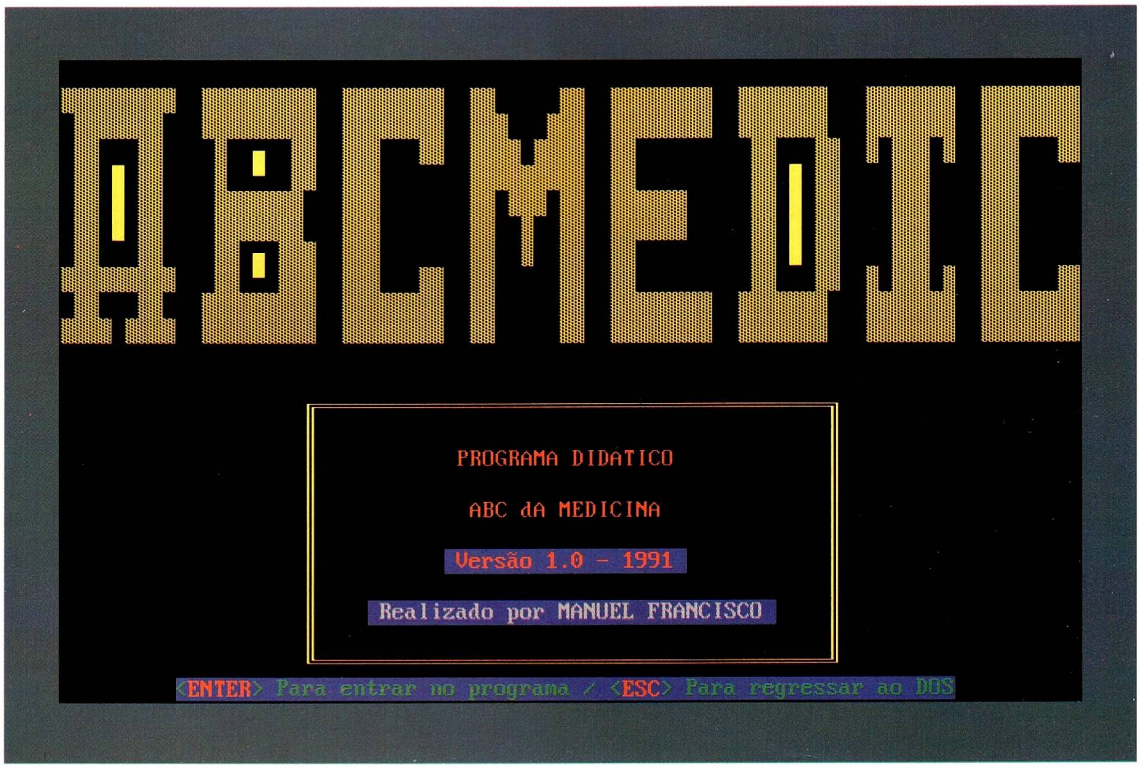
À VENDA NA «MONTRA SPOOLER»

DIODE Grupo Getronics

Rua Rodrigo da Fonseca, n.º 72- 4.º - Dto. – 1200 LISBOA – Tels. 386 41 57 - 386 41 65 - 386 31 58 - 386 31 64 • Fax 386 09 87

COLABORAÇÃO DO LEITOR

ABCMEDIC



Continuamos, nesta rubrica, a dar a conhecer aos nossos leitores os programas concorrentes ao Passatempo «Programas Didáticos», cuja classificação ficou situada entre os dez primeiros lugares. Cabe a vez ao ABCMEDIC, um interessante programa sobre medicina e cuidados de saúde. O autor, Manuel Francisco, dá-nos conta do que é o programa e do seu funcionamento.

> O programa foi escrito em dBASE III Plus, com recurso a alguns comandos do Clipper que o dBASE não suporta

O programa **ABCMEDIC** surge no âmbito do último concurso, «Programas didáticos», lançado pela nossa revista **Spooler**.

Sendo um iniciado em informática, com grandes lacunas e limitações, aceitei o desafio proposto, deitei mãos ao computador, passando largas horas em diálogo com esta máquina terrível, que parece ter sempre razão.

Trabalhei dentro de uma área que domino com mais facilidade do que a informática, por razões profissionais, e eis que surgiu o **ABCMEDIC** para um concurso.

O programa foi escrito em dBASE III Plus, com recurso a alguns comandos do Clipper que o dBASE não suporta (os ficheiros fonte não funcionam em dBASE de uma forma interactiva). Tenta ser um programa de consulta, muito simples, abordando alguns temas da medicina de que todos nós já ouvimos falar. Faz parte do programa uma base de dados, com a finalidade de arquivo de termos médicos.

CONSTITUIÇÃO DO PROGRAMA

O programa é constituído pelos seguintes ficheiros:

- **Abcmedic.prg:** módulo principal do programa
- **Help1.prg**: módulo com informação de ajuda
- **Cuid.prg**: módulo com informação sobre cuidados corpo
- **Regime.prg** e **nutri.prg**: módulos com informação sobre nutrição
- **Actua.prg, actua1.prg, prev.prg, socorrer.prg** e **trauma.prg**: módulos com informação sobre acidentes
- **Doenca.prg**: módulo com informação sobre doenças
- **Int.prg**: módulo com as introduções das várias opções
- **Menu0.prg**: módulo que gera o ecrã para a entrada de dados no glossário – **Cons.prg**: módulo para consulta da base de dados
- **Apaga.prg**: módulo para eliminar registos da base de dados
- **Imprime.prg**: módulo para impressão de registos da base de dados
- **Actual.prg**: módulo para actualizar registos da base de dados
- **Rotina.prg**: módulo que contém algumas funções e procedimentos.

Estes módulos foram compilados pelo Clipper dando origem ao executável ABCMEDIC.EXE. Faz parte do programa uma base de dados – MEDICINA.DBF – para arquivo dos registos dos termos médicos, um ficheiro indexado por ordem alfabética – TEMAS.NTX – e um *batch file* para inicializar o programa – ABC.BAT.

FUNCIONAMENTO

O programa funciona tanto em monitores monocromáticos como policromáticos, identificando-os no início do ficheiro **abcmedic.prg**.

Quando se corre pela primeira vez o programa, será

criado o ficheiro indexado TEMAS.NTX. No caso da base de dados MEDICINA.DBF não se encontrar no mesmo directório do ABCMEDIC.EXE, será pedido o directório em que se encontra, sendo copiado de seguida para o directório corrente.

O funcionamento do programa é bastante simples, havendo um diálogo permanente com o utilizador sempre que se tem que tomar uma decisão. Após o programa carregado, é pedida a confirmação para entrada no menu principal, bastando teclar ENTER para entrar ou ESC para regressar ao DOS. Após a entrada no menu principal, existem várias opções de consulta. Para o conseguir existem 2 opções:

1 – Mover a barra que aparece posicionada sobre a opção HELP, com as teclas do cursor até à posição pretendida e digitar ENTER.

2 – Digitar a letra inicial, que aparece em evidência no início de cada opção. Esta forma, como é evidente, é a mais rápida.

Na opção CUIDADOS CORPO, pela impossibilidade de encontrar termos com letras diferentes, a última opção não funciona pela pressão da letra inicial, tendo que se recorrer às teclas do cursor. Este foi um dos problemas que me surgiu e que não consegui ultrapassar, escolhendo uma outra letra, mesmo que não fosse no início da palavra.

Outro problema que surge e que não consegui ultrapassar dentro dos conhecimentos que tenho de dBASE e Clipper foi o seguinte: percorrendo o menu com as teclas do cursor, a letra que aparece em alta intensidade ou a vermelho, deixa de ter estas características. Desde já aceito sugestões dos vários leitores para ultrapassar este problema.

SIGNIFICADO DAS MENSAGENS

Aparecem 3 tipos de mensagem ao longo do programa:

1 – Mensagens informativas: Informam o utilizador do conteúdo de cada opção nos vários menus, aparecendo em rodapé.

2 – Mensagens que obrigam à digitação de uma tecla para sair ou entrar, confirmar, etc.

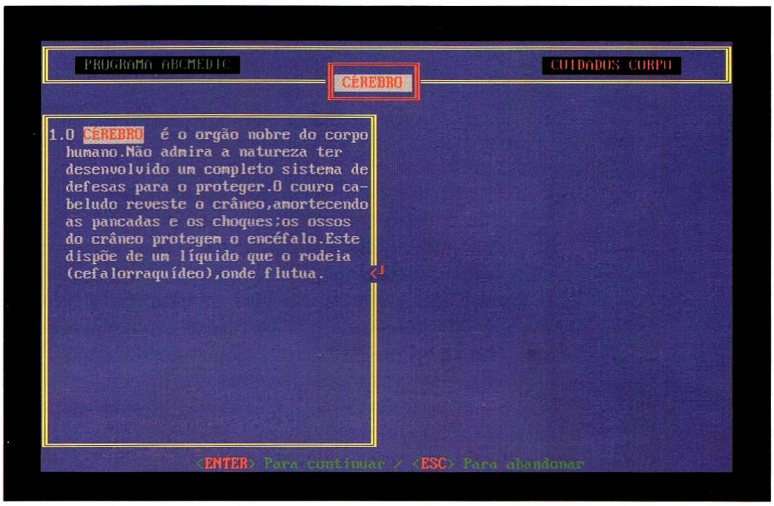
3 – Mensagen sonora: é emitido um sinal sonoro sempre que é pedida a digitação de uma tecla, e essa tecla não corresponde às alternativas dadas. A mesma mensagem é ouvida quando a consulta de determinada informação terminou.

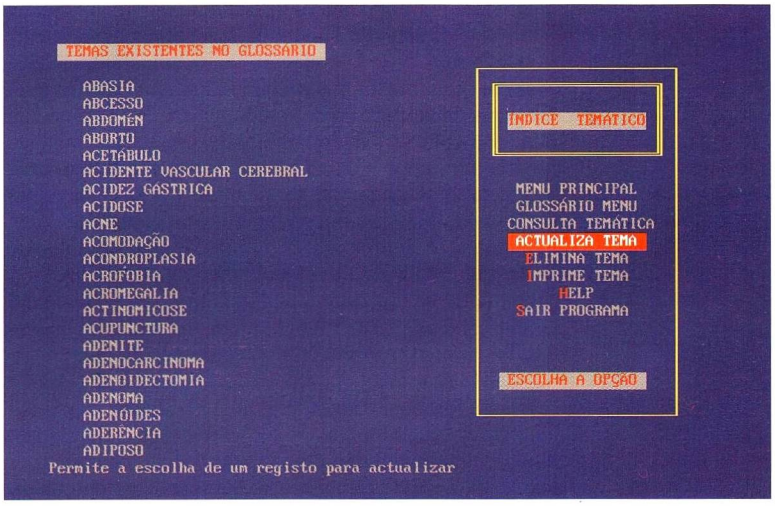
Quando estamos a consultar uma opção de texto, surge a mensagem «<ENTER> Para continuar/<ESC> Para Abandonar»; premindo <ENTER> o texto continua a ser mostrado; premindo <ESC> regressamos ao menu anterior. A partir do momento em que se ouve o sinal sonoro associado com a palavra «FIM», premindo <ENTER> volta-se ao menu anterior, premindo <ESC> volta-se ao menu principal.

OPÇÕES «MENU PRINCIPAL»

No menu principal aparecem várias opções de selecção. A seguir apresento uma informção resumida do seu conteúdo:

1 – HELP: Acesso a um pequeno ficheiro onde se encontra a informação elementar para se entender o funcionamento do programa.





2 – CUIDADOS CORPO: Principais cuidados a dispensar no dia a dia aos principais orgãos do corpo humano, de forma a mantê-lo saudável e livre de doenças.

3 – NUTRIÇÃO: Princípios da nutrição humana, tendo em consideração os vários nutrientes e as várias fases da vida humana. Contém alguns conselhos para se praticar uma alimentação racional.

4 – ACIDENTES: Prevenção de acidentes e algumas medidas a tomar no caso de eles se verificarem.

5 – DOENÇAS: Alguma informação sobre doenças, desde as que matam, as consideradas graves, as que incapacitam e as doenças sem gravidade.

6 – GLOSSÁRIO: Acesso à base de dados MEDICINA.DBF para criar, consultar, actualizar, eliminar ou imprimir os termos médicos contidos na base.

7 – SAIR: Abandona o menu principal, pedindo confirmação de abandono, para regressar ao DOS.

Este programa constitui, assim, um pequeno contributo que a **Spooler** me proporcionou. Espero também, começar aqui um diálogo com outros leitores na troca de experiências, de ideias, de críticas, através duma simples carta. Aguardo as vossas sugestões.

Manuel Francisco

Sabia que...

... a variável de ambiente PATH não pode conter mais de 127 caracteres?

... em situações específicas de trabalho, onde a PATH ultrapassaria os 127 caracteres, o DOS possui um comando que permite «rodear» o problema?

... é possível acrescentar novas vias à PATH corrente sem alterar o AUTOEXEC.BAT e restabelecer o sistema (vulgo *reset*)?

Pois bem, se sabia considere-se devidamente informado. Passe para o artigo seguinte da **Spooler**. Se não sabia, nada de complexos: isto da informática é uma verdadeira «caixa de surpresas». Talvez, por isso, seja tão fascinante.

Nada melhor que «ver com os próprios olhos» para iluminar os espíritos mais cépticos. Façamos uma pequena experiência com o nosso AUTOEXEC.BAT. Comecemos por «pôr a salvo» o ficheiro existente que é suposto estar a funcionar sem problemas, copiando-o para uma disqueta. Seguidamente, voltemos ao C:\ e vamos editá-lo com o EDIT (ou com qualquer outro editor de texto):

```
C:\>EDIT AUTOEXEC.BAT
```

Localize a linha da PATH e conte os caracteres (incluindo os 4 da palavra PATH e o espaço entre a palavra e o primeiro directório). Muito provavelmente a sua via de pesquisa limita-se ao indispensável e o número de caracteres não ultrapassará os 50 ou 70 (ou menos no caso de utilizadores mais «tímidos»...). Mas imagine que, por razões de ordem profissional ou simples capricho, desejava possuir a seguinte via de pesquisa:

```
PATH C:\DOS; C:\STACKER; C:\DOSUTIL; C:\WINDOWS; C:\ALDUS;
C:\PM4; E:\CLIPPER5\BIN; E:\BORLANDC\BIN; E:\DBASE; E:\UTIL\
COMPRIME; E:\UTIL\TDRAW; E:\VIRUS
```

Dirá o leitor: – Mas que disparate! Para quê tanta complicação? E 141 caracteres!?! – Certo. Poderíamos discutir o assunto durante horas. Mas o que está em causa, por agora, é a questão do limite dos 127 caracteres. Portanto, faça o favor de continuar. Imagino que, para além dos directórios constantes da sua PATH, terá outros no seu disco rígido. Provavelmente até terá duas partições. Logo, não lhe será difícil «arranjar» *drives* e directórios para acrescentar à via existente. Vá lá: digite os nomes, tudo certinho e sempre na mesma linha. Já está? Óptimo. Agora conte o total de caracteres. Quantos tem? Só 123?!? Assim não vale!. A experiência só resulta com mais de 127! Não tem mais directórios? Então crie mais um, ou dois, se necessário. Claro que terá de sair do EDIT gravando previamente as alterações já feitas ao AUTOEXEC.BAT, para não ter de repetir tudo do princípio.

Bem, isto foi só um «aparte». Conseguiu 129? 133? Explêndido. Vamos imaginar que obteve os 141 do meu exemplo acima. Agora grave o ficheiro e saia do EDIT. Depois, na linha de comando (vulgo *prompt*) digite:

```
C:\>AUTOEXEC
```

Para esta experiência não se torna necessário restabelecer o computador, pois só se pretende «introduzir» a nova PATH bastando, para isso, executar o AUTOEXEC. Mas, pensará o leitor: – Se era só para introduzir a nova PATH, não poderia tê-lo feito a partir da linha de comando?– De facto, não. É que esta «história» dos 127 caracteres começa, na realidade, pela linha de comando: o tal limite aplica-se a qualquer cadeia de caracteres, quer seja numa linha de ficheiros de comandos, quer introduzida pelo teclado. Tem dúvidas? Então experimente digitar uma letra qualquer «de carreirinha» e vá contando as «tecladas»...E não se alarme com o «grito» de protesto que ouvirá quando tentar introduzir o centésimo vigésimo oitavo carácter...

Mas voltemos à nossa experiência. Executado o AUTOEXEC.BAT, chegou o «momento da verdade»: vamos ver o que resultou daquela cadeia de caracteres que escrevemos, ou seja, «em que estado ficou a nossa PATH». Para isso temos duas alternativas: o comando SET ou o comando PATH. Escolha o último por ser mais directo (e talvez menos conhecido...):

```
C:\>PATH
```

e o resultado:

```
C:\>C:\DOS; C:\STACKER; C:\DOSUTIL; C:\WINDOWS;
C:\ALDUS; C:\PM4; E:\CLIPPER5\BIN; E:\BORLANDC
\BIN; E:\DBASE; E:\UTIL\COMPRIME; E:\UTIL\T
```

Como se vê, o computador «rejeitou» os 13 últimos caracteres da cadeia original (DRAW;E:\VIRUS), pois a «capacidade de digestão» da linha de comando foi ultrapassada. E se chamássemos o TheDraw ou o Scan a partir de C:\ ou de E:\, receberíamos a familiar mensagem *Bad command or file name,* pois os directórios onde aqueles programas estão instalados não ficaram incluídos na via de pesquisa.

– Muito bem –, dirá o leitor, – Continuo a pensar que uma tal PATH é um exagero. Além disso, para quê, se o DOS não a aceita? — Independentemente de não concordar consigo quanto ao «exagero» (um dia destes vamos tomar uma bica e teremos ocasião de discutir o assunto), chegou o momento de lhe revelar a forma de «rodear» o problema.

O DOS possui um comando externo (introduzido a partir da versão 3.1) que, quase sempre, fica no rol dos esquecimentos. Trata-se do SUBST. Este «humilde e ignorado» destina-se a criar *drives* lógicas (não «físicas», repare-se) atribuindo-lhes uma letra. Mas faz mais: também permite **atribuir uma letra a uma *drive* associada a um directório**.

Se o nosso sistema possuir duas *drives*, um disco rígido com duas partições e a segunda partição «aumentada» com o Stacker, teremos ocupadas as letras de **A** a **E.** Logo, estarão livres as letras de **F** até **Z.** É para a estas últimas que iremos «desenterrar» o SUBST, **substituindo um directório completo por uma simples letra do alfabeto.** Vamos experimentar? Mas antes é necessário ultrapassar outra «esquisitice» do DOS: por omissão, a última

letra reconhecida é o **E**. Mais uma vez teremos de recorrer a outro comando raramente utilizado: o LASTDRIVE, específico ao CONFIG.SYS, destinado a informar o sistema da «última letra que iremos utilizar para *drives*». Terá de ir ao CONFIG.SYS e, se pretender utilizar as letras **F** e **G**, acrescentar o comando:

```
LASTDRIVE=G
```

Grave a alteração e saia do EDIT. Posto isto, terá de reinicializar o computador para que a nova instrução tenha efeito. Agora digite na linha de comando:

```
C:\>SUBST F:  E:\UTIL\TDRAW
```

Agora faça:

```
C:\>DIR F:
```

E verá aparecer no ecrã o conteúdo do directório E:\UTIL\TDRAW. Curioso, não é? Não percamos mais tempo e vamos aplicar imediatamente esta «prendazinha» ao nosso AUTOEXEC.BAT. Edite-o novamente e introduza as seguintes alterações:

```
@echo off
C:\DOS\SUBST F: E:\UTIL\TDRAW
C:\DOS\SUBST G: E:\VIRUS
PATH C:\>C:\DOS; C:\STACKER; C:\DOSUTIL;
C:\WINDOWS; C:\ALDUS; C:\PM4; E:\CLIPPER5\BIN;
E:\BORLANDC\BIN; E:\DBASE; E:\UTIL\COMPRIME;
F:; G:
```

Grave o ficheiro, saia do EDIT e execute o AUTOEXEC.BAT. Na linha de comando experimente o DIR F: ou G:. Tudo bem? Agora execute o comando PATH:

```
C:\>C:\DOS; C:\STACKER; C:\DOSUTIL; C:\WINDOWS;
C:\ALDUS; C:\PM4; E:\CLIPPER5\BIN; E:\BORLANDC
\BIN;  E:\DBASE; E:\UTIL\COMPRIME;  F.;G:
```

Como vê, a seguir ao directório E:\UTIL\COMPRIME aparecem as «drives lógicas» **F:** e **G:** que substituem os directórios E:\UTIL\TDRAW e E:\VIRUS, respectivamente. Para pôr tudo a limpo, execute SUBST na linha de comando e obterá:

```
F: => E:\UTIL\TDRAW
G: => E:\VIRUS
```

Já estou a ouvir o comentário do leitor: – Então, se a última letra é o **Z**, porque não indicar logo LASTDRIVE=Z e ficar com o sistema preparado para posteriores necessidades? – Claro que pode fazê-lo. Mas não é aconselhavel pois, a partir da letra **E** (por omissão), cada letra que incluirmos com SUBST «come» cerca de 88 bytes na memória convencional. Façamos um pequeno cálculo:

Número de letras de **F** a **Z** = 21
21 x 88 = 1.848 bytes
1.848 / 1.024 = 1,8K

Só duas letras a mais (**F** e **G**) = 2 x 88 = 176 bytes. Para quê consumir 1,8K quando pode fazer a festa só com 176 bytes? Mais tarde, sempre que o leitor precisar de outros directórios, basta alterar o LASTDRIVE em conformidade.

Para terminarmos este «Sabia que...» dedicado ao PATH, voltemos um pouco atrás e contemos o número de caracteres da nossa via de pesquisa após termos introduzido os novos directórios representados pelas letras **F** e **G**.: a contagem dá-nos 115 caracteres, incluindo a palavra PATH e o espaço. Isto significa que ainda dispomos de 12 caracteres para acrescentar um directório.

Imaginemos que o leitor foi presenteado com um interessante utilitário e decidiu instalá-lo num directório chamado ENSAIOS na *drive* C:. Também pensou que seria conveniente ter este novo directório na sua via de pesquisa, a fim de poder ensaiar o dito utilitário a partir de **qualquer *drive* ou directório**. Estou já a vê-lo a chamar o EDIT para alterar a PATH do seu AUTOEXEC.BAT ou acrescentar uma nova letra por meio do SUBST. Calma! Vou mostrar-lhe uma forma muito mais elegante e cómoda de atingir esse objectivo, evitando «mexer» no AUTOEXEC, no CONFIG e restabelecer o sistema. Está a postos?

O «segredo» consiste na criação de um ficheiro de comandos (vulgo *batch file)* muito simples, mas eficaz. Com o EDIT, ou simplesmente na linha de comando com a instrução COPY CON, crie um ficheiro a que pode dar o nome de VIA.BAT, com as seguintes linhas:

```
@echo off
rem VIA.BAT para acrescentar uma via à PATH
corrente.
PATH=%PATH%;%1
```

Depois de gravar o VIA.BAT em C:\ e regressar à linha de comando, digite:

```
C:\>VIA C:\ENSAIOS
```

Nem precisa de executar o comando PATH para ver o resultado: este aparece automaticamente no ecrã:

```
C:\>C:\DOS; C:\STACKER; C:\DOSUTIL; C:\WINDOWS;
C:\ALDUS; C:\PM4; E:\CLIPPER5\BIN; E:\BORLANDC\
BIN; E:\DBASE; E:\UTIL\COMPRIME; F.;G:;C:\ENSAIOS
```

Isto mostra-nos que o novo directório foi incluído na via de pesquisa e, como vê, nem foi preciso modificar e executar o AUTOEXEC. Pode, desde já, chamar o seu novo utilitário a partir de qualquer *drive* ou directório, pois o sistema «sabe» onde ir procurá-lo.

Mas, como foi isto possível? Não há qualquer mistério: os caracteres **%** (só aplicáveis em ficheiros de comandos) destinam-se a executar uma «substituição». Desta forma, **%PATH%** será substituída pela via corrente, seguindo-se um ponto e vírgula; a variável **%1** será substituída pela palavra (parâmetro) que foi digitada na linha de comando logo a seguir a VIA, isto é, C:\ENSAIOS. Simples, não?

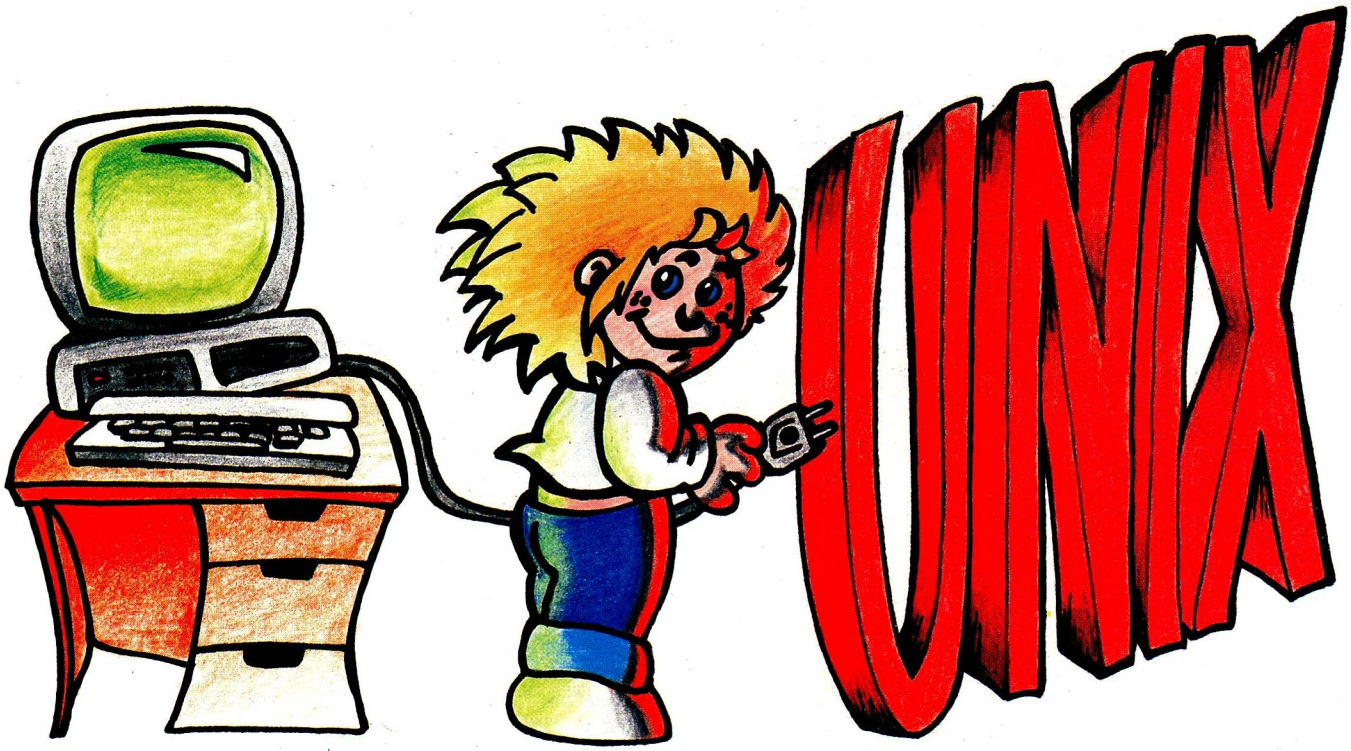
Claro que se tratou de um «expediente de emergência» sem carácter definitivo. Sempre que desligar o computador ou o restabelecer, a via de pesquisa retoma a sua forma original. Se pretender que o directório C:\ENSAIOS passe a figurar permanentemente na PATH corrente, terá de o incluir no AUTOEXEC.BAT. Também pode, como vimos acima, substituí-lo por uma letra «livre» (poderia ser o **H**), não esquecendo de alterar o LASTDRIVE=H no CONFIG.SYS.

E é tudo por hoje. «Sabia que...» voltará com novas e excitantes experiências no próximo número da **Spooler**.



Renato Casquilho

LIGANDO AO UNIX



> “A ideia é acrescentar uma nova capacidade ao sistema de ficheiros do DOS, que é de grande utilidade no Unix...”

O MS-DOS já anda por aí há bastante tempo, e sofreu um bom punhado de revisões, pelo que já aprendeu alguns truques com outros sistemas operativos, desde o Macintosh até ao Unix. Quem conhece este último sabe que deixa o DOS a «milhas» em termos de utilitários, que podemos combinar para conseguir os efeitos mais incríveis. É claro que a oportunidade não foi deixada em claro e, quer a Microsoft, quer outros fabricantes, têm disponibilizado em DOS o equivalente a alguns dos mais usados utilitários do Unix. O que vos proponho desta vez segue esta linha de acção, mas vai um pouco mais longe. A ideia é acrescentar uma nova capacidade ao sistema de ficheiros do DOS, que é de grande utilidade no Unix: **ligações simbólicas**, ou como são vulgarmente conhecidas, *links*.

LIGAÇÕES

Para explicar muito rapidamente o que são as ligações, é necessário lembrar como é guardado um ficheiro em disco. Antes de mais existe num directório uma referência com o nome do ficheiro, um apontador para os dados nele contidos e mais alguma informação adicional. Imaginemos agora que é possível existirem várias entradas de directório para um mesmo bloco de dados, com outros nomes e mesmo em directórios diferentes. A estas entradas adicionais chamamos **ligações**, uma vez que ler ou escrever nestes ficheiros terá exactamente o mesmo efeito que ler ou escrever no ficheiro original, do qual estas ligações são indistinguíveis. No mundo Unix são também conhecidas por *hard links*.

Existe outro tipo de ligações que funciona de um modo ligeiramente diferente, embora o efeito final seja semelhante. Trata-se das ligações **simbólicas**, que mais não são que pequenos ficheiros contendo o nome de um outro. Uma marca na entrada do directório correspondente indica que não devem ser tratados como ficheiros verdadeiros, mas sim como ligações para o ficheiro cujo nome contêm. Estes têm algumas vantagens, tais como poderem apontar para ficheiros não existentes ou em *file systems* diferentes. Um *file system* em Unix é mais ou menos equivalente a uma unidade de disco em DOS. É este tipo de ligações que os utilitários incluídos na disqueta **Spooler** deste mês pretendem duplicar.

NA DISQUETA

Podem encontrar um conjunto de três utilitários que passo a apresentar antes de mais, para poderem desde já

experimentar. O primeiro deles, e o mais importante de todos, é o LINKTSR.COM que, como o nome indica, é um programa residente em memória que se encarrega de alterar algumas funções do DOS (mais sobre este assunto adiante). A única coisa que têm a fazer com ele é carregá-lo em memória uma vez e esquecer que ele existe. Chamo aqui a atenção aos que usam DOS 5.0 numa máquina 386, que o poderão carregar em memória alta através do comando LH.

Depois poderão começar a criar *links* usando o LN. A sintaxe para utilização deste utilitário está descrita no ficheiro LN.HLP, que pode ser lido quer através de um TYPE, quer através da invocação do LN com o parâmetro **/?** ou **/H**. A sintaxe deste comando é semelhante ao seu homónimo existente em Unix. O objectivo é permitir aos utilizadores uma transição fácil entre os dois sistemas operativos.

Resta fazer referência ao LDIR que produz um resultado semelhante ao DIR, mas que tem em conta a existência de ligações, mostrando os ficheiros por elas apontados. Mais uma vez, podem encontrar informação complementar no LN.HLP.

ASPECTOS TÉCNICOS

Embora para alguns leitores bastem os parágrafos anteriores, existem outros, para os quais a **Spooler** é especialmente dedicada, que querem sempre saber o porquê. Ora aí vai.

O primeiro problema que se me deparou na elaboração deste conjunto de programas, foi como havia de distinguir a ligação simbólica de um ficheiro normal. No Unix isto é feito através de um atributo, tal como os que estabelecem as permissões de leitura e escrita. Tentei fazer o mesmo em DOS, pois em cada entrada de directório existem 8 bits reservados para atributos, dos quais só 6 têm um significado atribuído pelo DOS. Porém, a Microsoft, muito sensatamente diga-se, incluiu uma verificação dos parâmetros da função «Muda Atributo», impedindo o utilizador de usar os bits restantes, garantindo assim compatibilidade com futuras versões de DOS que eventualmente os usem. Só alterando directamente os directórios em disco o poderia fazer, o que era bastante arriscado, e muito pouco aconselhável.

Não podia também mudar o significado de nenhum dos atributos permitidos sem tropeçar neste ou naquele problema, o que descobri ao tentar usar o atributo «System» que no DOS actual é completamente inútil, sendo apenas uma das relíquias da compatibilidade CP/M com que o DOS 1.0 vinha carregado. Teria pois que existir outra solução.

Percorrendo a estrutura de uma entrada de directório, e excluindo os vários campos uns atrás dos outros, surgiu-me a ideia de usar o campo reservado ao tamanho do ficheiro. Como? Poderia fazer com que as ligações tivessem um comprimento definido (111 na versão final) o que daria para fazer uma primeira triagem. Porquê 111? Primeiro: comporta o tamanho máximo de uma especificação de ficheiro em DOS, além de algum espaço extra para a assinatura. Segundo: facilmente identificável numa listagem de directório pelo utilizador. Terceiro: e por que não?

Para ter a certeza que o ficheiro de 111 bytes em questão era mesmo uma ligação, bastava incluir uma assinatura no seu conteúdo. Pode aqui parecer que se poderia optar logo à partida pela solução da assinatura, mas tal não seria aconselhável, pois a verificação da assinatura em todos os ficheiros que fossem abertos ou executados poderia degradar o desempenho do sistema

“O primeiro problema que se me deparou na elaboração deste conjunto de programas, foi como havia de distinguir a ligação simbólica de um ficheiro normal”





> As funções baseadas em handles podem considerar-se divididas em dois grupos lógicos: de gestão de ficheiros e de entrada/saída

DOS que o tratariam como um ficheiro normal, e quais teriam que ser modificadas.

As funções que o DOS oferece para tratamento de ficheiros dividem-se em dois grupos: as baseadas em FCB's (*File Control Block* - Bloco de Controlo do Ficheiro) e as baseadas em *handles*, que são simples inteiros, atribuidos pelo sistema a cada um dos ficheiros abertos. As primeiras apareceram com a versão 1.0 do DOS, não conseguem lidar com directórios e são consideradas obsoletas. O seu uso é desaconselhado pela própria Microsoft. Já o segundo grupo que surgiu como DOS 2.0 é bastante mais flexível, ou não fosse ele moldado a partir dos serviços Unix com o mesmo objectivo. À partida decidi não incluir suporte para *links* através do primeiro grupo, por já não ter interesse. Curiosamente, pouco depois descobri que o COMMAND.COM do DOS 5.0 ainda utiliza muitas das funções baseadas em FCB's, ao que me parece por facilitarem o processamento dos meta-caracteres * e ?.

As funções baseadas em *handles* podem considerar-se divididas em dois grupos lógicos: de gestão de ficheiros e de entrada/saída. Entre as primeiras incluem-se as funções «Apagar'Ficheiro», «Mudar Nome», «Mudar Atributos», que devem incidir sobre a ligação em si e não sobre o ficheiro. Seria indesejável, por exemplo, que ao tentar apagar uma ligação se apagasse o ficheiro para o qual ela aponta, passando a ligação a representar um ficheiro inexistente.

Quanto às funções de entrada/saída, estas só são acessiveis através do *handle* obtido na abertura do ficheiro. Todas as operações daí para a frente involverão apenas esse código, e não o nome. Era pois este o sítio ideal para introduzir o programa residente.

Basicamente, o que este faz é interceptar todas as chamadas de abertura a um ficheiro. De seguida verifica se é um ficheiro «autêntico» ou se é uma ligação simbólica. No primeiro caso deixa prosseguir a operação, no segundo substitui o nome da ligação pelo seu conteúdo e prossegue também, retornando na mesma o *handle* esperado, que o programa chamador usa muito normalmente como bem entender.

O programa que tentou abrir o ficheiro não pode ver que algo de anormal se passou, escreve e/ou lê nesse *handle*, actuando assim sobre o ficheiro apontado pela ligação.

Algo de semelhante acontece com a função «Executa Ficheiro» permitindo as ligações para ficheiros executáveis.

Decidi ainda incluir a função «Cria Ficheiro» que é muitas vezes usada para limpar um ficheiro existente e que se não fosse tratada causaria a destruição da ligação.

Decidi ainda suportar ligações aninhadas, ou seja, um *link* pode apontar para outro, que por sua vez pode apontar para um terceiro, e assim sucessivamente. Aqui surge um problema: se uma ligação apontar para si mesma, teríamos um ciclo infinito e consequente bloqueio do sistema cada vez que o tentássemos abrir. A maneira mais simples de evitar que tal aconteça, é estabelecer um contador de iterações com um valor máximo definido como um valor arbitrariamente grande, na versão corrente de 100. Isto proíbe o encadeamento de mais de 100 ligações (alguém estava com essas intenções?) ou por outro lado, como nos interessa, de fazer ligações recursivas. Nestes casos, é devolvido o *handle* para a própria ligação, pelo que é uma das maneiras de aceder ao seu conteúdo com o LINKTSR.COM instalado.

Quanto aos utilitários de apoio, LN e LDIR, pouco há a dizer do modo como funcionam. O único ponto que vale a pena salientar é o modo como o LDIR lê o conteúdo da ligação para o mostrar na listagem do directório. Se for aberto para leitura através das funções normais, obtemos o conteúdo do ficheiro por ele apontado, e não o nome, como seria desejável neste caso. A solução passa pelo emprego das funções baseadas em FCB's, já mencionadas acima, que não sofreram nenhuma alteração. Para mais pormenores não haverá nada melhor do que consultar o código fonte em C destes dois uitlitários, que como é já habitual acompanham os executáveis na disqueta.

EM USO

Além do que é referido no ficheiro LN.HLP, é preciso dizer que, como a única validação feita ao conteúdo da ligação é a busca de meta-caracteres * e ?, que como é lógico não são permitidos, se podem conseguir efeitos estranhos. Em primeiro lugar é permitida a especificação relativa de nomes de ficheiros, pelo que o seu efeito estará dependente do directório corrente quando a ligação for usada, e não do directório a partir do qual foi criada. Embora isto possa vir a ser útil em alguns casos, na maior parte das vezes será incómodo, pelo que aconselho a que se façam as ligações epecificando o percurso completo do ficheiro em questão, incluindo a letra identificadora da unidade de disco. É também possivel criar ligações para ficheiros inexistentes. O que acontece nestes casos é que ao usarmos a ligação teremos respostas do estilo *File not found"*ao tentar aceder à ligação, como se de facto não existisse.

Quanto a questões de compatibilidade com programas existentes não encontrei problemas, tendo inclusive testado várias aplicações sob Windows, incluindo a recente versão 3.1. Apenas um reparo àqueles programas que criam cópias de segurança dos ficheiros que alteram, geralmente colocando-lhes a extensão BAK. Isto é feito de um modo que inevitávelmente interfere com as ligações, pois esses programas mudam o nome do ficheiro original, que é a ligação, e criam um novo com o nome do original. Se o pretendido era a alteração do ficheiro apontado pela ligação, o objectivo não é conseguido, pois o que acontece é que a ligação passa a ser a cópia de segurança sendo criado um ficheiro novo com o conteúdo alterado.

Outro problema que poderia surgir é a tentativa de uso das ligações sem o programa residente em memória. Com ficheiros de dados o problema não é muito grave. O que veremos aparecer se fizermos um TYPE a uma ligação nestas condições é a assinatura, o nome do ficheiro apontado e algum lixo até perfazer os 111 bytes. Se se tratasse de uma ligação para um executável, o problema era mais grave, pois se executado daria origem a um estoiro do sistema. Para evitar isto, os primeiros bytes da assinatura são na verdade código executável que retorna imeditamente ao DOS. Se executarmos assim uma ligação sem termos o LINKTSR.COM em memória o que acontece é a *prompt* a ser devolvida imediatamente sem outras consequências.

SUGESTÕES

Falta apenas mostrar-lhes casos concretos em que podem utilizar ligações simbólicas de modo a melhorar o ambiente DOS.

Em primeiro lugar podem usar ligações para ficheiros executáveis em substituição do comando PATH. Nos PC's com discos lentos, ao executarmos um programa, o sistema por vezes demora uma eternidade enquanto pesquisa a longa lista de directórios especificadas no PATH. A solução é colocar no directório \DOS, onde presumo estar o sistema, ligações para os diversos programas e garantir assim a sua acessibilidade a partir de qualquer sítio sem degradar o desempenho, pois podemos substituir o extenso PATH por um directório apenas.

Podemos também restringir o acesso a um determinado comando apenas a alguns directórios usando ligações, enquanto que com o uso do PATH ficam disponiveis para todo o sistema.

garantir assim a sua acessibilidade a partir de qualquer sítio sem degradar o desempenho, pois podemos substituir o extenso PATH por um directório apenas.

Podemos também restringir o acesso a um determinado comando apenas a alguns directórios usando ligações, enquanto que com o uso do PATH ficam disponiveis para todo o sistema.

Mais importante são as ligações para ficheiros de dados, para os quais o PATH não serve para nada. Temos o APPEND, mas este é um tanto imprevisivel, pelo que eu não gosto muito dele. Podemos também simular efeitos semelhantes aos conseguidos com SUBST, JOIN e ASSIGN, pois as ligaçõpes podem apontar, como já foi dito, para unidades de disco diferentes.

No entanto as ligações simbólicas são para mim úteis sobretudo em duas situações: distribuição dos ficheiros de um projecto por diversos directórios, e suporte ao tão badalado OLE (*Object Linking and Embedding*) do Windows 3.1.

No primeiro caso, servem-me para distribuir os ficheiros de um projecto por diversos directórios, e mesmo por diversas unidades de disco, sem ter problemas com cópias desactualizadas. Por exemplo, num projecto em que estou a trabalhar e que involve varios módulos em linguagem C e *assembly*, que partilham alguns ficheiros, a única opção que tinha era mantê-los todos num mesmo directório de modo a estarem prontamente acessíveis, o que implicava uma confusão nada propícia a bons resultados. Agora posso tê-los em directórios separadas, partilhando apenas o necessário através de ligações simbólicas cuidadosamente escolhidas.

Quanto ao OLE, o melhor é dizer primeiro em que é que este consiste, dado que, para muitos, é ainda uma novidade, deixando desde já claro que nada tem a ver com as lides tauromáquicas! Para aqueles que estão familiarizados com o Windows 3.0, de certeza que já se serviram do *clipboard* para cortar, por exemplo, um desenho do PaintBrush para um documento Write. O OLE leva este conceito mais longe. A ideia base é a actualização automática dos recortes quando o original é alterado.

Voltando ao nosso exemplo, se estivermos a usar o OLE e posteriormente alterarmos o desenho, ele é automaticamente actualizado no ficheiro Write. Isto pode ser feito de duas maneiras: ligando ou «embebendo» o objecto em questão: daí o nome. No primeiro caso, o que acontece é que o ficheiro onde colamos um objecto, apelidado de «cliente», guarda o nome do ficheiro original e da aplicação a que pertence, apelidada de «servidora». Cada vez que usarmos o cliente, o Windows encarrega-se de nos apresentar uma versão actualizada dos objectos por ele contidos. A desvantagem desta abordagem é que se mudarmos de lugar o ficheiro de onde recortamos o objecto, a ligação quebra-se, ficando nós numa situação semelhante àquela em que estavamos com o Windows 3.0 com o seu simples *Cut & Paste*.

Se por outro lado o recorte ficar embebido no ficheiro cliente, a ligação ficará quebrada desde logo. A vantagem em relação a um corte «pré-OLE», se assim lhe podemos chamar, é que teremos a possibilidade de o alterar, uma vez que este é guardado no formato da aplicação que lhe deu origem, e não no formato da aplicação destino, como acontecia com a simples colagem do 3.0.

No System 7.0 do Macintosh, a ultima versão do sistema operativo da Apple, existe algo de semelhante ao OLE, com características próprias e bastante úteis. Neste caso um objecto que vai ser cortado e colado noutro documento é guardado numa «pasta» reservada para o efeito, ao que se chama «Publicar um objecto». (Para aqueles que não conhecem o célebre computador da maçã e toda a gíria a ele associada, uma pasta é o equivalente a um directório.) Outra aplicação pode a partir daí ir buscar o que muito bem entender à referida pasta, o que é



chamado «Subscrever um objecto». A vantagem deste esquema é que as ligações são inquebráveis, pois todas as aplicações têm um ponto de encontro onde podem trocar os objectos, que é a dita pasta.

Pois bem, com OLE e com ligações simbólicas podemos recriar a flexibilidade do sistema da Apple. No meu PC tenho um directório na raiz a que chamei PUBLIC.OLE. Quando quero transferir informação de um ficheiro para outro, movo a origem para o referido directório deixando no seu lugar uma ligação simbólica. No destino faço então todos os recortes que quero a partir do ficheiro que está em \PUBLIC.OLE. Qualquer alteração no ficheiro original é imediatamente reflectida através da ligação no ficheiro publicado, e daí, por intermédio do OLE, para todos os sítios onde foi usado.

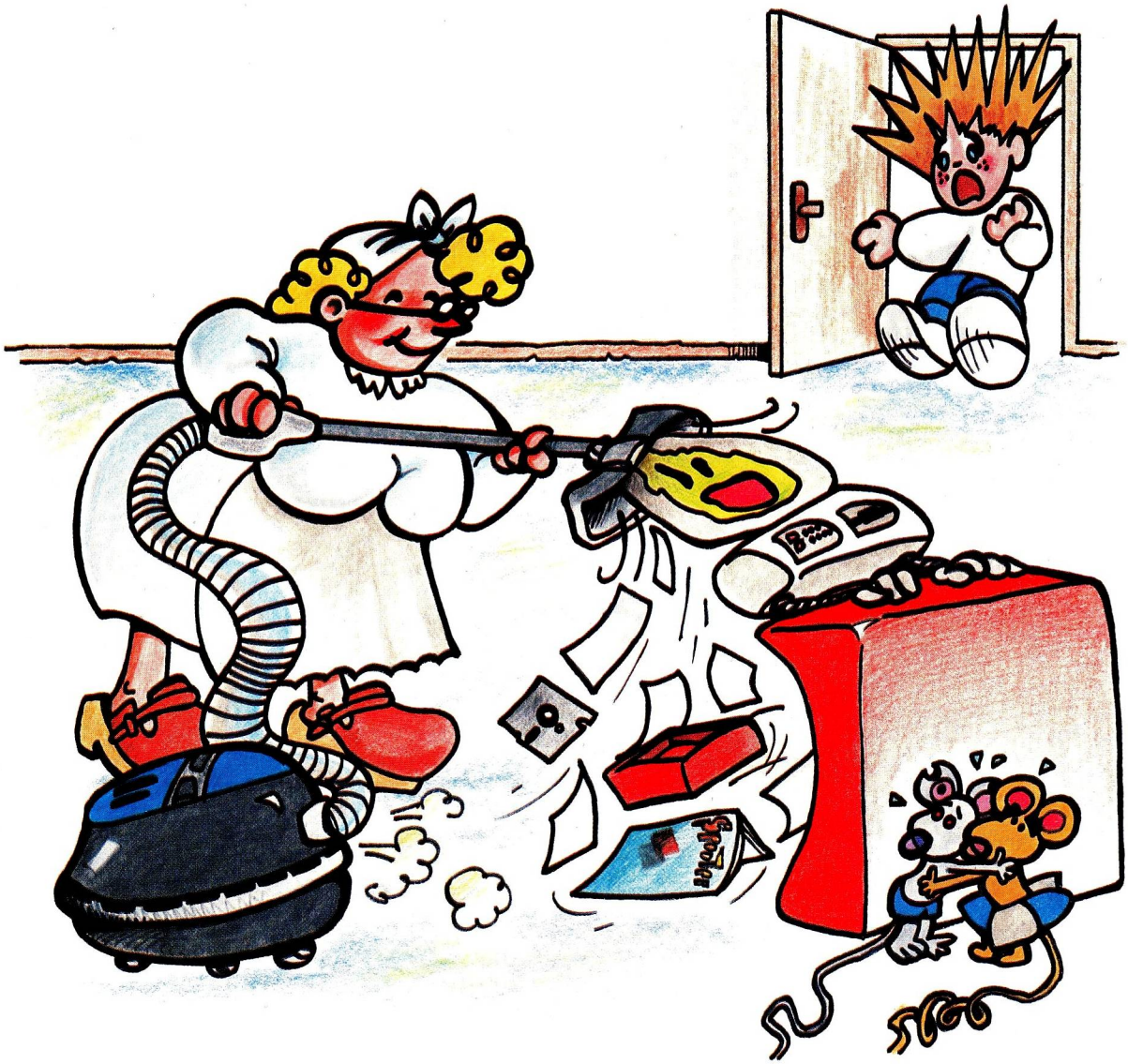
Deixando as janelas e voltando ao C:\>, que é ainda o ambiente de trabalho da maioria de nós, podem criar-se algumas ligações bastante excêntricas, fazendo ligações, não para ficheiros, mas para *device-drivers* tais com PRN, AUX, CON, NUL, etc. Experimentem!

Espero que este conjunto de utilitários lhes seja útil, pelo menos tanto quanto me tem sido a mim, permitindo-lhes dar mais um passo em frente do DOS «simples» que cada vez satisfaz menos utilizadores.

José Orlando Pereira

LIMPA

UM ASPIRADOR PARA O SEU DISCO



Uma das omissões de todas as versões do sistema operativo MS-DOS (incluindo a 5.0), é a capacidade de, com um único comando, remover directórios e/ou subdirectórios, sempre que estes não estejam vazios. Provavelmente, a Microsoft fez esta opção para evitar que utilizadores mais distraídos causassem erros que mais tarde lhes viriam a trazer muitas dores de cabeça. Para além de remover tudo o que existe dentro de um directório (ficheiros e subdirectórios), este programa tem outro modo de funcionamento em que só apaga ficheiros coincidentes com uma máscara especificada, não apagando directórios. Normalmente a «limpeza» é efectuada tanto no directório especificado, como em subsequentes. Desde já saliento que este programa não evita erros do utilizador. No entanto, é sempre pedida confirmação antes de apagar, seja o que for.

A SINTAXE BÁSICA

Ao executarmos o programa, se não for especificado nenhum parâmetro, é-nos apresentado o seguinte ecrã de ajuda:

LIMPA 2.0 por Carlos Ladeira 21/06/1992

Apaga ficheiros e subdirectórios de um local especificado.

LIMPA [-i] [-r] [-l] <pathname>

i	Os ficheiros são apagados de forma irrecuperável
r	Cria um ficheiro com o relatório da operação
l	Restringe a operação ao directório especificado

<pathname> (3 formatos possíveis)

[d:][...\]dir\	Apaga todos os ficheiros e subdirectórios de "dir".
[d:][...\]dir	Igual ao anterior, apagando também "dir".
[d:][...\]nome[.ext]	Apaga só ficheiros coincidentes com nome.ext".

Para o funcionamento do programa é necessário especificar uma **<pathname>** que indica o local onde será efectuada a «limpeza». Para além desta, é também possivel especificar vários comutadores. Segue-se uma descrição detalhada de cada um deles:

O comutador **-i** faz com que todos os ficheiros sejam apagados de forma irrecuperável. Depois de apagado o ficheiro, é totalmente impossível recuperar a informação lá contida. Isto é conseguido escrevendo «lixo» por cima do conteúdo do ficheiro.

O comutador **-r** cria no directório em que está o programa (**LIMPA.EXE**) o ficheiro **LIMPA.REL**. Este conterá o nome de todos os ficheiros e directórios que foram apagados durante a operação. Esta informação poderá ser útil mais tarde para uma eventual recuperação dos ficheiros apagados. No final da operação, é sempre mostrado o local onde o ficheiro de relatório foi criado.

O comutador **-l** força a operação a ser local, isto é, só são apagados ficheiros no directório especificado. Todos estes comutadores são facultativos e podem ser usados conjuntamente.

Falta-nos ainda ver em pormenor o parâmetro **<pathname>**, que pode assumir uma de três formas possiveis:

[d:][...\]dir\
[d:][...\]dir
[d:][...\]nome[.ext]

Os elementos entre parênteses rectos ([...]) podem ser omitidos e, no caso de o serem, serão substituídos pelos valores por omissão correspondentes do sistema operativo. O primeiro caso apaga tudo o que existir (ficheiros e subdirectórios) no directório **[...\]dir** da *drive* **d:**. Quanto ao segundo caso, é idêntico ao anterior, com a diferença que o directório **dir** também é apagado. Por fim, o terceiro caso apaga só os ficheiros que coincidirem com **nome[.ext]** (que pode incluir os metacaracteres: * e **?**) no directório **[...\]** e subsequêntes da *drive* **d:**.

EXEMPLOS

Seguem-se alguns exemplos que espero ajudem na compreensão do funcionamento deste programa:

LIMPA c:\lixo Apaga a directório **\lixo** da drive **c:** e tudo o que lá existir.

LIMPA -i \abc Apaga ficheiros de forma irrecuperável e subdirectórios dentro do directório **\abc** da *drive* corrente.

LIMPA -i -r -l *.bat Os ficheiros (***.bat**) são apagados irrecuperavelmente do directório corrente da *drive* corrente. É criado um ficheiro com o relatório da operação.

Boas e prudentes «limpezas».

Carlos Ladeira



PROGRAMAÇÃO EM C

(PARTE IV)



INTRODUÇÃO

Nesta lição continuaremos a focar aspectos muito importantes da linguagem C. Entre eles, destaca-se o estudo da operação com **ficheiros**. Este assunto é de vital importância na linguagem C. Salienta-se também a descrição do *header* **SETJMP.H** que, por falta de espaço, não foi possível incluir na lição anterior. Vejamos a lista dos assuntos abordados nesta lição:

- Classes de variáveis
- Declaração de tipos pelo programador
- Tipos enumerados
- Conversão de tipos
- Ficheiros
- Programa exemplo (versão 4)

DESCRIÇÃO DO *HEADER* SETJMP.H

As rotinas descritas neste *header* providenciam uma forma simples de retomar de uma função para um local diferentre do local de onde esta foi chamada. Imagine, por exemplo, que num determinado programa existem 10 funções (**f1** a **f10**) e que **f1** chama **f2** que por sua vez chama **f3**, e assim sucessivamente, até que **f9** chama **f10**. Já em **f10**, devido a uma divisão por zero (por exemplo), é necessário um retorno até a função **f2** para resolver esta situação. Normalmente, para efectuarmos esta operação, seria necessário que fosse feito, sucessivamente, de **f9** a **f2**, um ou mais testes a uma variável (que sinalizava a divisão por zero) para o retorno ser efectuado. Ora, facilmente se compreende que uma situação deste tipo só iria complicar o programa. Imagine agora que em vez de gerar toda esta confusão, bastava uma chamada a uma função para retomar de **f9** para **f2**. Basicamente, é isto que se pode fazer com as funções definidas no *header* **setjmp.h**. É até muito simples: basta a chamada a uma função para «marcar» o local de retorno, e uma chamada posterior a outra função para retornar para o local pretendido. A marca do local de retorno é guardada numa variável do tipo **jmp_buf**, que se encontra definida em **setjmp.h** juntamente com mais duas funções, seguidamente descritas:

- int setjmp(jmp_buf var)

Esta função guarda em **var** os dados que posteriormente serão utilizados para retomar para o local de onde a chamada a esta função foi efectuada. Numa chamada posterior a **longjmp**, o retorno será como se fosse o retorno da chamada a **setjmp**, com a diferença no valor de retorno. Numa chamada directa a **setjmp** o valor de retorno é

retorno. Numa chamada directa a **setjmp** o valor de retorno é sempre zero; será um valor diferente de zero, quando for um retorno de **longjmp**. Esta situação é normalmente resolvida num **if**, como por exemplo:

```
if ( setjmp( var) == 0 )  a( );   /* retorno da chamada a
                                     setjmp */
else b( );                        /* retorno da chamada
                                     a longjmp */

– void longjmp( jmp_buf var, int retval )
```

Uma chamada a esta função restaura a informação gravada em **var** na última chamada a **setjmp.** A execução continua como se fosse um retorno de **setjmp** com um valor indicado por **retval** (diferente de zero) na chamada a **longjmp**. Um bom exemplo da utilização destas duas funções encontra-se na calculadora que temos vindo a desenvolver ao longo das lições. A primeira versão que utiliza estas funções é **CALC2.C**.

CLASSES DE VARIÁVEIS

Modificadores de acesso

Existem dois modificadores de acesso em C: **const** e **volatile**. Estes são utilizados para alterar a forma como as variáveis podem ser acedidas ou modificadas. Na declaração de uma variável, se um destes modificadores estiver presente, deverá preceder a definição do tipo. Por exemplo:

```
const float pi = 3.1415;
volatile char flag;

const
```

Esta classe de variáveis tem como característica não deixar que o seu valor seja alterado directamente no programa. As únicas duas formas possiveis de alterar o valor de uma variável da classe **const** são através da inicialização ou através da afectação indirecta (via *pointer)*. Quanto à utilidade destas variáveis, só na criação de variáveis em que, por exemplo, o seu nome representa uma constante numérica (**const float PI = 3.1415;** . Um outro local onde este modificador é utilizado com frequência, é na protecção de parâmetros (do tipo *pointer)* de funções contra a sua alteração. Imagine, por exemplo, que tem uma determinada função que aceita um parâmetro que é um *pointer* para uma *string* (**char** *). O conteúdo desta *string* não pode ser alterada, mas devido a um erro de programação dentro da função, existe uma instrução que altera o conteúdo da *string*. Problemas deste tipo, grande parte das vezes são a causa de erros que demoram eternidades a ser encontrados. Para evitar tudo isto, os parâmetros de uma função podem ser protegidos com o modificador **const**. Por exemplo:

```
void funcao1 ( const char * ptr )
{
  ....
  *ptr = 'C';              /* será sinalizado um erro */
  ....
}

volatile
```

Um pouco diferente do modificador **const,** é o modificador **volatile**. Este último informa o compilador que a variável à qual ele está associado poderá ter o seu valor alterado de formas não muito explícitas. Isto é, a variável pode ser alterada, por exemplo, por uma rotina de *interrupt*.. Para quem não sabe, rotinas de *interrupt* são «funções» que o computador executa, interrompendo momentaneamente os nossos programas para que essas funções executem uma determinada tarefa. Imagine, agora, que uma dessas *interrupts* alterava (propositadamente) a variável **xyz** (no local abaixo indicado) fazendo parte do nosso programa que se encontra em execução:

```
...
xyz = 20;
a = xyz;
/**** aqui ocorre a interrupt ****/
/**** e altera xyz para 50 ****/
b = xyz;
...
```

Se **xyz** não for declarada **volatile**, muito provavelmente **b** ficará com o valor 20 e não 50. Isto porque o compilador, numa tentativa de optimizar o mais possivel o programa, só vai buscar o valor de **xyz** à memória da primeira vez, pois da segunda o valor de **xyz** já se encontra, quase de certeza, num registo do processador. Seria uma perda de tempo aceder novamente à memória. No entanto, se a variável **xyz** fosse declarada **volatile** o compilador não fará este tipo de optimizações com ela, e neste último exemplo **b** ficaria com o valor pretendido, isto é, 50.

Como se viu, este modificador faz com que o compilador não optimize as variáveis. Facilmente se compreende que este modificador só deve utilizar-se quando estritamente necessário, pois pretende-se que os programas sejam o mais optimizados possível.

Modificadores da forma de atribuição

São quatro e utilizam-se para informar o compilador sobre a forma como ele deve atribuir espaço para armazenar o conteúdo das variáveis. São os seguintes: **extern**, **static**, **register** e **auto**.

– extern

Em C existe a possibilidade de dividir o programa fonte em vários blocos (ficheiros), os quais serão compilados separadamente e depois ligados para produzir o programa executável. Imagine, agora, que num dos blocos necessitávamos de utilizar uma variável global que estava definida noutro bloco. Para isto teríamos de recorrer ao modificador **extern** para informar o compilador que a declaração que o segue é referente a uma variável já declarada noutro módulo. Segue-se um exemplo:

```
/** MODULO 1 **/
int a = 20, b = 10;
char c;

void main ( void )
{
  ...
}

void x ( void )
{
  ...
}


/** MODULO 2 **/
extern int a, b;
extern char c;
```

“Em C existe a possibilidade de dividir o programa fonte em vários blocos (ficheiros), os quais serão compilados separadamente...”

```
void y ( int i )
{
  a = b + i;
}

z ( void )
{
  b = a + c;
  return c + b;
}
```

– static

> **Variáveis estáticas (static), são variáveis globais com uma característica muito especial**

Variáveis estáticas (**static**), são variáveis globais com uma característica muito especial. Essa característica que as diferencia das globais, tem a ver com os locais onde elas poderão ser utilizadas. Uma variável estática, embora seja global, só é «conhecida» dentro da função ou ficheiro, se foi declarada, respectivamente, dentro de uma função ou de um ficheiro. No caso da variável ser declarada num ficheiro (fora de qualquer função), ela só é conhecida dentro desse ficheiro. Todos os outros módulos do programa (se existirem) não «saberão» da sua existência. No caso de ser declarada dentro de uma função, a variável estática comporta-se tal e qual como uma variável global, com a diferença que essa variável só poderá ser referenciada dentro da função em que foi declarada. Neste último caso, possuem uma característica muito interessante: preservam o seu valor entre chamadas à função em que foram declaradas. Por exemplo:

```
SOMA ( INT VALOR )
{
int total = 0;          /* Inicializa só na primeira vez */
total += valor;         /* Soma mais esta parcela */
return total;           /* Retorna o total no momento */
```

– register

Tradicionalmente, este modificador só deverá ser utilizado com variáveis do tipo **char** e **int**. No entanto, segundo o **ANSI C**, este modificador poderá ser utilizado com qualquer tipo de dados. As variáveis do tipo **register** deverão ser utilizadas sempre que possivel, pois permitem aumentar a velocidade dos programas. Este aumento de velocidade é conseguido usando registos (variáveis internas do microprocessador). Assim, em vez do valor da variável ficar localizado na memória, ele fica guardado num registo, poupando o tempo que seria necessário para o processador aceder à memória. Este tipo de variáveis só deve ser utilizado em funções (variáveis locais). Uma situação muito comum para este tipo de variáveis é como contador em ciclos **for**. Segue-se um exemplo para demonstrar o uso do modificador **register**:

```
int soma ( int min, int max )
{
  register int i, temp;  /* i e temp são register */

  for ( i = min; i <= max; i++ ) temp += i;
  return temp;
}
```

– auto

Este modificador, tal como o **register**, só deve ser utilizado com variáveis declaradas dentro de funções. As variáveis afectadas por este modificador,são «criadas» no início da execução da função e «destruídas» no final da função. Como, por omissão, todas as variáveis declaradas dentro de uma função são **auto**, este modificador raramente é utilizado.

DECLARAÇÃO DE TIPOS PELO PROGRAMADOR

Em C é possível definir novos nomes de tipos de dados já existentes. Para além dos tipos já definidos, poderão ser definidos outros. Basta recorrer à palavra chave **typedef**. Este processo permite tornar os programas mais portáteis (entre compiladores) e compreensíveis. A sintaxe para esta palavra chave é a seguinte:

```
typedef tipo nome;
```

tipo representa um tipo de dados permitido em C e **nome** é um identificador válido que representará o nome do «novo tipo», o qual não poderá ser um nome já definido. Segue-se um exemplo:

```
typedef char byte;
```

Neste caso foi criado o tipo **byte** que tem as mesmas características do tipo **char**. Depois desta declaração poderemos, por exemplo, escrever a seguinte declaração:

```
byte x;     /* é igual a  char x;  */
```

E agora um exemplo um pouco mais complicado, em que uma estrutura é associada a um «novo tipo»:

```
typedef struct {
  char * nome;
  char * morada;
  char idade;
  } pessoa;

...

/* na linha seguinte só a variável tu é inicializada */
pessoa eu, tu = { "José", "Rua das flores", 30 };
```

TIPOS ENUMERADOS

Uma outra característica importante da linguagem C, é a enumeração. Esta é uma das formas mais utilizadas quando se pretende associar nomes a constantes inteiras. As enumerações são muito utilizadas porque têm uma relação muito próxima com a vida real. Vejamos, por exemplo, uma enumeração dos números 0 a 9:

```
zero, um, dois, três, quatro, cinco, seis, sete, oito, nove
```

A declaração de uma enumeração é semelhante à de uma estrutura, com a palavra-chave **enum** a sinalizar o início da enumeração. Vejamos a sintaxe:

```
enum nome { lista_de_enumeracoes } lista_de_variáveis ;
```

nome, que pode ser omitido, identifica a enumeração, **lista_de_enumeracoes** é uma lista dos elementos enumerados separada por vírgulas. Por último, a **lista_de_variáveis**, que também pode ser omitida, é uma lista de variáveis separada por vírgulas. O próximo exemplo define uma enumeração dos números de 0 a 9:

```
enum numeros_0_9 {zero, um, dois, tres, quatro, cinco,
seis, sete, oito, nove };
enum numeros_0_9 contador1, contador2, total = zero;
```

Como as variáveis enumeradas são tratadas como inteiros, o valor numérico de cada elemento enumerado será um inteiro. A cada elemento da enumeração é dado um valor numérico igual à soma do anterior com 1. No caso do primeiro elemento da enumeração, é-lhe atribuído zero, isto se nenhum valor lhe for atribuído pelo programador. Por exemplo, tendo em atenção a enumeração anterior, a seguinte chamada: **printf("\n %i %i %i", zero, cinco, sete);** mostraria no ecrã: **0 5 7**. Para atribuir um valor a um elemento da enumeração, basta afectá-lo na declaração. Por exemplo:

```
enum { vinte=20, vinte_e_um, vinte_e_dois, trinta=30,
trinta_e_um };
```

Os valores de cada um dos elementos enumerados do exemplo anterior seriam os seguintes:

vinte	20
vinte_e_um	21
vinte_e_dois	22
trinta	30
trinta_e_um	31

CONVERSÃO DE TIPOS (*CASTS*)

Imagine que tem uma variável do tipo inteiro (**int**) e que numa determinada expressão pretende utilizar o seu valor como sendo um número de vírgula flutuante (**float**). Para isto teria de recorrer à conversão de tipos. A forma geral é a seguinte:

```
(novo_tipo) expressao
```

Onde **novo_tipo** deverá de ser um tipo válido do C. Observe o seguinte exemplo:

```
#include <stdio.h>

void main ( void )
{
  int x;

   x=35;
  printf( "\n %d / 2 é %f", x, (float) x / 2 );
}
```

FICHEIROS

O **ANSI C** define um conjunto muito rico de funções para operar com ficheiros, com as quais poderemos escrever ou ler qualquer tipo de dados de ou para um ficheiro. Para operar com um ficheiro é necessário associá-lo a um nome (que será a sua designação). Depois de escolhido o nome, é necessário abrir o ficheiro para podermos ler ou escrever dados. No final, teremos de fechar o ficheiro. Cada ficheiro tem toda a informação que lhe é relacionada numa estrutura do tipo **FILE**. Nesta estrutura são armazenados dados relativos ao ficheiro. Entre estes destaca-se o **indicador da posição corrente no ficheiro**, sendo aí que será lido/escrito o próximo carácter.

A definição desta estrutura, bem como a definição das funções para operar com ficheiros, encontra-se no *header***stdio.h**. Existe ainda uma definição quanto ao tipo de ficheiro: **binário** ou **texto**. Um ficheiro binário é uma sequência de bytes que termina com a marca de fim de ficheiro (EOF); um ficheiro de texto difere porque poderá

“O ANSI C define um conjunto muito rico de funções para operar com ficheiros...”

GRANDE CAMPANHA NATAL

FOTOCOPIADORAS ..... DESDE 142 000$00 (+ IVA)
FAXs ..... DESDE 78 000$00 (+ IVA)
MÁQ. ESCREVER MECÂNICAS ..... DESDE 13 000$00 (+ IVA)
ELECTRÓNICAS ..... DESDE 29 000$00 (+ IVA)
BIPs ..... DESDE 22 000$00 (+ IVA)
TELEFONES S/FIOS ..... DESDE 8 800$00 (+ IVA) ATÉ AOS DE LONGA DISTÂNCIA
WALKIE TALKIES ..... DESDE 13 000$00 (+ IVA) ATÉ AOS DE LONGA DISTÂNCIA
TELEMÓVEIS (GSM ou Analógico) ..... DESDE 135 000$00 (+ IVA)
COMPUTADORES

Consulte os nossos sensacionais preços.



Ninguém vende mais barato

CASA VIOLA

SEDE: Rua da Assunção, 67 – LISBOA – Telef.: 342 46 47 – Fax: 342 67 22
FILIAIS: BRAGA, ESTORIL, VISEU, PORTIMÃO e SETÚBAL

Graficria PUB

ser dividido em linhas, estas separadas pelo caracter '\n'. Segue-se uma tabela com a descrição das funções mais comuns na operação com ficheiros:

NOME	FUNÇÃO
fopen()	Abre um ficheiro
fclose()	Fecha um ficheiro
putc()	Escreve um carácter num ficheiro
getc()	Lê um carácter de um ficheiro
fputs()	Escreve uma string num ficheiro
fgets()	Lê uma string de um ficheiro
fwrite()	Escreve mais que um byte num ficheiro
fread()	Lê mais que um byte de um ficheiro
fseek()	Altera o indicador de posição no ficheiro
fprintf()	É um **printf** com *output* num ficheiro
fscanf()	É um **scanf** com *input* de um ficheiro
feof()	Retoma verdadeiro se for encontrado o fim de ficheiro
remove()	Apaga um ficheiro
fflush()	Transfere todos os dados do buffer do ficheiro para o ficheiro

No entanto, para que todas estas funções funcionem, é necessário que elas «saibam» qual o ficheiro a operar, pois poderemos ter mais que um ficheiro aberto em simultâneo. Para isto, **fopen** (na abertura do ficheiro) retoma um ponteiro do tipo **FILE** para uma estrutura que vai armazenar os dados referentes ao ficheiro aberto. Sempre que pretendermos operar com esse ficheiro, será necessário referenciar o tal ponteiro.

Abrindo um ficheiro

Para abrir um ficheiro utilizamos a função **fopen** que tem o seguinte protótipo:

```
FILE *fopen( const char *nome, const char *modo);
```

nome é um ponteiro para uma *string* com o nome do ficheiro, o qual deverá ser um nome válido para o sistema operativo em uso. Quanto à *string* apontada por **modo**, determina como o ficheiro será aberto, de acordo com a tabela abaixo:

MODO	SIGNIFICADO
r	Só leitura
w	Só escrita
a	Anexa ao fim
b	Modo binário

Os caracteres presentes na tabela anterior, poderão ser combinados dentro da *string* que indica o modo. Se durante a abertura do ficheiro for detectado algum erro, **fopen** retomará **NULL** e o ficheiro não será aberto; caso contrário será retomado um ponteiro a ser utilizado sempre que se pretender utilizar esse ficheiro. Se **fopen** for utilizado para abrir em modo de escrita um ficheiro já existente, este último será destruído e criado um novo. No entanto, se for feita uma anexação, todos os dados enviados para o ficheiro serão anexados no fim do existente. Segue-se um exemplo de um bloco de programa que utiliza **fopen** para abrir o ficheiro "TESTE" no modo leitura/escrita:

```
FILE *ficheiro;
...
ficheiro = fopen( "teste", "rw" );
if (ficheiro == NULL) {
  printf( "\n ERRO na abertura do ficheiro" );
  return -1;
  }
...
```

Fechando um ficheiro

A função **fclose** é utilizada para fechar um ficheiro aberto por **fopen**. Antes de fechar o ficheiro, **fclose** escreve no ficheiro todos os dados que permaneçam no *buffer* do ficheiro. O protótipo de **fclose** é o seguinte:

```
int fclose(FILE *fp);
```

fp é o ponteiro retomado por **fopen** no momento da abertura do ficheiro. Se tudo correr bem ao fechar o ficheiro, o retorno será zero. Um valor diferente de zero indicará uma situação de erro.

Escrevendo um carácter num ficheiro

putc é utilizada para escrever um carácter num ficheiro, tendo o seguinte protótipo:

```
int putc(int ch, FILE *fp);
```

fp é o identificador do ficheiro (retomado por **fopen**); **ch** é o carácter a colocar no ficheiro. Se não ocorrer erro durante esta operação, **putc** retomará **ch**, caso contrário retomará **EOF**.

Lendo um carácter de um ficheiro

Para ler um carácter de um ficheiro utiliza-se a função **getc**, que tem o seguinte protótipo:

```
int getc(FILE *fp);
```

Mais uma vez **fp** identifica o ficheiro. Será retomado **EOF** se for encontrado o fim do ficheiro; caso contrário será retomado o carácter lido.

Lendo/ecrevendo *strings*

Tal como **putc** e **getc** para caracteres, também existem funções para ler e escrever *strings* completas. Para escrever existe **fputs** que tem o seguinte protótipo:

```
int fputs(const char *str, FILE *fp);
```

str é um ponteiro para uma *string* e **fp** identifica o ficheiro onde escrever. Quanto ao valor de retorno, em caso de erro será **EOF**. Para ler uma *string* de um ficheiro será utilizada a função **fgets** que tem o seguinte protótipo:

```
char *fgets(char *str, int tamanho, FILE *fp);
```

fgets lê caracteres do ficheiro indicado por **fp**, enquanto não tiver lido **tamanho** caracteres ou encontrar um '\n'. Os caracteres são colocados no *buffer* **str** que deverá ser suficientemente grande para guardar os caracteres lidos. Retomará um ponteiro para **str**, ou então **NULL** no caso de fim de ficheiro ou erro.

Lendo/escrevendo mais que um byte

Para ler e escrever tipos de dados maiores que um byte, o **ANSI C** providencia duas funções, **fread** e **fwrite**. Os seus protótipos são os seguintes:

```
size_t fread(void *buffer, size_t bytes, size_t num, FILE
*fp);
size_t fwrite(void *buffer, size_t bytes, size_t num, FILE
*fp);
```

buffer será um ponteiro para o local onde estão os bytes a escrever no caso de **fwrite**, ou onde serão colocados os caracteres lidos no caso de **fread**. **bytes** indica o número de bytes a ler/escrever, enquanto que **num** indica o número de vezes que o valor expresso em **bytes** deverá ser lído/escrito. **fp** identifica o ficheiro a ler/escrever. Quanto ao tipo **size_t** é equivalente a **unsigned**. Quanto ao retorno destas duas funções será o valor do parâmetro **num** ou menor se ocorrer erro ou for encontrado o fim de ficheiro na situação de leitura. Por exemplo:

```
while ( ! feof (fp)) *p++= getc( fp );
```

Apagando ficheiros

Quando se opera com ficheiros, é comum ser necessário apagar um outro ficheiro. Para isso, este tem de estar fechado. A função utilizada para apagamento de ficheiros tem o seguinte protótipo:

```
int remove( char *nome );
```

nome é um ponteiro para uma *string* que contém o nome do ficheiro. Se este for apagado, **remove** retomará zero; caso contrário retomará um valor diferente de zero.

Limpando o buffer de um ficheiro

A função **fflush**() tem como objectivo escrever no ficheiro todos os dados que, eventualmente, possam existir no seu *buffer* . O seu protótipo é o seguinte:

```
int fflush( file *fp )
```

Mais uma vez, **fp** identifica o ficheiro; se **fp** for **NULL**, serão «esvaziados» todos os *buffers* para ficheiros abertos para *output*. O valor de retorno será zero se a operação for bem sucedida; caso contrário será **EOF**.

Alteração do indicador de posição no ficheiro

O indicador de posição no ficheiro pode ser facilmente alterado para o local desejado (desde que dentro do ficheiro). Para esta operação utiliza-se a função **fseek**(), com o seguinte protótipo:

```
int fseek( file *fp, long num_bytes, int origem );
```

fp identifica o ficheiro; **num_bytes** é a «distância» a que fica a nova posição de origem; **origem** deverá ser uma das *macros* especificadas na seguinte tabela:

ORIGEM	NOME DA *MACRO*
Princípio do ficheiro	SEEK_SET
Posição corrente	SEEK_CUR
Fim do ficheiro	SEEK_END

Nesta tabela, **SEEK_SET** faz com que o indicador de posição seja movimentado **num_bytes** desde o princípio do ficheiro. **SEEK_CUR** movimenta **num_bytes** a partir da posição corrente. **SEEK_END** movimenta do princípio para o fim do ficheiro. Convém recordar que **num_bytes** é **long**, o que permite o suporte de ficheiros com dimensão superior a 64K.

– FPRINTF() e FSCANF()

Tal como as funções **printf**() **e scanf**() enviam dados para o ecrã, existem duas similares cujo *output* é um ficheiro. Os seus protótipos são os seguintes:

```
int fprintf( file *fp, const char *formato, ...);
int fscanf( file *fp, const char *formato, ...);
```

fp identifica o ficheiro. Os restantes parâmetros já foram vistos aquando do estudo das funções **printf**() e **scanf**().

Determinando o fim do ficheiro

Quando estamos a ler dados de um ficheiro, convém saber se já foi atingido o seu fim. Para o efeito existe a função **feof**(), com o seguinte protótipo:

```
int feof( file *fp );
```

Retomará zero se não for encontrado o fim do ficheiro identificado por **fp**. No caso do retorno ser diferente de zero, significa que já não existem mais dados para ler.

PROGRAMA EXEMPLO (VERSÃO 4)

Nesta nova versão da nossa calculadora foram introduzidas algumas potencialidades. Entre elas destacam-se a inserção de variáveis (A a Z), funções de cálculo matemático e a possibilidade de gravar o conteúdo das variáveis num ficheiro, para que este possa ser utilizado na próxima execução do programa. A fonte da «VERSÃO 4» da calculadora poderá ser encontrada no ficheiro **CALC3.C.**

NA PRÓXIMA LIÇÃO

No próximo artigo (que será a última lição deste curso), para além do estudo das directivas do compilador que não foram integradas nesta lição, faremos uma breve abordagem a um dos compiladores mais divulgados, o TURBO C 2.01. Estudaremos o «Ambiente de Desenvolvimento Integrado» (IDE). Será feita uma breve introdução, seguida do estudo dos projectos. Por último, será feita uma abordagem à «mistura» de rotinas em *Assembly* com linguagem C. E, como não poderia deixar de ser, será discutida a versão definitiva da nossa calculadora.



Carlos Ladeira

CAPTUM



O «package» de programas CAPTUM é um conjunto de ferramentas que lhe permitirão capturar imagens em alta resolução EGA (640x350) ou VGA (640x480) até 16 cores e incorporá-las muito facilmente nos seus programas produzidos em Turbo C ou Turbo Pascal.

Trata-se de um programa residente, pensamos que muito original e também um nadinha sofisticado.

Foi integralmente realizado sem recurso a «Libraries» para além das que acompanham o Turbo C, de modo a ser o mais didático possível.

O programa pode ser melhorado em vários (ou muitos) aspectos e, por esse motivo, o código fonte foi integralmente disponibilizado.

O código fonte deste programa é ele mesmo ainda um bom exemplo de como se podem incorporar as imagens capturadas pelo CAPTUM.EXE num programa em Turbo C.

FICHEIROS INCLUÍDOS

– CAPTUM.EXE: Programa de captura.
– CONVERTE.EXE: Conversor das imagens capturadas para *bit map* adequado às funções *putimage* do Turbo C e Turbo Pascal.
– BINTOC.COM: Converte *bit map* para *character array* em linguagem C.
– BINTOPAS.COM Converte *bit map* para *character array* em linguagem Turbo Pascal.
– MODO.COM Permite efectuar mudança de modo gráfico.
– O código fonte integral do CAPTUM.EXE
– O código fonte integral do CONVERTE.EXE
– O código fonte integral dos BINTOC.COM e BINTOPAS.COM
– Mini demonstração em Turbo Pascal Pronto para compilar.
– Mini demonstração em Turbo C Pronto para compilar.

SÃO AS SEGUINTES AS FERRAMENTAS

CAPTUM.EXE – O programa de captura.

UTILIZAÇÃO

Para instalar em memória este programa digite simplesmente CAPTUM <ENTER>. Para reactivá-lo utilize simultâneamente as teclas SHIFT e 5 do teclado numérico.

Em principio só deverá poder reactivá-lo caso se encontre em modo gráfico no qual o CAPTUM possa cumprir a sua missão, isto é, num dos modos acima indicados e que correspondem aos modos gráficos do BIOS com os números 15, 16, 17 e 18.

Contudo, se o tentar activar em modos texto de 80 colunas, ele oferece-lhe uma oportunidade para ser desinstalado. Nos restantes modos gráficos o CAPTUM.EXE faz «orelhas moucas» e não emerge. Os modos melhores para trabalhar são os modos de 16 cores, isto é, os modos 16 e 18.

Uma vez activado num dos modos gráficos adequados, o CAPTUM apresenta-lhe o seu painel de controlo e um rectângulo que define uma área de captura inicial. A área a capturar pode ser movida, alargada ou encolhida em qualquer direcção. Para tal poderá fazer uso do rato, picando com a seta sobre os botões do painel de controlo, caso disponha de um rato compatível Microsoft.

Se não tiver rato pode utilizar as teclas do cursor que obterá o mesmo resultado. Neste caso deverá utilizar uma das teclas <SHIFT> juntamente com as teclas do cursor para aumentar ou diminuir em qualquer dos sentidos o tamanho do rectângulo de captura.

Quer possua rato quer não, pode utilizar sempre qualquer uma das teclas <CTRL> juntamente com a/ou as outras, ou ainda o botão do rato para acelerar, quer o movimento do rectângulo, quer as suas alterações de tamanho.

O botão direito do rato ou a tecla <ESC> fazem o CAPTUM.EXE desaparecer do ecrã, mas continuar residente em memória. A tecla <END> tenta desinstalá-lo quando ele está activo. E, naturalmente, a tecla <ENTER> ou picar o cursor do rato em cima da «máquina fotográfica», capturam a imagem delimitada pelo rectângulo.

DESEMPENHO

O CAPTUM.EXE deposita as imagens capturadas no directório e *drive* onde foi instalado. As imagens vão sendo numeradas na extensão, começando em .001 e indo até .999. Se houver nesse directório já imagens com o nome CAPTURA e uma extensão numérica, o CAPTUM.EXE não escreve por cima das mesmas, mas vai aumentando a numeração da extensão até encontrar um nome disponível. O programa captura em qualquer parte do ecrã.

Pode reparar que quando o rectângulo se aproximar do painel de controlo, este muda de localização. Se o rectângulo se tornar excessivamente grande, o painel de controlo desaparece e o controlo do rectângulo só pode ser efectuado com o teclado. Contudo, logo que o programa volte a encontrar espaço para o painel, o mesmo volta a aparecer.

Para controlo exacto da área de captura, o CAPTUM.EXE apresenta no painel de controlo um conjunto de mostradores que nos informam a posição dos cantos superior esquerdo e inferior direito do rectângulo.

O CAPTUM.EXE não deverá ser utilizado para captura de imagens de tamanho superior a 64 KB, pois as funções do Turbo C e Turbo Pascal têm esse valor como limite. Se o tamanho do rectângulo se tornar tão grande que se aproxime desse limite, passará a ouvir um sinal sonoro de aviso sempre que tentar aumentar ainda mais esse tamanho.

O CAPTUM.EXE faz uso dos métodos mais seguros existentes para controlo de TSR's e foi amplamente testado, pelo que o seu comportamento deverá ser excelente na generalidade dos casos. Não aconselhamos, todavia, a sua utilização em ambientes multitarefa como janelas DOS em Windows ou Desqview, pois o programa não foi preparado para tal e poderá eventualmente produzir resultados insatisfatórios.

CONVERTE.EXE – As imagens capturadas pelo CAPTUM deverão ser «passadas» seguidamente por este programa.

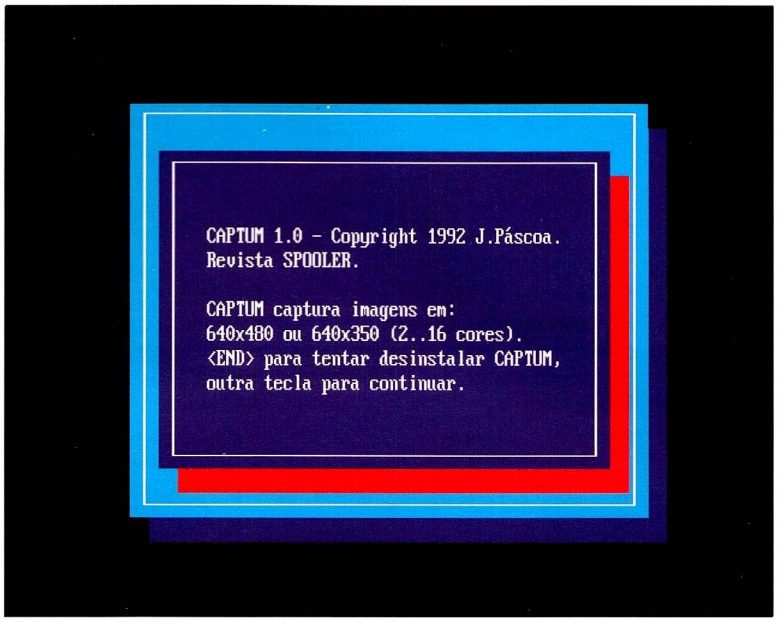
CONVERTE.EXE converte essa imagens em *bit map* idêntico ao utilizado pelas funções **putimage( )** do Turbo C e Turbo Pascal.

As funções deste programa poderiam, naturalmente, ter sido integradas no próprio CAPTUM.EXE, contudo isso ter-nos-ia obrigado a aumentar significativamente o tamanho residente do mesmo ou pelo menos a recorrer a memória expandida, diminuindo a compatibilidade com os computadores que não dispõem da mesma.

Uma segunda alternativa, talvez melhor, teria de passar por uma detalhada análise do algoritmo da função **getimage( )**, mas isso talvez fique para uma próxima revisão do programa.

BINTOC.COM – BINTOPAS.COM:Convertem o *bitmap* obtido pelo CONVERTE.EXE em *arrays* adequados aos Turbos C e Pascal.

Naturalmente que estes dois programas são suficientemente genéricos e podem ser utilizados mesmo para algumas outras finalidades alheias ao *package*.



MODO.COM – Este programa foi trazido do *package* V860 para aqui.

Os utilizadores que queiram fazer um teste rápido ao CAPTUM.EXE podem, após a respectiva instalação, fazer MODO 18 ou MODO 16 seguido de ALT <NUM 5> e aí está o CAPTUM a emergir pronto para trabalhar.

Para praticar é boa ideia começar por tentar capturar algumas partes de imagens .GIF ou .PCX. Pode também fazer os seus desenhos com um programa de desenho como o PaintBrush (versão DOS) ou o DrawPerfect e capturá-los com o CAPTUM.

Pode ainda capturar imagens de programas feitos por outros, mas não seria má ideia perguntar primeiro aos autores se eles se importam. De um modo ou outro verá que é muito fácil produzir com o auxílio do CAPTUM programas de belo efeito e aspecto profissional.

Se a imagem capturada apresentar uma paleta redefinida, pode ter de levar isso em conta no programa que fizer para «projectar»essa imagem.

Também não seria má ideia dispor de um editor das imagens de modo a poder alterar ou retocar as imagens capturadas. Por limitações de espaço, não incluímos neste *package* esse editor, mas enviaremos à cobrança a quem nos solicitar para a **Spooler** o CAPTEDIT.EXE com o respectivo código fonte.

Como o tratamento de imagens gráficas é altamente aliciante, decidimos apresentar, num dos próximos números, um novo conjunto de instrumentos. Continue, portanto, em sintonia connosco.

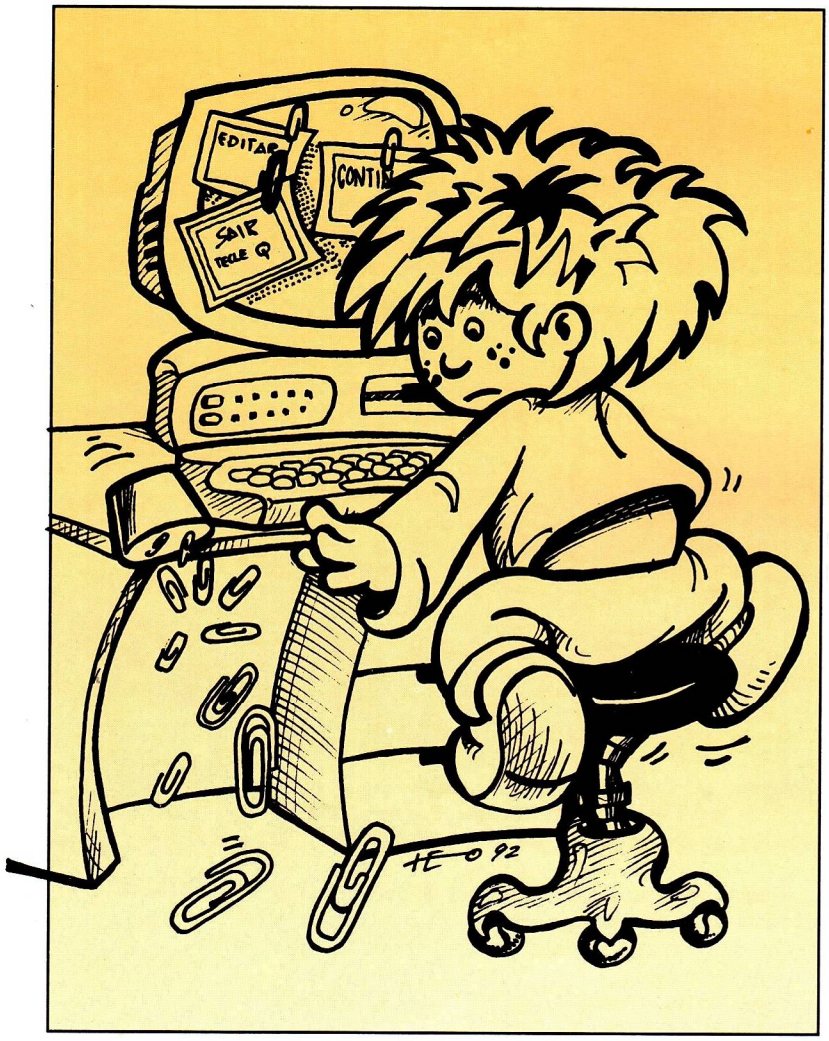
NOTA: Os *arrays* obtidos a partir das imagens capturadas podem também ser utilizados com a função **_putimage( )** do Microsoft C, embora a compatibildade com a sua quase-homónima do Turbo não seja total (pelo menos ao nível da paleta).

José Páscoa

> “O CAPTUM.EXE não deverá ser utilizado para captura de imagens de tamanho superior a 64 KB, pois as funções do Turbo C e Turbo Pascal têm esse valor como limite”

PROGRAMAR EM CLIPPER

(PARTE IV)



O Clipper Summer87 é uma linguagem poderosa que consegue responder a variadas situações. O seu único limite é a imaginação do programador. Com a possibilidade de ligação a funções escritas em *Assembly* e C, o Clipper consegue controlar todo o sistema informático, de uma forma simples, rápida e segura.

Mas, para que tal aconteça, o programador deve ter muita vontade e dedicação e não se limitar a executar os comandos e funções como eles vêm no manual. É necessário **ler, pensar e tentar**. Depois de ver como os comandos e as funções funcionam, o programador fica habilitado a criar o seu próprio estilo: os tipos de ecrãs, menus e o modo de funcionamento de todo o programa.

O leitor, como potencial programador, deverá ser autocritico, ou seja, deverá experimentar o seu próprio programa e certificar-se que a forma como construiu os ecrãs, os menus e toda a aplicação em geral, é funcional e de simples utilização. Não se esqueça: o seu programa vai ser utilizado por vários operadores e cada um terá uma opinião própria sobre o seu trabalho. Ao escrever um programa tente ir ao encontro das necessidades que esses operadores terão ao longo de casa sessão.

Por exemplo: as operações mais utilizadas (inserção e alteração de dados), coloque-as de modo a que possam ser seleccionadas fácilmente; crie ecrãs atractivos e de fácil compreensão (não encha o ecrã com informação desnecessária); quando efectuar alguma função mais demorada, nunca deixe o ecrã vazio sem qualquer mensagem do que o programa está a fazer, pois pode acontecer que o utilizador se canse de ficar à espera e desligue o computador, pensando que este encravou. Uma simples mensagem, como por exemplo: "Organizando a Base de Dados - Aguarde um momento", pode evitar que esta situação aconteça.

Vamos então lançar mãos à obra e entrar no Clipper de cabeça!

COMANDOS DO CLIPPER
Tratamento do ecrã

A utilização conjunta destes comandos (e funções), permite mostrar a informação de um modo atractivo e organizado. A criação de janelas facilita a organização dos dados, dividindo-os segundo as suas características. Com estes comandos é possível criar, apagar e mover janelas dentro do ecrã.

Criar Janelas (@... BOX)

```
@ <lin1>,<col1>,<lin2>,<col2> BOX <limitadores>
```

Este comando desenha uma janela, tendo como coordenadas a linha e coluna do topo (<lin1> e <col1>) e a linha e coluna da base (<lin2> e <col2>). Os <limitadores> são nove caracteres que têm como missão desenhar os contornos da janela. Estes caracteres devem estar contidos entre aspas ("").

Exemplo:

Para desenhar uma janela com linhas duplas, ocupando todo o ecrã e enchendo-a com a letra **S**, o comando seria o seguinte:

```
limitador=chr(201)+chr(205)+chr(187)+chr(186)+ ;
   chr(188)+chr(255)+chr(200)+chr(186)+"S"
@ 00,00,24,79 BOX limitador
```

Apagar Janelas (@...CLEAR)

```
@ <lin1>,<col1> CLEAR TO <lin2>,<col2>
```

Para apagar uma janela préviamente desenhada, utiliza-se este comando. As janela é apagada desde <lin1>,<col1> até <lin2>,<col2>. Caso estes dois últimos valores não sejam especificados, por omissão tomam respectivamente os valores 24 e 79 (a base do ecrã).

Exemplo:

```
@ 10,10 CLEAR TO 12,70
@ 12,70 CLEAR TO
```

Cores das Janelas (SET COLOR TO)

As cores suportadas pelo Clipper são oito:

COR	LETRA
preto	N
azul	B
verde	G
prússia	BG
vermelho	R
magenta	RB
castanho	GR
branco	W

Utilizando o sinal (+), as cores ficam brilhantes:

COR	LETRA
cinzento	+N
azul brilhante	+B
verde brilhante	+G
prússia brilhante	+BG
vermelho brilhante	+R
magenta brilhante	+RB
amarelo	+GR
branco brilhante	+W

O caracter (*) torna as cores intermitentes.

Exemplo:

```
SET COLOR TO +w/r,b/w,gr,,w/r
SET COLOR TO *+gr/b
```

Mover Janelas (SaveScreen e RestScreen)

```
var=SaveScreen(<lin1>,<col1>,<lin2>,<col2>)
RestScreen(<lin1>,<col1>,<lin2>,<col2>,var)
```

As coordenadas indicam qual a área a mover. Utilizando a variável **var** para gravar a imagem, a janela pode ser movida, bastando para isso definir novas coordenadas. Caso se mantenham as mesmas, a imagem é redesenhada no mesmo local.

Exemplo:

```
@ 07,10,14,50 BOX "+-+|+-+|S"          && desenhar a janela
ant_area=SaveScreen(07,10,14,50)       && copiar a janela
@ 07,10,14,50 BOX "#########"          && desenhar nova janela
@ 20,00 say "Prima uma tecla para mover a janela..."
inkey(0)
nova_area=SaveScreen(07,10,14,50)      && copiar nova janela
RestScreen(07,10,14,50,ant_area)       && repor a antiga janela
RestScreen(00,00,07,40,nova_area)      && mover a nova janela
```

SISTEMA DE MENUS

```
@ <lin>,<col> PROMPT <opção> MESSAGE
<mensagem>
```

Este comando define um menu de barras, tendo como coordenadas <lin> e <col>. O nome da opção é indicado com o item <opção> e <mensagem> é uma descrição explicatória. Para posicionar esta descrição numa determinada linha, utiliza-se o comando

SET MESSAGE TO <linha>.

O número da opção seleccionada é guardado numa variável definida através do comando **MENU TO <variável>**.

O seguinte exemplo ilustra a utilização conjunta destes três comandos, na construção e activação de um menu de barras.

```
SET MESSAGE TO 1                          && mensagens aparecem na linha 1
SET WRAP ON                               && permite o movimento contínuo
                                             das opções
SET COLOR TO +w/n,+w/b                    && cores para as opções e barra
                                             de selecção
@ 00,00 PROMPT " Inserir "  MESSAGE  " Acrescentar um registo "
@ 00,09 PROMPT " Alterar "  MESSAGE  " Alterar o registo"
@ 00,18 PROMPT " Apagar "   MESSAGE  " Apagar o registo"
@ 00,26 PROMPT " SAIR "     MESSAGE  " Sair do Programa"
MENU TO opcao                             && activação dos PROMPT's
@ 10,10 say "Seleccionou a opção "+str(opcao,1)
```

PROGRAMA DE EXEMPLO (AGENDA2.PRG)

Nos artigos anteriores, foi elaborado um programa em dBASE que pretendia servir de exemplo aos comandos estudados. Para melhor entender a relação entre o dBASE e o Clipper, vamos reescrever o programa, mas utilizando as potencialidades que este compilador nos oferece. O ficheiro CLIP4.LZH que pode encontrar na disqueta, dá origem ao seguinte ficheiro, depois de descompactado:

AGENDA2.PRG - listagem do programa agenda (versão para Clipper)

O programa AGENDA2 é a passagem fiel do programa escrito em dBASE para Clipper. O objectivo é mostrar ao leitor as diferenças entre as duas linguagens.

> **“O programa AGENDA2 é a passagem fiel do programa escrito em dBASE para Clipper.”**

Por motivos de espaço na disqueta, não é possivel enviar o programa executável, pelo que é necessário proceder à respectiva compilação e ligação. Para isso, execute os seguintes comandos na linha de comando do DOS:

```
C:\>CLIPPER agenda2
C:\>TLINK agenda2, agenda2, mapa,  c:\clipper\
clipper  c:\clipper\extend
C:\>CLIPPER1
```

NOTA: é necessário que os ficheiros CLIPPER.EXE e TLINK .COM sejam acedidos através da *path* do DOS e, caso não tenha as bibliotecas na directoria C:\CLIPPER, indique a correta.

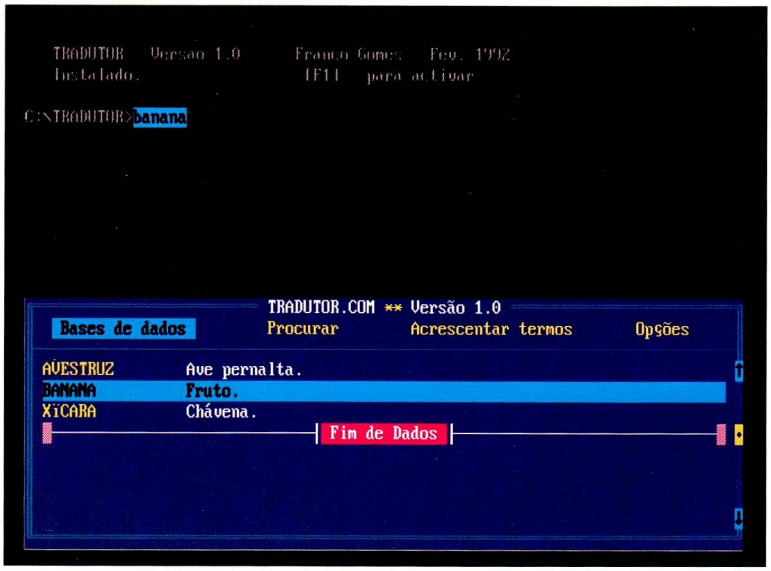
NO PRÓXIMO NÚMERO...

- Comandos de repetição (FOR...NEXT; DO WHILE);
- Comandos de Data.

Victor Manuel Monteiro

TRADUTOR

(PARTE II)



No número anterior falei das rotinas que permitem ao TRADUTOR ficar residente em memória. Agora é a vez de comentar o programa propriamente dito.

O objectivo deste utilitário é o de manusear ficheiros de dados (caracteres ASCII) com uma estrutura própria: uma palavra «termo» com 15 caracteres e um «significado» que pode conter até 60 caracteres.

É o «termo» – que o programa converte sempre para maiúsculas não acentuadas – que sofrerá todas as operações de pesquisa e de ordenação. O «significado» que «mora» logo a seguir ao respectivo «termo» não é dado nem achado para o caso.

Não vou explicar como se utiliza o programa pois, passados dois meses, o leitor já deve estar perfeitamente familiarizado com todos os comandos e opções disponíveis além de que o próprio programa contém todas as informações necessárias.

Decerto já está a usar os ficheiros de dados que colocámos à sua disposição na «Montra Spooler/Shareware(*)» e/ou terá começado o seu próprio dicionário, ou qualquer outro tipo de dados, pois o tradutor pode ser usado para outro tipo de informações, tais como comandos de programação e respectivas descrições, etc., etc.

A parte que se segue poderá, talvez, interessar quem não esteja familiarizado com a linguagem *assembly*, pois as técnicas usadas têm aplicação em qualquer outra linguagem. Não vou sequer «dissecar» qualquer das muitas rotinas. Os confrades «assembliófilos» encontram nos ficheiros-fonte comentários em profusão.

ACTIVAÇÃO

Quando se activa o TRADUTOR (já residente em memória), premindo a tecla de arranque (HotKey), são motivadas as seguintes acções:

- Copia-se toda a imagem presente no ecrã para um outro espaço do *buffer* da imagem visando a sua posterior reconstituição;
- Guardam-se as coordenadas referentes à posição do cursor;
- Esconde-se o cursor numa posição não visível (na linha 25, por exemplo), uma vez que é impossível desligá-lo;
- Isola-se, destaca-se, converte-se para maiúsculas se necessário e guarda-se a palavra que esteja perto do cursor;
- Verifica-se em que metade do ecrã se deve imprimir o menú. Este vai ser impresso de modo a permitir visionar a palavra que está em destaque;
- Procura-se e regista-se o nome do próprio programa (alguns utilizadores rebaptizam os programas). Obtém-se também a descrição drive\directório(s) onde se encontra o TRADUTOR;
- Procura-se no directório obtido na acção anterior ficheiros com a extensão DIC. São listados e contados todos os ficheiros encontrados. A contagem é zero se não existirem ficheiros *.DIC.
- É impresso no ecrã o menú do TRADUTOR.
- Abre-se um ficheiro *.DIC e respectivo *.IDX. Se existirem diversos ficheiros de dados é aberto aquele que foi utilizado pela última vez, ou o primeiro da lista se se tratar da primeira activação do tradutor. Se não existir qualquer ficheiro a acção suspende-se neste ponto e aguardam-se instruções do teclado.
- Carrega-se o ficheiro *.IDX em memória e começa-se a procurar uma palavra que se «pareça» com aquela que foi «pescada» perto do cursor. Explicarei o método de pesquisa mais adiante. Esta pesquisa não se efectuará se o material «pescado» no *prompt* não for considerado útil.

(*) O ficheiro SIGLAS.DIC da disqueta «shareware» ref.ª SH122, continha um pequeno bug. Dois termos continham letras minúsculas o que prejudicava a pesquisa e ordenação. Para corrigir esse inconveniente basta passar para maiúsculas os referidos caracteres. Para proceder a esta correcção poderá editar o ficheiro no Debug do DOS, PcTools ou, simplesmente, apagar os referidos termos dentro do TRADUTOR e tornar a escreve-los. Isto é válido para quem adquiriu a referida disqueta em Outubro, depois dessa data a disqueta já se encontra devidamente corrigida.

- Se a palavra existir no dicionário activo, esta é apresentada em modo destacado e a acção passa a depender das instruções provenientes do teclado; se não existir o termo, é apresentada em destacado a ocorrência mais próxima e pergunta-se ao utilizador se pretende acrescentar o novo termo aos dados.

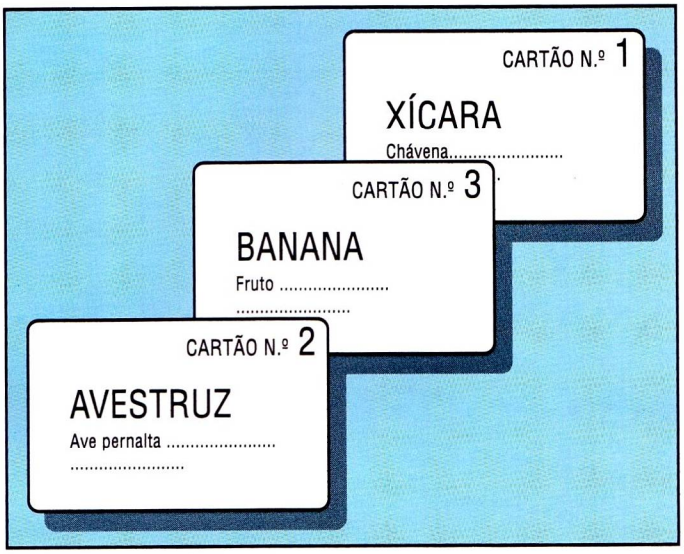


Figura 1

PESQUISA

Para cada ficheiro *.DIC existe um ficheiro homónimo com a extensão IDX. Se abrirem um dos ficheiros DIC, por exemplo no Debug do DOS, verificarão que este é constituido por caracteres ASCII sem quaisquer caracteres de controlo.

Poderão ainda observar que os «termos» não se encontram ordenados alfabeticamente e que aos «termos» se seguem os respectivos «significados» e a estes outros «termos» sem nenhuma separação entre eles.

Era perfeitamente possível «lidar» apenas com este ficheiro de dados mantendo-o sempre ordenado.

Cada vez que um novo termo fosse acrescentado, este seria intercalado na sua posição correcta. Mas, como o Tradutor se propunha (e propõe) tratar com ficheiros de tamanho considerável, praticamente ilimitados, as operações de pesquisa e ordenamento (colocar por ordem alfabética) tornavam-se tanto mais lentas quanto maior o comprimento do ficheiro.

Depois de muitas experimentações optei por utilizar um segundo ficheiro (o tal de IDX).

Para melhor compreensão imagine que o incubiam de organizar um dicionário, utilizando pequenos cartões numerados.

Você começaria, por exemplo, a escrever «Xícara» e respectivo significado no cartão n.º 1. Depois escreveria no cartão n.º 2 o termo «Avestruz», no n.º 3 «Banana», etc.

Para ordenar alfabeticamente os cartões bastaria trocar-lhes as posições. Cada novo cartão seria intercalado na sua posição correcta. Lógico (figura 1).

Este procedimento seria adequado ao manuseamento do ficheiro DIC. Fácil, compreensível, lógico mas... lento!!!

Imagine agora que em vez de escrever os seus «termos» nos tais cartões os escrevia num pequeno bloco de apontamentos com as páginas numeradas. Um termo e respectivo significado em cada página. Sem hipóteses de ordenação (não vale arrancar as folhas). Para encontrar rapidamente um determinado termo você teria que fazer um ÍNDICE – usando os tais pequenos cartões. Estes poderiam conter as primeiras letras do «termo» e o número de página do bloco de apontamentos onde poderia encontrar todos os pormenores (ver figura 2).

Pois é, estou de acordo: não lembra a ninguém tamanha canseira!





Figura 2

Curiosamente foi este último procedimento que o TRADUTOR elegeu. O tal bloco de apontamentos é o ficheiro DIC; os cartões são o ficheiro IDX!

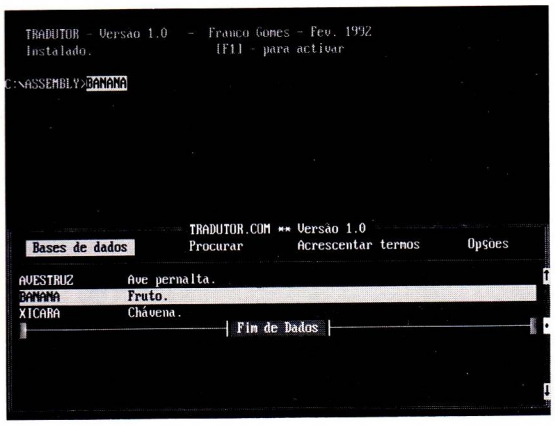
Cada «cartão» de IDX tem cinco bytes com informações acerca de DIC:

41	4B	00	00	00	42	96	00	00	00	58	00	00	00	00
A					B					X				

Figura 3

O exemplo da figura 3 reflecte o conteúdo do ficheiro IDX (em numeração hexadecimal) que seria criado se lançassemos os nossos termos (xícara, avestruz e banana) no TRADUTOR, criando, para o efeito, um novo dicionário. O 1.º byte contém o 1.º carácter do «termo» (no exemplo da fig. 3 o valor hexadecimal 41 é o correspondente ao carácter «A» maiúsculo); os quatro bytes seguintes a posição dentro do ficheiro DIC (endereço) referente a esse «termo». O primeiro grupo de 5 bytes de IDX corresponde à ocorrência menor em DIC e o último grupo à maior ocorrência, ocorrência esta que poderá até situar-se no início dos dados em DIC. Confuso?!

Cada vez que se acrescentar um novo termo a DIC este



é escrito, confortavelmente, no fim dos dados, ficando a ordenação a cargo de IDX. Este procedimento é também adequado quando se apaga um termo. De facto não é necessário retirá-lo de DIC mas, somente, de IDX (neste caso e, para evitar fichas inúteis em DIC, está disponível uma opção para reestruturação total de DIC e IDX que deve utilizar quando proceder a apagamentos massivos).

Quando se solicita ao TRADUTOR uma pesquisa (e para tal basta escrever o termo a pesquisar no *prompt* ou no processador de texto e accionar o *hotkey*), este começa por carregar em memória – num determinado espaço reservado para o efeito – um bloco de dados lidos do princípio do ficheiro IDX. O 1.º carácter da nossa palavra é comparado com o carácter contido em cada campo dos dados de IDX. Se o nosso 1.º carácter é superior ao de IDX, é carregado em memória outro bloco de dados de IDX, até ser encontrada uma igualdade ou, o nosso 1.º carácter se manifestar menor que qualquer carácter em IDX.

Se foi encontrada uma igualdade, recolhe-se o valor dos quatro bytes seguintes do campo de IDX. Este valor não é mais que o endereço «file pointer» do ficheiro DIC que contém o termo completo e respectivo significado.

Lê-se para memória não só essa «ficha» como várias outras, imediatamente anteriores e posteriores àquela. Comparam-se então todos os caracteres da nossa palavra com os do termo recolhido de DIC. Se a ocorrência for encontrada esta aparacerá evidenciada no ecrã, acompanhada de ocorrências anteriores e posteriores. Se não foi encontrada, o utilizador é avisado do facto, e será evidenciada, no ecrã, a ocorrência existente mais próxima.

Como poderá verificar todo este procedimento exaustivo e minucioso é feito num lapso de tempo pelo TRADUTOR. Isto deve-se às capacidades da linguagem *Assembly* e também à técnica utilizada.

No TRADUTOR poderão coexistir até 8 dicionários. Em boa verdade o TRADUTOR poderia conter um número qualquer de dicionários. Foi uma limitação ditada pelo facto de a janela «Bases de Dados» do menú, quando em meio formato, só comportar a descrição de oito ficheiros. Isto poderia, evidentemente, ser resolvido com uma rotina que fizesse *scroll* aos itens da janela, mas já não havia «pachorra» para mais. Prometo, numa futura versão deste utilitário, introduzir esse e outros melhoramentos. Um deles será possibilitar ao TRADUTOR aceder a uma estrutura de dados em cadeia, ou seja, para além do «termo» e significado, ser possível abrir mais «páginas» de dados.

Até lá, despeço-me com amizade, desejando a todos os leitores Boas Festas e um Próspero Ano Novo.

Franco Gomes

PROGRAMAR EM ASSEMBLY (PARTE IV)



Durante os três primeiros artigos sobre linguagem *Assembly* foi nossa principal preocupação tentar «preparar o terreno» para que o leitor menos conhecedor dos «meandros» internos do PC, poder vir a ter sucesso no estudo desta importante linguagem de programação.

Mas agora chegou (finalmente) a altura de nos debruçarmos sobre os aspectos da programação propriamente dita.

O leitor ainda deverá estar recordado que prometemos no artigo anterior que iríamos desenvolver o tema dos *interrupts* através de um pequeno programa que deles fizesse uso. Não se trata de nada de excepcional, pois efectivamente mais difícil seria fazer um programa em *Assembly* que não utilizasse os ditos *interrupts,* já que estes permitem, entre outras coisas, aceder a um vasto «manancial» de rotinas existentes quer no DOS, quer no BIOS.

Os *interrupts* mais utilizados para essa finalidade são os *software interrupts* (referidos no artigo anterior) e as respectivas rotinas estão amplamente documentadas em muitos livros sobre a programação em *Assembly* e também em outras linguagens. De entre todos esses *interrupts*, o mais pródigo em rotinas úteis é o **INT 21h** e recebe o prestigioso nome de *DOS Functions* (ou aqui para nós, Funções DOS).

Seleccionamos uma das Funções DOS carregando o registo **AH** com o valor correspondente a essa Função e, como é frequentemente necessário passar também parâmetros à função (aliás o mesmo aconteceria nas funções da matemática), os respectivos valores são carregados nos registos adequados (e de acordo com o previsto pela sintaxe dessa função), depois disso tudo feito emite-se um **INT 21h** e aguarda-se (uns microsegundos) pela resposta.

Duas Funções DOS foram escolhidas para fazer parte do nosso exemplo, a função **2Ch** (que foi baptizada com o nome de GET TIME) e a função **09h** (com o nome de DISPLAY STRING).

Um outro *interrupt* muito útil, o **INT 10h**, também consta do nosso pequeno programa. Este *interrupt* é designado por *ROM BIOS Video Services* ou simplesmente Video_IO (sendo IO abreviatura de *Input Output).*

No nosso programa utilizamos os serviços **02h** (*Set Cursor Position*) e **03h** *(Read Cursor Position).*

E como não há dois sem três, incluímos ainda um outro *interrupt,*: o **INT 16h**, o qual recebe o nome de *ROM BIOS Keyboard Services* ou apenas Keyboard_IO.

O serviço **01h** do **INT 16h** designa-se por *Report Whether Character Ready* (isto é, propõe-se informar se está algum carácter no *buffer* do teclado à espera de ser lido).

O nosso exemplo ilustra bem a sintaxe das funções que se baseiam nos *interrupts* que acabamos de apresentar, e consideramos ser desnecessário detalhá-las mais, não só por limitações de espaço, e porque é muito fácil compreender o que fazem, mas muito particularmente porque existem literalmente milhares de outras rotinas diferentes que podem ser chamadas através de *interrupts*.

O mais completo «tratado» sobre quais são as rotinas acedíveis por *interrupts*, o que fazem e como obter os seus serviços é um ficheiro electrónico de nome INTERRUPT.LST de um senhor chamado Ralph Brown e cuja última revisão que temos conhecimento ocupa nada menos que 1,7 MB em disco após a descompressão (podendo por conseguinte ultrapassar as 700 páginas se for impresso ...).

É distribuído em regime *shareware* pelas BBS, e recomendamos que o tente obter «a todo o custo» (dica: experimente ver se faz parte da «Montra Spooler»...).

> “... De entre todos esses interrupts, o mais pródigo em rotinas úteis é o INT 21h e recebe o prestigioso nome de DOS Functions...”

Vamos agora tratar de dois temas muito importantes para se poder compreender o funcionamento do programa que aqui reproduzimos. O primeiro é construir um «esqueleto» (este não irá ter ossos) e o segundo é, muito naturalmente, explicar ao leitor o que fazem mnemónicas *Assembly* do tipo das que estão presentes no programa.

Vamos começar pelo esqueleto.

O ESQUELETO

> "O programa que apresentamos tem uma estrutura .COM; isto equivale a dizer que, no momento do carregamento em memória, todos os segmentos coincidem..."

O programa que apresentamos tem uma estrutura .COM; isto equivale a dizer que, no momento do carregamento em memória, todos os segmentos coincidem, isto é,o segmento de código **CS** (*code segment*) tem o mesmo valor do segmento de dados **DS** (*data segment*), o mesmo valor do segmento extra **ES** (*extra segment*) e ainda o mesmo valor do segmento de pilha **SS** (*stack segmet*).

Por este motivo os programas de estrutura .COM estão limitados (e condenados também) a não poderem ultrapassar os 64 KB de tamanho.

Outra importante característica dos programas de estrutura .COM é que são sempre carregados para execução no endereço com deslocamento (ou *offset*) **0100h** (isto é, 256 em decimal).

Para a construção do nosso programa o leitor deverá estar munido de um dos conjuntos de ferramentas que indicámos em tempos, isto é:

```
O MASM, o LINK e o EXE2BIN
ou
o TASM, o TLINK e o EXE2BIN
ou em alternativa
o A86
```

Deverá dispor ainda de um editor de texto que escreva texto não formatado (ou ASCII puro). O EDIT.COM que acompanha o MS-DOS 5.0 serve muito bem (algum leitor que se sinta à vontade com o EDLIN.COM também o poderá utilizar, embora hesitemos em o recomendar).

O exemplo foi elaborado para ser assemblado sem qualquer modificação quer pelo MASM, quer pelo TASM, quer ainda pelo A86. Neste exemplo, quem dispuser do A86 fica com a vantagem acrescida de prescindir da fase de ligação, quer pelo LINK quer pelo TLINK, contudo fica o leitor alertado que nem sempre tal é possível (seria efectivamente bom de mais).

O EXE2BIN também é normalmente dispensável com ficheiros de estrutura .COM para quem utilize a dupla TASM, e TLINK com o *switch* **/t**.

Com estas ideias bem claras vamos então apresentar um esqueleto que possa servir para assemblar qualquer programa com estrutura .COM.

Ei-lo:

```
codigo SEGMENT
ASSUME cs:codigo, ds:codigo, es:codigo,
ss:codigo
ORG 100h

codigo ENDS
END inicio
```

Este é o esqueleto clássico, ou obtido com **directivas convencionais**. O MASM e o TASM actuais aceitam alegremente um outro tipo de directivas designadas por **directivas simplificadas** e que são baseadas nos modelos de memória das linguagens de alto nível.

Para os programas de estrutura .COM seria qualquer coisa como:

```
.MODEL TINY
.CODE
ORG 100h

END inicio
```

O A86 não compreende este último tipo de directivas, contudo e pelo menos na versão que dispomos, assembla sem erro (aliás podíamos ter substituído MODEL TINY e CODE por quaisquer outras palavras) pois este «assume» muito filosóficamente pelo **ORG 100h** (ORiGin 100h) que se trata de um .COM.

Mas referindo-nos de novo ao esqueleto clássico, fique consciente que o pseudo-operador para a definição de um segmento de programa inicia-se sempre com:

```
nome SEGMENT
[alinhamento][combinação][uso][classe]
```

e termina com:

```
nome ENDS
```

Os programas de estrutura .COM só têm um segmento de programa, mas os ficheiros de estrutura .EXE poderão ter tantos quantos necessitemos ou desejemos.

As opções entre parêntesis rectos normalmente não são utilizadas com ficheiros de estrutura .COM; de qualquer modo aqui ficam as principais pois são importantes para os ficheiros de estrutura .EXE (para o assemblador, um ficheiro de estrutura .COM é um caso particular de um ficheiro de estrutura .EXE), conforme veremos no próximo artigo.

Opções de *alinhamento*:

PARA – É o *default* e informa o assemblador que o segmento de programa se deve iniciar em um parágrafo (múltiplo de 16 bytes).

BYTE – O segmento de programa pode-se iniciar em qualquer endereço.

WORD – O segmento de programa inicia em endereço par.

PAGE – O segmento de programa inicia em endereço múltiplo de 256.

– Opções de *combinação*:

AT "expressão" – Informa que esse segmento está localizado no parágrafo definido pela "expressão". O assemblador não gerará código para esse segmento; esta opção apenas visa facilitar a referenciação de dados existentes em endereços fixos da memória.

PRIVATE – É o *default*, e faz com que o assemblador não combine esse segmento com outro mesmo que ambos tenham o mesmo nome.

PUBLIC – Todos os segmentos com o mesmo nome são reunidos de modo a ficarem a formar um único segmento. Esta opção é normalmente utilizada quando se assemblam separadamente vários módulos de um programa que deverão ser combinados pelo *linker*.

STACK – Quando esta opção é referenciada, logo após o carregamento em memória do programa o registo de segmento **SS** apontará para esse segmento de programa e o registo **SP** apontará para o fim dele.

– Opções de *uso:*

São utilizadas para programas que executam apenas em processadores 80386SX ou superior, pelo que fique o leitor só a saber que existem e que um dia pode vir a precisar de as usar.

– Opções de *classe:*

Controlam a ordem e o lugar dos segmentos em memória. Segmentos com a mesma classe são colocados juntos. O nome da classe é rodeado por apóstrofes ou aspas. Alguns nomes usuais para as classes são CODE, DATA e STACK, embora quem tenha experiência em linguagens de alto nível talvez já tenha lido outros nomes mais «arrevezados» como BSS e CONST.

Finalmente, é sempre necessário também informar o assemblador do término de um segmento de programa através de:

```
nome ENDS
```

Olhando novamente para o nosso esqueleto clássico, vemos um dos pseudo-operadores que mais «má fama» têm, o célebre **ASSUME**.

Imagine o leitor que «estava na pele» do assemblador, e que lhe aparecia a instrução *assembly* seguinte:

```
mov ax, [ 200h ]
```

o leitor sabe, (pois já o referimos) e o assemblador também sabe muito bem, que o objectivo da instrução *Assembly* acima é carregar no registo **AX** o conteúdo da posição de memória de deslocamento **200h** dentro do segmento de dados **DS**.

Efectivamente, ele sabe muito bem isso, o que ele pode ter dúvidas, e ao ponto de o fazer hesitar na produção do código máquina adequado, é para qual dos segmentos do programa o registo de segmento **DS** está a apontar naquele momento. E aqui surge uma das razões principais da necessidade do ASSUME: ajudar o assemblador na produção de código máquina.

E agora dirão alguns leitores, que sabem muito bem ou têm ouvido dizer, que o A86 dispensa os ASSUME e, por conseguinte, este assemblador parece mais evoluído que o MASM e o TASM.

Efectivamente o A86 poderá com alguma segurança dispensar os ASSUME em programas de estrutura .COM (área em que o A86 é especialista).

E as versões mais recentes do MASM e do TASM também dispensam os ASSUME, normalmente sem perigo, em ficheiro de estrutura .COM...

Em resumo, é de boa política manter sempre os ASSUME, mas quem lhes for adverso poderá muitas vezes (existem excepções) eliminá-los em ficheiros de estrutura .COM.

Finalmente, e olhando mais uma vez para o esqueleto clássico, vemos que tudo termina com um aparentemente contraditório:

```
END inicio
```

O operando "inicio" (poderia ter qualquer outro nome) da directiva **END** é o nome de um *label* (ou *procedure*) definido algures dentro do programa e que marcou o ponto de entrada do mesmo. Nos ficheiros de estrutura .COM a primeira instrução *Assembly* deve estar antecedida desse *label* ou então pertencer a um *procedure* com esse nome (conferir pelo nosso programa). E já dissemos ao leitor o essencial sobre esqueletos para produzir tantos ficheiros .COM quantos deseje.

Vamos agora cumprir a segunda parte da nossa missão e tentar explicar o que fazem as mnemónicas das instruções *Assembly* do nosso programa.

AS MNEMÓNICAS

As instruções que constituem o programa são compostas por mnemónicas das mais frequentemente utilizadas pela linguagem *Assembly*, e, tão vulgares elas são, que seria possível construir muitos outros programas diferentes restringindo-nos apenas a este pequeno e limitado



grupo. Todas as instruções *Assembly* são compostas pela mnemónica propriamente dita seguida de zero, um ou dois operandos.

Uma instrução com zero operandos é, por exemplo, a instrução **RET.**

Quando o processador encontra esta instrução, retira do topo da pilha um *word* e coloca-o no registo **IP**. O resultado é a execução ser transferida para a instrução apontada pelo valor do registo **IP** dentro do segmento de código **CS**.

O oposto da instrução **RET** é a instrução **CALL *Subrotina.*** Esta instrução guarda na pilha o valor de deslocamento da instrução que se lhe segue dentro do segmento de código **CS** e deposita no registo **IP** o valor do deslocamento do ponto de destino.

Nota: Se o *procedure* alvo fosse do tipo *far,* o valor **CS** também iria para a pilha antes do processador depositar no registo **CS** o valor do novo segmento.

Como se vê, as instruções **RET** e **CALL *Subrotina*** proporcionam dois meios de alterar o fluxo de execução de um programa, mas existem outras alternativas:

– O salto incondicional **JMP *label***

Aqui a transferência do fluxo de execução é feita carregando sem mais delongas no registo **IP** o valor do deslocamento de *label* dentro do segmento **CS** (em *procedures near*).

– O salto condicional **J(condição) *label***

No nosso exemplo vemos a variante **JZ** (salta na condição de o bit **Zero Flag** ser **1**).

Muitas outras possiblidades existem tais como **JC** *(jump if carry flag)*, **JA** *(jump if above)*, **JO** *(jump if overflow flag)* , todas elas relacionadas com o estado das várias *flags*, excepto uma: **JCXZ (jump if CX = 0)**.

A mnemónica que mais se utiliza, não só neste mas em quase todos os programas, é a **MOV**, e apresenta-se em várias variantes consoante o modo de transferência\endereçamento que está a ser efectuado e este pode ser por exemplo:

–Entre registos
Um registo recebe o valor existente em um outro.
Como é o caso de: **MOV AL,CH**

–Dado Imediato para o Registo
É transferido para um registo um dado, presente no segundo operando da instrução. E aqui é o caso de: **MOV AH, 2Ch**

– Directo do Registo para a Memória
Um endereço de memória (dado normalmente pelo nome de uma variável) recebe o valor existente num registo. Como em: **MOV [MESMO_SEG], DH**

Nota: Alguns assembladores dispensam os parêntesis rectos à volta do nome das variáveis.

– Do acumulador para uma posição de memória indexada pelos registo SI ou DI.
É o que se passa com o nosso: **MOV [SI], AL**

Repare agora na instrução do programa:

```
MOV SI, OFFSET [HORAS + 1]
```

OFFSET é um operador interno do assemblador que nos permite fazer com que ele calcule para nós (neste caso) o valor do deslocamento correspondente à posição de memória do segundo byte da variável HORAS (também serviria para calcular o deslocamento de *labels* se fosse caso disso).

> “Todas as instruções Assembly são compostas pela mnemónica propriamente dita seguida de zero, um ou dois operandos...”

Outras duas mnemónicas muito úteis são **OR** e **XOR**.

OR executa (como seria de esperar) a operação lógica de nome "OU" bit a bit entre dois operandos e devolve o resultado no primeiro deles afectando alguns *flags*, designadamente o *Zero Flag*. Ex: OR AL, AL

Aqui,quando **AL** for nulo o resultado também é nulo, e o **Zero Flag** vem igual a **1**, o que nos permite (vd. nosso exemplo) que a instrução de salto condicional que se lhe segue nos conduza para o label **SAIDA**.

XOR executa um "OU EXCLUSIVO" bit a bit entre dois operandos. Uma das utilidades do XOR é pôr um registo a zero como em:

```
XOR AL, AL
```

A mnemónica **DEC** subtrai uma unidade ao conteúdo do operando como em: DEC SI

O CPU é capaz de efectuar divisões quer com números só positivos (mnemónica **DIV**), quer com números positivos e negativos (mnemónica **IDIV**).

No nosso programa ficámo-nos pela primeira dessas possibilidades:

DIV BL divide o conteúdo do registo **AX** pelo do registo **BL**, o quociente virá em **AL** e o resto em **AH**.

Finalmente, a mnemónica **CMP** compara dois operandos e o resultado é passado para as *flags* adequadas.

Existem quase tantas alternativas para a mnemónica CMP como para a MOV, contudo no nosso exemplo ficámo-nos pela:

```
CMP DH,[MESMO_SEG]
```

que compara o conteúdo do registo **DH** com o conteúdo da posição de memória dada pela variável MESMO_SEG dentro do segmento **DS**.

Para além das mnemónicas o leitor irá reparar que antecedemos todos os dados com **DB** (Define Byte) ou **DW** (Define Word). Tanto uma como outra são directivas para o assemblador entender que o que se segue são dados e não instruções *Assembly* propriamente ditas.

Mas vamos então ver o programa que nos serviu de base a todas as explicações que demos até aqui. Poderá encontrar o seu código fonte na disqueta que acompanha a revista. Para assemblar e ligar faça respectivamente:

```
MASM relogio;
LINK relogio;
EXE2BIN relogio.exe relogio.com

ou

TASM relogio;
TLINK /t relogio;

ou ainda

A86 relogio.asm
```

Obterá um simpático programa que faz uma quantidade realmente fabulosa de trabalho em apenas 137 bytes de código máquina. É esta uma das maravilhas do *Assembly*.

E, como no próximo artigo vamos analisar os «misteriosos» ficheiros de estrutura .EXE, contamos consigo aqui, de pé firme.

José Páscoa

D. O. S.

CLS FICHA Nº. 15

COMANDO CLS

1ª. VERSÃO DOS 2.0 DR

ÚLTIMA VERSÃO DOS 2.0 DR

NÚMERO DE ALTERAÇÕES 0

SINTAXE

CLS

GRUPO Interno

FUNÇÃO Afixação

INTERRUPTORES (DOS)

INTERRUPTORES (DR 5.0)

DESCRIÇÃO

Este comando permite limpar o seu ecrã.Pode ser usado directamente na linha de comando ou integrado num ficheiro BATCH

EXEMPLOS

CLS Limpa o ecrã

DICAS

D. O. S.

FOR FICHA Nº. 17

COMANDO FOR

1ª. VERSÃO DOS 2.1 DR 3.21

ÚLTIMA VERSÃO DOS 2.1 DR 3.21

NÚMERO DE ALTERAÇÕES 0

SINTAXE Utilizado em ficheiro batch

FOR %% variável IN (conjunto de ficheiros) DO comando %% (parâmetros)

Utilizado na linha de comando

FOR % variável IN (conjunto de ficheiros) DO comando % (parâmetros)

GRUPO Interno

FUNÇÃO Procedimentos (Batch)

INTERRUPTORES (DOS)

INTERRUPTORES (DR 5.0)

DESCRIÇÃO

Permite a execução repetitiva de um comando.
É tipicamente um comando utilizado em ficheiros batch, mas pode ser igualmente utilizado na linha de comando.

EXEMPLOS SINTAXE QUANDO UTILIZADO EM BATCH

FOR %%F IN (DESPESAS.WKS DESPESAS.DOC) DO COPY A: /V

Copia os ficheiros DESPESAS.WKS e DESPESAS.DOC para o drive A: e verifica a cópia executada de cada vez que o ficheiro Batch onde se encontra esta linha de comando é invocado.

SINTAXE QUANDO UTILIZADO NA LINHA DE COMANDO

FOR %F IN (\CARTAS*.TXT *.DOC) DO TYPE %F

Faz o TYPE de todos os ficheiros com a extensão TXT e todos com a extensão DOC na directoria \CARTAS, directamente da linha de comando.

FOR %J IN (A B) DO DIR/P %J

Faz a listagem do drive a: e em seguida a do drive b:

DICAS

1 – A variável tem de ser apenas de uma letra. A variável pode ser referenciada pelo comando qualquer número de vezes, incluindo o zero.

2 – Se está a utilizar o MS-DOS, e deseja ver o interior de uma série de ficheiros em ASCII, o comando TYPE *.txt, não funcionará, podendo então utilizar o comando FOR para, numa só operação, poder fazê-lo.
Se está a utilizar o DR DOS 5.0, o comando TYPE *.txt já funcionará, tornando-se desnecessário utilizar o comando FOR.

3 – O conjunto de ficheiros sobre o qual vai incidir o comando, se estiver vazio, provoca a sua não execução.

4 – A linha de comando não pode exceder 127 caracteres, sob pena de provocar a paragem total do sistema.

D. O. S. — ECHO — FICHA Nº. 16

COMANDO	ECHO
1ª. VERSÃO	DOS 2.0 DR DOS 3.21
ÚLTIMA VERSÃO	DOS 3.30 DR DOS 5.0
NÚMERO DE ALTERAÇÕES	1 DOS 1 DR DOS 1
SINTAXE	**ECHO [ON] [OFF] [MENSAGEM]**
GRUPO	Interno
FUNÇÃO	Afixação (Batch)
INTERRUPTORES (DOS)	
INTERRUPTORES (DR 5.0)	

DESCRIÇÃO

Este comando permite visualizar ou não (ON/OFF) o texto constante num ficheiro BATCH.

EXEMPLOS

ECHO ON REM OLÁ ECHO COMO VAIS?

Todo este texto será visualizado no ecrã quando o ficheiro batch for invocado

ECHO OFF REM OLÁ REM COMO VAIS?

Este texto não será visualizado

DICAS

Este comando tem 4 funções distintas, a saber:
A – Activa a função de ECHO se estiver ON; B – Desactiva a função ECHO se estiver OFF; C – Mostra a mensagem que estiver na mesma linha do comando
D – Mostra se o comando está activo ou não (digitar na linha de comando)

Exemplos

A –ECHO ON B-ECHO OFF C-ECHO BOM DIA! D-ECHO
2 –A partir da versão 3.3 do DOS, se colocar no seu ficheiro BATCH,o simbolo @ antes do comando ECHO OFF, não será visualizada a sua execução.
3 –Pode utilizar este comando para testar a sua impressora.
ECHO este é um teste > com2:
Se não obtiver resultados, verifique as ligações da sua impressora.
4 –Pode utilizar este comando para testar o seu modem
ECHO atdt4937251 > com1:
4937251 é o hipotético número do modem que quer contactar.
atdt informa o modem compativel HAYES para marcar o número indicado em modo tone (no caso do modo pulse deverá introduzir o código respectivo, pelo que deve consultar o manual do seu modem) Deverá ouvir a marcação do número e a recepção da chamada.Não poderá no entanto enviar os seus dados, dada ser essa a função do software.

D. O. S. — PAUSE — FICHA Nº. 18

COMANDO	PAUSE
1ª. VERSÃO	DOS 1.0 DR 3.21
ÚLTIMA VERSÃO	DOS 1.0 DR 3.21
NÚMERO DE ALTERAÇÕES	1 DOS 5.0 1 DR 5.0
SINTAXE	PAUSE (Observações)
GRUPO	Interno
FUNÇÃO	Afixação (Batch)
INTERRUPTORES (DOS)	
INTERRUPTORES (DR 5.0)	

DESCRIÇÃO

Este comando serve para suspender a execução de um ficheiro BATCH. Pode igualmente servir para mostrar no ecrã qualquer mensagem.

EXEMPLOS

da.bat 1ª versão

```
ECHO ON
:begin
copy a: *.*
pause INSIRA OUTRA DISQUETA NO DRIVE A:
goto begin
^z
```

Este pequeno ficheiro BATCH, sempre que invocado, efectuará a cópia de todos os ficheiros no drive a: para o drive e directório onde estiver a trabalhar, suspende a sua execução mostra-lhe a mensagem que se encontra na linha do comando e após carregar em qualquer tecla voltará ao drive a: e procederá à cópia dos ficheiros existentes na disqueta aí inserida.

cda.bat 2ª versão

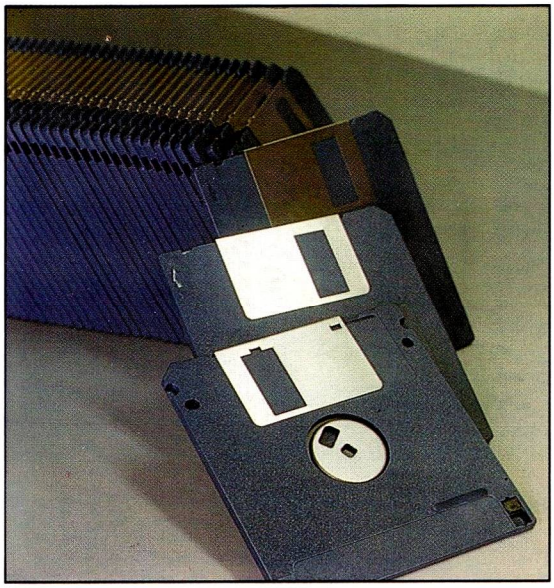
```
@ECHO OFF
:BEGIN
COPY A:*.*
ECHO INSIRA OUTRA DISQUETA NO DRIVE A:
PAUSE
GOTO BEGIN
^z
```

A diferença entre a 1ª versão e a 2ª, está na necessidade de activar o comando ECHO para que a mensagem possa ser visualizada.

DICAS

Para quebrar uma situação de LOOP, como a evidenciada no exemplo dado, deverá fazer uso da combinação de teclas CTRL+C

DISQUETAS BULK (sem marca)



10 DISQUETAS 3,5" DS/DD
Refª. D040 **Preço: 1 095$00**

10 DISQUETAS 3,5" DS/HD
Refª. D041 **Preço: 1 660$00**

«RATO» SEM FIOS Genius HiMouse



– **Liberdade de movimentos:** pode usá-lo até 2 metros do computador sem quaisquer ligações com cabos.
– **Tecnologia de infra-vermelhos:** sem interferência em sinais rádio
– **Poupa energia:** Desliga-se automaticamente quando não em uso. A unidade receptora possui sempre 2 baterias adicionais recarregadas e prontas a usar.
– **Software de primeira:** inclui o extraordinário ZSoft Paintbrush IV
– **Inclui:** Adaptador 9/25 pinos, unidade receptora, 3 baterias recarregáveis, disqueta de instalação e ZSoft Paintbrush IV.

Refª. RSF1 **Preço: 10 783$00**

10% de desconto
para assinantes
em todos os produtos da Montra
TODOS OS PREÇOS APRESENTADOS JÁ INCLUEM O IVA.

SOUND BLASTER PRO



SISTEMA PROFISSIONAL PARA RECOLHA, EMISSÃO E TRATAMENTO DE SOM, COMPATÍVEL DOS 3.0 E WINDOWS 3.0 (E SUPERIORES).

Gravação estéreo; compatibilidade mono com AdLib e Sound Blaster; Twin FM chips proporcionando 22 vozes; misturador estéreo digital/analógico FM, DAC, line-in, microfone, CD e altifalante interno de PC; software para regulação de fade, pan, e volumes direito e esquerdo, mistura com microfone; cabo Midi *standard*; *Interface* CD-ROM com conector interno para CD-Audio; porta para *Joystick*; Amplificador de microfone com AGC (Automatic Gain Control); 4 W por canal; Controlo manual de volume na placa; etc...

A embalagem Sound Blaster Pro contém: Carta Sound Blaster Pro 16 bit; Cabo RCA Audio; Cabos Midi; Disquetas de 3,5" e 5,25"; Software SPJ Sequencer; Editor de voz II; SBTALKER (software texto/voz); Software FM (orgão inteligente); reprodutor CD; demonstração MMPLAY; *drivers* para DOS e Windows.

Requisitos em Hard e Software: PC AT compatível (286 ou superior), mínimo de 512 Kb RAM, DOS 3.0 ou superior, carta EGA ou VGA (VGA recomendada)

Refª SB01 **Preço: 48 256$00**

KITS PARA LIMPEZA DE DRIVES



Ref.ª G001
Disqueta 3 ½
320$00

Ref.ª G002
Disqueta 5 ¼
320$00

Kits de limpeza de drives incluindo disqueta e frasco com o líquido de limpeza. Qualidade comprovada.

TAPETE PARA MOUSE

Tapete para Mouse de elevada qualidade. Anti-estático.
Permite-lhe uma maior precisão e facilidade no seu trabalho.

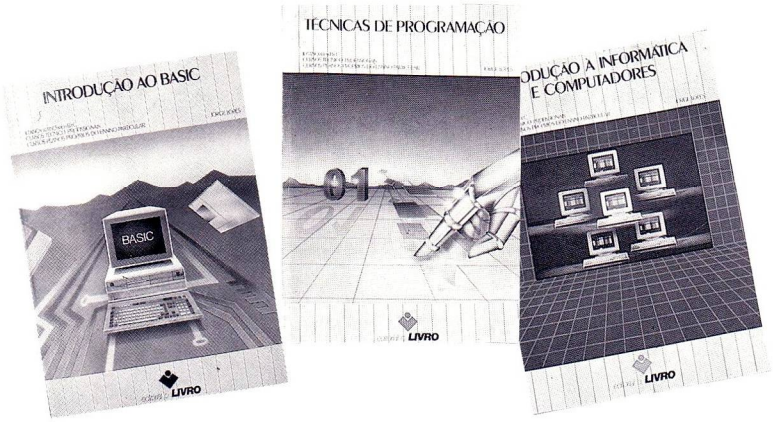
Ref.ª G005 PREÇO: 875$00

DOS 5.0 (EM PORTUGUÊS)

Conheça as novas e poderosas potencialidades do sistema operativo MS-DOS 5.0. Tudo em português.

Refª. MS01	COM DISQUETAS DE 5,25"	20 300$00
Refª. MS02	COM DISQUETAS DE 3,5"	20 300$00

LIVROS EDITORIAL O LIVRO (Em português)

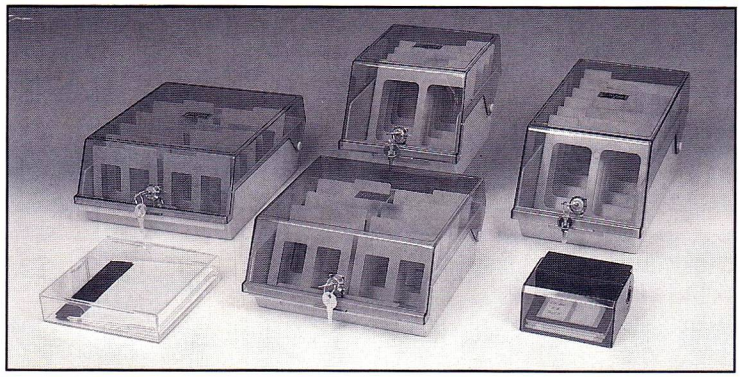


– INTRODUÇÃO À INFORMÁTICA E COMPUTADORES
REF.: L001 PREÇO: 2 000$00

– TÉCNICAS DE PROGRAMAÇÃO
REF.: L002 PREÇO: 1 750$00

– INTRODUÇÃO AO BASIC
REF.: L003 PREÇO: 2 560$00

CAIXAS ARQUIVADORAS PARA DISQUETAS



– 10 disquetas 3,5"	Ref.: H004	Preço:	227$00
– 10 disquetas 5,25"	Ref.: H005	Preço:	302$00
– 50 disquetas 5,25"	Ref.: H006	Preço:	1 033$00
– 100 disquetas 5,25"	Ref.: H007	Preço:	1 200$00
– 50 disquetas. 3,5"	Ref.: H008	Preço:	1 033$00
– 100 disquetas. 3,5"	Ref.: H010	Preço:	1 282$00



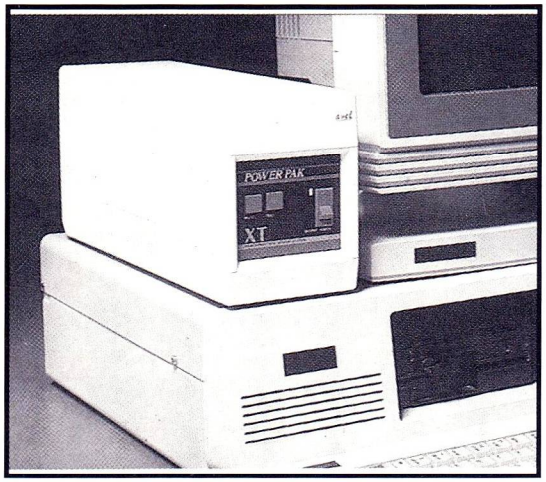
DBase III Plus
PROGRAMAÇÃO AVANÇADA

Refª. GR01 COM DISQUETA DE 5,25" 1 890$00

Refª. GR02 COM DISQUETA DE 3,5" 1 995$00

INVISUAIS: Quando solicitado forneceremos em disqueta suplementar o texto integral do livro.

POWER PAK – AT



Unidade de Estabilização e Manutenção de energia eléctrica, indicada para protecção da configuração completa de PC AT.
Bateria selada, sem manutenção.
Potência máxima: 550 VA – Autonomia: 25 minutos.

GARANTIA DE 1 ANO

Refª. PK01 PREÇO: 52 003$00

PAPEL CONTÍNUO

Embalagem de 500 folhas de papel contínuo (zebra americana) 12x9,5"

Refª. P020 Preço: 785$00

MODEMS BEST DATA



Modems de alta qualidade a preços notáveis.

Refª BD01 Modem interno – 2400 bps
W/MNP 2-7 por software V.22 e V.22 bis) **Preço: 14 945$00**

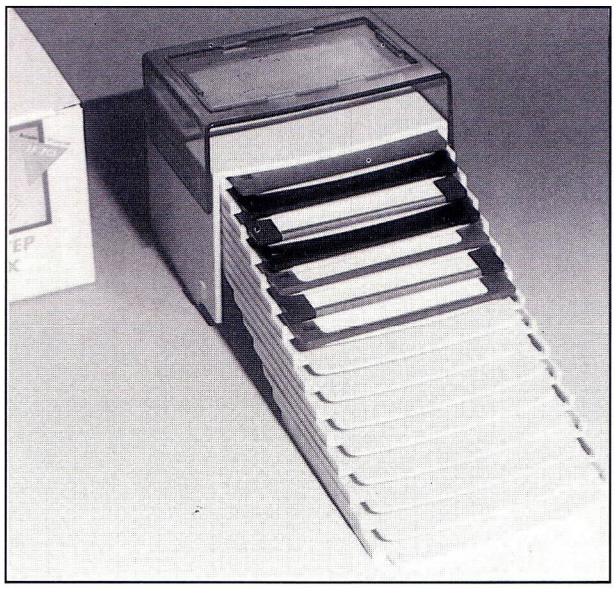
Refª BD02 Modem externo– 2400 bps
W/MNP 2-7 por software V.22 e V.22 bis) **Preço: 22 605$00**

Refª BD03 Fax/Modem interno – 9600 bps Fax – 2400 bps Modem
W/MNP 7 por software V.22 e V.22 bis) **Preço: 24 898$00**

Refª BD04 Fax/Modem externo – 9600 bps Fax – 2400 bps Modem
W/MNP 7 por software V.22 e V.22 bis) **Preço: 41 691$00**

Nota: *estes Modems aguardam registo de homologação*

CAIXA ARQUIVADORA - 15 DISQ. 3,5"



Caixa arquivadora «elástica» para o acondicionamento perfeito das suas disquetas de 3,5". Versátil e compacta.

Refª BX01 **Preço: 750$00**

BBSpooler

☎ (01) 475 22 89

Dias úteis: das 18.30 às 9.00 horas
Feriados e fins-de-semana: 24 horas

REEDIÇÃO DOS NÚMEROS ESGOTADOS (1 a 6) DA SPOOLER

Depois de esgotada a reedição dos números 1, 2 e 3 da Spooler e para satisfazer a procura manifestada por inúmeros leitores vamos tornar a imprimir os números 4, 5 e 6, agrupados num só volume.

Esta edição tem uma tiragem reduzida e não será distribuida.

O único modo de possuir um exemplar é reservá-lo, desde já, preenchendo o cupão anexo (ou fotocópia) e remete-lo para os nossos serviços:

SPOOLER MAGAZINE
Rua Duarte Pacheco Pereira,
69-A – Damaia-de Baixo
2700 AMADORA

CUPÃO DE RESERVA DAS EDIÇÕES ESPECIAIS SPOOLER 1-2-3 E SPOOLER 4-5-6

NOME .....

ENDEREÇO .....

.....

C. POSTAL ..... LOCALIDADE ..... TELEF. .....

DESEJO RECEBER A EDIÇÃO ESPECIAL

☐ «SPOOLER 1 - 2 - 3»
☐ «SPOOLER 4 - 5 - 6»

☐ COM 3 DISQUETAS DE 5 1/4" ..... 1 200$00
☐ COM 2 DISQUETAS DE 3 1/2" ..... 1 200$00

Não mande qualquer pagamento agora. A edição especial ser-lhe-á enviada à cobrança.
Aos preços indicados serão acrescidos os portes de envio e taxa de cobrança dos C.T.T.

Assinatura

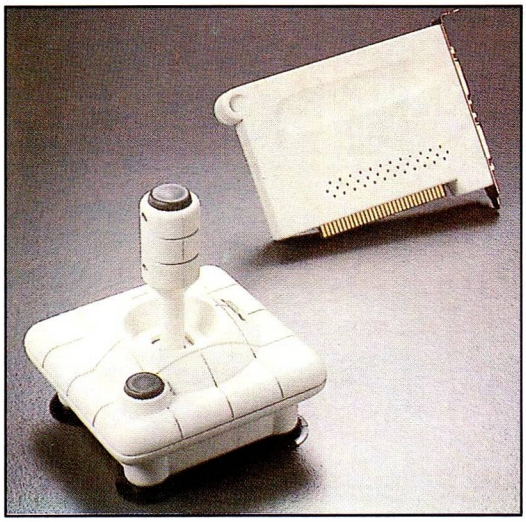
QUICKJOY M-5



Compatível para PC/XT/AT este joystick de *design* ergonómico é o ideal para os seus jogos de acção.
A placa pode ser adquirida em separado.

Refª G004 **Preço: 3 951$00**

JOYSTICK + PLACA SV 203



Joystick para compatíveis PC/XT/AT e placa. Óptimo desempenho e *design* atractivo. Um novo dinamismo nos seus jogos predilectos. Um preço incrível!

Refª G003 **Preço: 6 334$00**

DISQUETAS PLATINUM PRÉ-FORMATADAS



Tenha sempre à mão disquetas de qualidade já formadas.

Caixas com 10 disquetas pré-formatadas:		
– 3,5" DS/DD	Ref.: PL01	Preço: 1 360$00
– 3,5" HD	Ref.: PL02	Preço: 2 290$00

PLACA QS-120 PARA JOYSTICK

Placa para joystick

Refª G04A **Preço: 3 861$00**

RATO MIC - MI-260



Regressado ao nosso convívio cá está o rato MIC, com nova embalagem e muito, muito mais barato!

Características: 260-2600 DPI, Compatível Microsoft e Mouse System, com comutador no corpo do rato. 3 teclas.
Inclui: Software de instalação e teste, ficheiros de ajuda (em inglês), ficha adaptadora 9/25 pinos.

Refª MIC1 **Preço: 2 500$00**

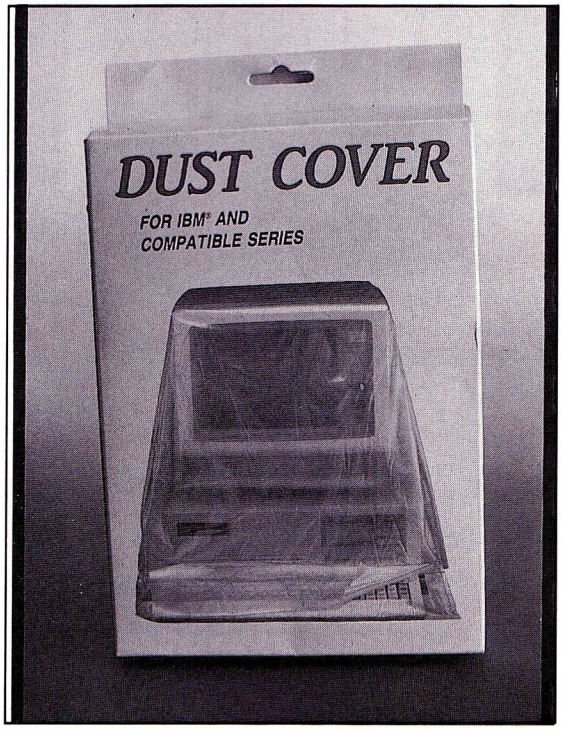
FITAS, TONNERS E TINTEIROS PARA IMPRESSORAS

PARA TODOS OS TIPOS DE IMPRESSORAS (E MÁQ. DE ESCREVER) MARCAS ORIGINAIS E COMPATÍVEIS.

Indique no cupão de encomenda a referência que pretende (ou a marca e modelo da impressora)

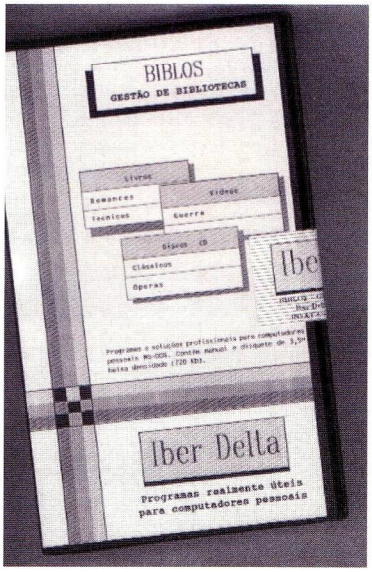
TEMOS OS MELHORES PREÇOS.

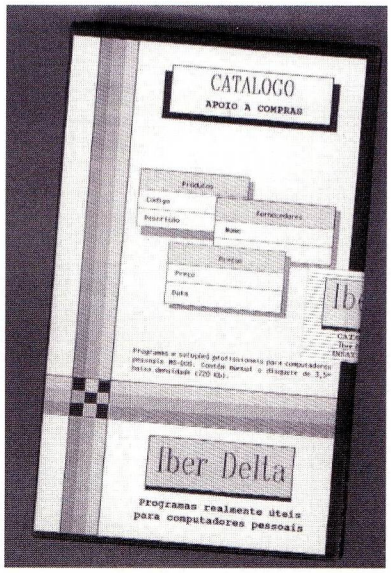
CAPA PROTECTORA PARA PCs E IMPRESSORAS



Capa protectora para computadores em plástico especial reforçado para a completa protecção do seu equipamento contra fumos, poeiras, humidade e sujidades de toda a espécie.
Adaptável a qualquer tipo de PC e impressoras.

Ref.: G006 – Capa para PC	**1 674$00**
Ref.: G007 – Capa p/ imp. 80 c.	**913$00**
Ref.: G008 – Capa p/ imp. 132 c.	**913$00**





SOFTWARE IBER

Programas e soluções profissionais para compatíveis MS-DOS.
Não requerem conhecimentos de Lotus 1-2-3 ou de MS-DOS.

Refª IB01*	**CAPITAL-P – FINANÇAS PESSOAIS.**
Refª IB02*	**CAPITAL-E – TESOURARIA DE PEQUENA E MÉDIA EMPRESA.**
Refª IB03	**CORREIO-E – REGISTO E CONTROLO DE CORRESPONDÊNCIA..**
Refª IB04*	**BALANÇOS – ANÁLISE DE BALANÇOS.**
Refª IB05	**BIBLOS – GESTÃO DE LIVROS, VIDEOS E DISCOS.**
Refª IB06	**CATÁLOGO – APOIO A COMPRAS – PRODUTOS FORNECEDORES – PREÇOS**

Em disquetas de 3,5" (720 Kb) PREÇO: 4 900$00

***Versões para Lotus 1-2-3 – Rev 2.10 ou mais recente**

CUPÃO MONTRA SPOOLER

Recorte ou fotocopie o cupão e remeta-o para:

SPOOLER MAGAZINE
Rua Duarte Pacheco Pereira, 69-A – Damaia
2700 AMADORA

ESTIMADO LEITOR:
Tentamos entregar as suas encomendas o mais rápido possível.
Quando um número da **Spooler** sai ao público, somos «bombardeados» com muitos milhares de pedidos.
Não desespere se não receber a sua encomenda no dia seguinte e **por favor não a devolva.** Os preços por nós praticados (quase isentos de lucro) não permitem o encargo adicional de ter que pagar os portes do retorno da encomenda. Comunique-nos se pretender, à *posteriori*, desistir da encomenda.

CUPÃO DE ENCOMENDA – MONTRA SPOOLER

Cliente nº.:
A PREENCHER PELOS NOSSOS SERVIÇOS

Desejo receber à cobrança (com os **custos acrescidos de portes e taxas**) os seguintes produtos:

Quant.	Refª.	Designação	Preço
		TOTAL	

NOME .....
MORADA .....
C. POSTAL ..... LOCALIDADE .....
TELEFONE ..... ❑ NÃO SOU ASSINANTE ❑ SOU ASSINANTE Nº .....

Spooler 19

FREE-SOFT E SHAREWARE

O valor indicado não paga os direitos destes programas, mas sim, apenas as próprias disquetas e o trabalho de cópia e distribuição. Será o utilizador dos programas tipo «shareware» a ter a responsabilidade de efectuar o respectivo pagamento, directamente aos autores, nos termos e condições por eles propostos. A Spooler limita-se a distribuir esses programas, devidamente testados e na sua versão integral.

As referências anteriores a SH112 podem ser consultadas no número 18 da Spooler.

NOVAS VERSÕES RELATIVAS A REFERÊNCIAS ANTERIORES:

A ref.ª SH68 passa a dispor de nova versão (3.00) do famoso V860 bem como a ref.ª SH58 que passa a dispor das recentes versões 99 do SCAN e CLEAN.

SH112 – GIF's Alta resolução
PLAT – Uma natureza...morta 640x512x16
SACKS – Uma natureza...viva 640x480x256

SH 114 – Jogos
ORION – Orion Odyssey
1MARS – Os monumentos de Marte
Refª. SH 113 – 1 disqueta de 5,25" ... 500$00
Refª. SH 114 – 1 disqueta de 3,5" ..... 600$00

SH 116 – Jogos
WHEELS2 – Roda da Fortuna para DOS
CRYSTAL – Crystal Caves
TK – Solitário para DOS
Refª. SH 115 – 1 disqueta de 5,25" ... 500$00
Refª. SH 116 – 1 disqueta de 3,5" ..... 600$00

SH 118 – FRAINT16
FRACTINT V 1.6 – Gerador e manipulador de fractais. Conjuntos de Julia, Mandel e muitos outros tipos. Animação em 3D. Grava em formato GIF.

SH 120 – Colecção Pintura – VELASQUEZ
ESOPO, INFANTE, JARDINS, COMTE, MENINAS

SH 122 – Dicionários para TRADUTOR.COM (Ver artigo neste número da Spooler)
ING_PORT.DIC – Dic. de Inglês-Português
FRN_PORT.DIC – Dicionário de Francês-Português
SIGLAS.DIC – Glossário de siglas, abreviaturas e acrónimos

SH123 – JOGOS
MAHJONG – O famoso jogo chinês
TZERO – Uma aventura aliciante
SPECTRUM – 18 jogos para o simulador do famoso Spectrum

SH124 – Utilitários diversos
VSUM – O famoso Virus Information Summary List de Patricia Hoffman
CSHOW844 – Visualizador de imagens digitalizadas (GIF, PCX, RLE, RIX, BMP, TIFF, IMG e Dr.Halo).
ALCHEMY – Conversor/Compressor de imagens digitalizadas. Inclui duas excelentes imagens.

SH125 – Utilitários Diversos
FPROT203 – Anti-vírus – Detecção e proteção do sistema.
DBACKL1 – Apagamento selectivo de ficheiros no disco.
RTM211 – Gestor TSR de tarefas, compromissos, etc.
ARCM130 – Interface para compressão/arquivo de ficheiros.
MODEMD40 – Diagnóstico de Modems e Portas de Comunicação.

SH126 – Utilitários para Windows
1470ICON – 1470 ícones para Windows!
CDBW – DataBase ToolKit para Windows
ELANA10 – Configuração para Windows 3.0
FP104 – FractalPaint - Fractais e outros desenhos
SCREENPC – Um ScreenSaver for Peace

SH127 – Utilitários para Windows
PBBSW152 – PowerBBS – Uma BBS para Windows
TOILET – Um W.C. para o Windows
BLACKO22 – Aprenda a utilizar o rato jogando o BLACKOUT
DRAWICON – Desenho e construção de ícones. Inclui biblioteca

SH128 – Utilitários para Windows
UNICOM30 – Programa de comunicações Versão 3.0
NCOMP121 – Viaje pelos E.U.A. e crie os seus próprios mapas
VGA – Drivers para emulação tipo MAC, NeXt ou OS Motif.

A PARTIR DESTE NÚMERO O *SOFTWARE* AQUI REFERENCIADO SERÁ FORNECIDO EM DISQUETAS DE 3,5" AO PREÇO DE 600$00 POR UNIDADE. AINDA PODEREMOS FORNECER DISQUETAS DE 5,25" - 360 Kb AO PREÇO DE 500$00 POR UNIDADE.

ASSINATURAS

Spooler

Recorte ou fotocopie o cupão, preencha-o com letra bem legível e remeta-o, acompanhado do respectivo pagamento em cheque ou vale de correio, para:

SPOOLER MAGAZINE
R. Duarte Pacheco Pereira, 69-A
Damaia – 2700 AMADORA

*Os números anteriores da **Spooler** podem ser encomendados à cobrança utilizando o cupão da **Montra Spooler**. Ao preço de capa serão acrescidos os custos dos portes e taxas de cobrança.*
Se preferir disquetas de 3,5", isto custar-lhe-á mais 140$00 por exemplar.

NOTA: Os números 1 a 6 e o 10 e 11 da **Spooler** encontram-se esgotados.

Spooler 20

CUPÃO DE ASSINATURA

Assinante nº:
A PREENCHER PELOS NOSSOS SERVIÇOS

NOME .....
DATA DE NASCIMENTO* ____ / ____ / ____
PROFISSAO* .....
TRABALHA COM COMPUTADORES?* SIM☐ NÃO☐
SE SIM ESPECIFIQUE A MARCA E MODELO* .....
ENDEREÇO .....
.....
C. POSTAL ..... LOCALIDADE ..... TELEF(S) .....

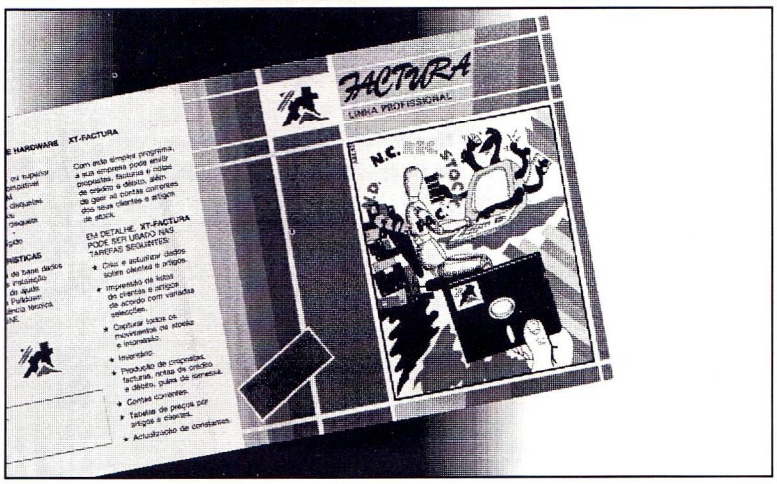
DESEJO ASSINAR A SPOOLER A PARTIR DO N.º 21 (inclusive) E DURANTE 12 NÚMEROS.

☐ **MODALIDADE COM DISQ. DE 5 ¼" ..... 5 500$00**
☐ **MODALIDADE COM DISQ. DE 3 ½" ..... 6 000$00**

PARA PAGAMENTO DA ASSINATURA ENVIO :
☐ O CHEQUE Nº. ..... S/ O BANCO .....
☐ O VALE DE CORREIO Nº. .....

** A resposta aos itens assinalados com um (*) é opcional.*

PROGRAMA PARA FACTURAÇÃO «FACTURA»



Este programa de fácil utilização, instalação simples, textos de ajuda claros e precisos, menus *pulldown*, obedece a todos os requisitos legais em vigor e permite-lhe:

- Criar e manter actuais dados sobre clientes e artigos;
- Imprimir listagens de clientes e artigos de acordo com variadas opções de selecção;
- Manter actualizado o movimento e inventário de stocks;
- Produzir e imprimir propostas, facturas, notas de crédito, notas de débito e guias de remessa.
- Contas correntes, tabelas de preços por artigos e clientes.

OBEDECE A TODAS AS DISPOSIÇÕES LEGAIS EM VIGOR

Com garantia e assistência técnica ON-LINE. Produzido por XT-Software, Lda.

Refª FT01 **Preço: 17 400$00**

PROGRAMA DE GESTÃO CONTABILÍSTICA «MAXICONTA»

MAXICONTA é um programa para Contabilidade, projectado para o utilizador que necessite de aplicações simples e eficazes.

PRINCIPAIS CARACTERÍSTICAS:

- Criação e actualização de contas por centros de custo, diários, contas e tipos de lançamento.
- Introdução de lançamentos e listagens por código; data; extractos; centro custo; tipo lançamento; diário; n.º de documento.
- Balancetes do Razão; Geral; Centro Custo; Tipo Lançamento.
- 12 meses em aberto.
- Balanço geral.
- Abertura e fecho de ano.
- Utilitários diversos incluindo formatação de disquetas, cópias de segurança. reposições, indexação recálculo e meses abertos.

OBEDECE A TODAS AS DISPOSIÇÕES LEGAIS EM VIGOR

Com garantia e assistência técnica ON-LINE. Produzido por XT-Software, Lda.

Refª MX01 **Preço: 23 200$00**

PROGRAMA PARA FACTURAÇÃO, «SUPERFACT»

Aplicação de Gestão Comercial Integrada, multi-empresa e multi-utilizador.

PRINCIPAIS CARACTERÍSTICAS:

- Programa multi-empresa
- Multi-utilizador
- Gestão de stocks
- Gestão de clientes, fornecedores e artigos.
- Gestão de contas correntes de clientes, fornecedores e artigos
- Emissão de todos os documentos relativos a facturação
- Listagens de gestão relativas a clientes, fornecedores e vendedores, avenças, inventário, pagamentos, recebimentos e saldos.
- Acumulados de vendas diárias e mensais
- Utilitários
- Impressão em papel pré-timbrado (2/3 factura e 1/3 recibo)

OBEDECE A TODAS AS DISPOSIÇÕES LEGAIS EM VIGOR

Com garantia e assistência técnica ON-LINE.
Produzido por XT-Software, Lda.

Refª FT02 **Preço: 58 000$00**

«AGENDA»

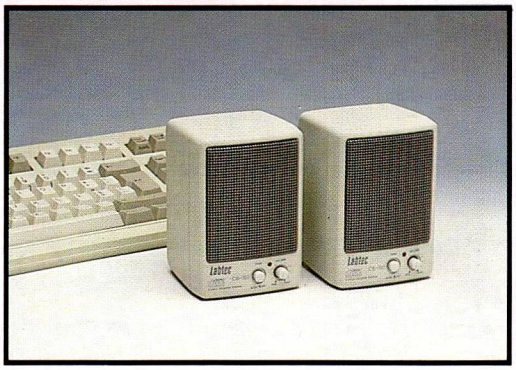
AGENDA é a solução para quem pretender organizar e rentabilizar o seu tempo.

PRINCIPAIS CARACTERÍSTICAS:

- **Lista de endereços e telefones:** Número ilimitado de fichas contendo informações diversas sobre empresas ou personalidades.
- **Agenda:** Marcação e visualização de entrevistas, chamadas telefónicas e tarefas.
- **Contas bancárias:** Simples mas eficiente controlo de todas as suas contas bancárias, incluindo gestão de cheques pré-datados e letras.
- **Editor de texto:** Faça os seus próprios documentos, sem necessidade de processadores de texto convencionais. Ideal para mailings, cartas, etc.
- **Etiquetas:** Emissão de etiquetas a partir da base de dados de endereço e com selecções variadas.
- **Plano de tesouraria:** Controlo de pagamentos e recebimentos.
- **Utilitários:** Calculadora, calendário, bases de dados de utilidade pública.

Com garantia e assistência técnica ON-LINE.
Produzido por XT-Software, Lda.

Refª AG01 **Preço: 13 920$00**

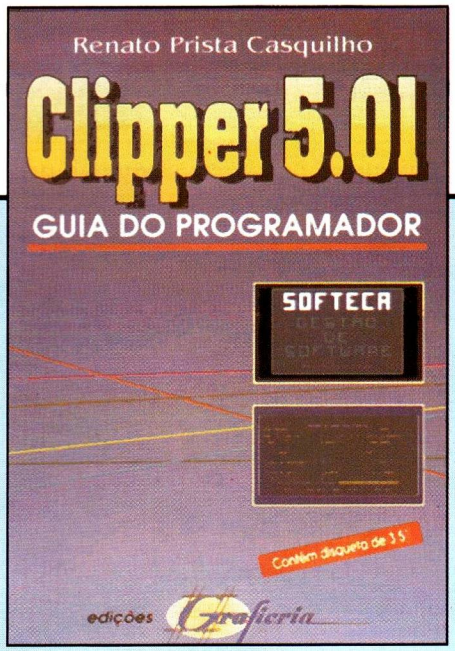


SISTEMA DE AMPLIFICAÇÃO COM ALTIFALANTES PARA COMPUTADOR

O melhor som estéreo para o seu PC

- Sistema de amplificação para Computador.
- Estereofónico com controlo individual de volume
- Alimentação interna por baterias 4 «C» (não incluídas) ou externa por DC 6V.
- Funciona com ou sem placas de som..
- Rated Output Power: 7,2 W 3,6 W x 2 P.O.P.M.
- Frequência resposta: 100-15 000Hz
- Peso (sem baterias): 355 g x 2
- Dimensões: 85 x 120 x 90 mm

Refª CL01 **Preço: 5 278$00**



Clipper 5.01 – Guia do Programador

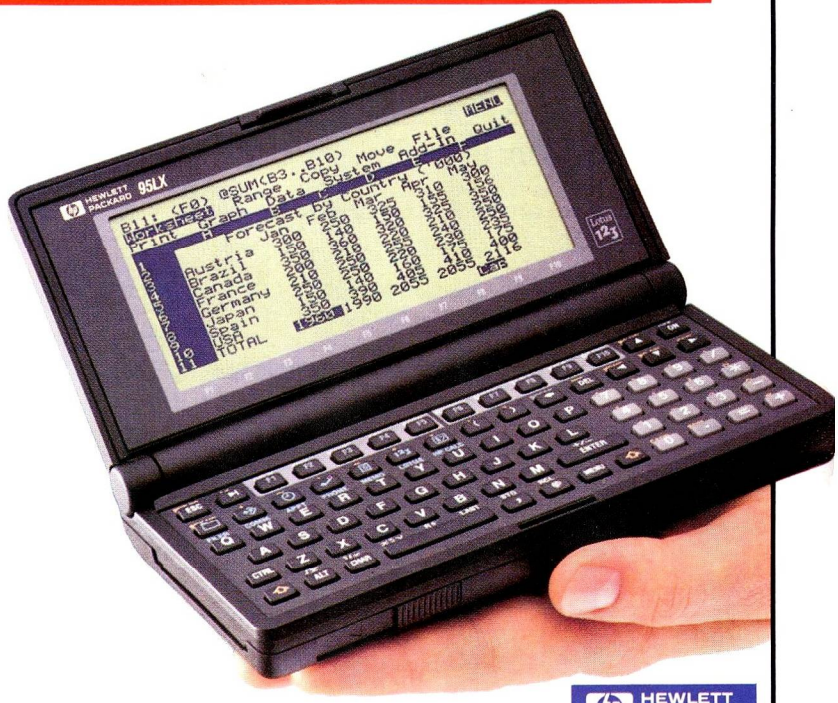
Este livro pretende ir ao encontro das preocupações e dificuldades encontradas por quantos pretendem iniciar-se em qualquer linguagem de programação. Destina-se, essencialmente, aos programadores de dBASE que pretendem dar o «salto» qualitativo para o poderoso ambiente facultado pelo Clipper 5.01, capaz de gerar aplicações sofisticadas, elegantes, rápidas e autónomas. Nele são analisados e exemplificados todos os comandos, funções e declarações, incluindo as novidades e correcções introduzidas pela versão 5.01.
A obra é acompanhada por uma disquete de 3 1/2", cujo conteúdo constitui uma demonstração das inúmeras capacidades desta linguagem.

Refª GR03 **Preço: 3 600$00**

O SEU PC DE BOLSO HP 95LX

Graficria pub

- 512 Kb a 1 Mb de ROM
- *Software*:
  - MS DOS®3.22
  - Lotus®1-2-3®, versão 2.2
  - Comunicação de dados
  - Editor de texto
  - Gestão de ficheiros
  - Calculadora H. P.
  - Agenda
  - Lista telefónica
- 512 Kb de RAM expansíveis até 1 Mb
- Ligação a PC, impressora, *modem*, etc.



Contacte:

DIMECRO
Representações Comerciais e Técnicas, Lda.
Rua Joaquim António de Aguiar, 41 - 6º Esqº – 1000 LISBOA
Telefones: 315 76 12/3/4/5 – Fax: 315 76 12

HEWLETT PACKARD
Distribuidor

SENSACIONAL CAMPANHA

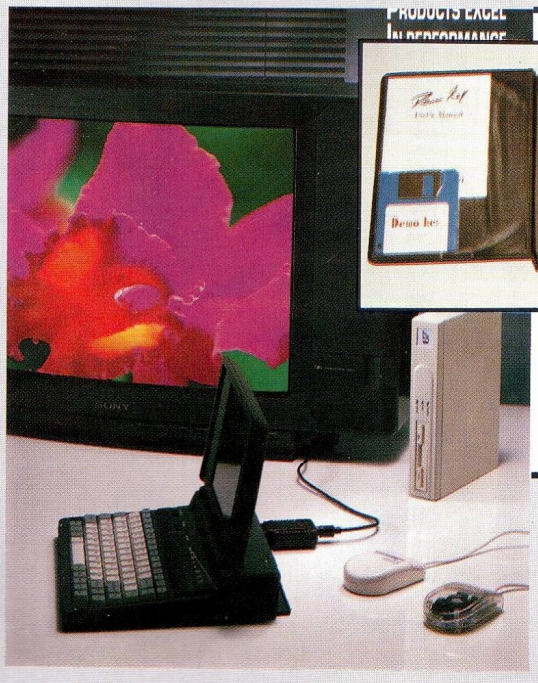
inves

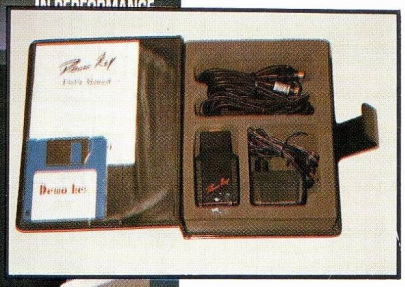


INVES NB-320S (Notebook)

- Processador 80386SX-33 MHz
- 2 a 4 Mb RAM
- 2 Portas série, 1 Paralela
- Porta 8 pinos para Handy Scanner
- Drive 3½" – 1,44 Mb
- Disco rígido 2½" – 60 Mb
- Ecrã cristais líquidos contra-iluminados VGA
- 640x480, 32 tonalidades de cinzento
- Socket para 80387SX
- Teclado 81 teclas, 12 funções, português
- Dimensões: 290x220x53 mm

O PODER DUM NOTEBOOK 386 A 33 MHz POR APENAS 259 000$00*!





DEMOKEY (VGA TO VIDEO)

Dispositivo que permite transformar em sinal vídeo qualquer saída VGA de computador.
Ideal para os possuidores de portáteis ou *notebook*, ou de monitores VGA monocromáticos, que assim podem visualizar os seus programas confortavelmente no monitor colorido do seu receptor de televisão ou videogravador.

- Resoluções até 640x480, 256 cores
- Saída vídeo NTSC ou PAL
- Portas de saída AV Vídeo e S-VHS Vídeo

Preço:
32 000$00*



SAT
SERVIÇO ASSISTÊNCIA TÉCNICA
TODAS AS MARCAS
Tel. 778 73 24
Fax 778 53 01

REPRESENTANTE OFICIAL:
INVESMÁTICA, LDA.
Rua Abel Botelho, 8-A – 1500 LISBOA
Telefs.: 778 73 24/6 Fax: 778 53 01

* TODOS OS PREÇOS APRESENTADOS ACRESCEM DE 16% DE IVA.

Graficria PUB

Consuldata

SERVIÇOS DE INFORMÁTICA, LDA.

UM MUNDO EM CONSUMÍVEIS PARA INFORMÁTICA

AS MELHORES MARCAS
O MELHOR ATENDIMENTO
A ENTREGA MAIS RÁPIDA
OS MELHORES PREÇOS!!!

DISQUETES DE 3", 3,5", 5,25" E 8"

DATA CARTRIDGES (STREAMTAPES)
DE 10, 20, 45, 60, 150, 250, 320,
525 MEGABYTES E 1,3 GIGABYTES

DISK CARTRIDGES E DISK PACKS
PARA GRANDES SISTEMAS

PASTAS PARA ARQUIVO DE FORMULÁRIOS
DE 9,5x12, 12x9,5, 12x15,5 E 15,5x12
E RESPECTIVOS SEPARADORES

FILTROS ANTI-MAGNÉTICOS E ANTI-ESTÁTICOS
EM REDE E VIDRO PARA TODOS OS TIPOS
DE ECRÃS DE COMPUTADOR

MOBILIÁRIO PRÓPRIO
PARA CENTROS DE INFORMÁTICA

TAPE CARTRIDGES PARA IBM 34800
INCLUINDO CARROS DE TRANSPORTE
E MÓVEIS ESPECIAIS DE ARQUIVO

D. D. S. - 4 MM (1,3 E 2,0 GIGABYTES)
- 8 MM (2,3 GIGABYTES)

E MUITO MAIS!
PARA MAIS INFORMAÇÕES CONTACTE-NOS
OU SOLICITE O CATÁLOGO CONSULDATA

Consuldata - Serviços de Informática, Lda.
Rua Quinta do Almargem, 7 A/B - 1300 LISBOA
Telef.: 364 98 00 Fax: 364 91 08