BIBLIObase

Linguagem de formatação do CDS/ISIS

Ferramenta de desenvolvimento e exploração de base de dados CDS/ISIS

Janeiro
2002

Ficha técnica

Título	BIBLIObase : Linguagem de formatação : ferramenta de desenvolvimento e exploração do CDS / ISIS
Autores	António Manuel Freire e Licínia Santos
Revisão	Ana Domingos
Edição	3ª edição
Copyright	© BIBLIO*soft* ™

Mini-micro CDS/ISIS, © UNESCO

Para mais informações consulte: **http://www.bibliosoft.pt**
Email: **bibliosoft@esoterica.pt**

Revisão 2.0 (Setembro 2000)

Catalogação recomendada

FREIRE, António Manuel, 1965-

BIBLIObase : Linguagem de formatação : ferramenta de desenvolvimento e exploração do CDS/ISIS / António Manuel Freire e Licínia Santos ; rev. Ana Domingos. – 2ª ed. revista. - Lisboa : BIBLIOsoft, 2000. – 61 p.

CDU 025.173:681.3.06(076)

I - Santos, Licínia, 1966-
II – Domingos, Ana, 1962-

Sumário

À memória de
Mr. Giampaolo DelBigio

CAPÍTULO 1 : APRESENTAÇÃO

Desde 1887, altura em que se iniciou a distribuição em Portugal, que o número de utilizadores do CDS/ISIS, desenvolvido pela UNESCO, cresceu consideravelmente.

Independentemente dos produtos criados com este software, uma característica muito especial atrai qualquer utilizador minimamente sensibilizado para a exploração deste tipo de bases de dados - a possibilidade de ler e formatar o conteúdo da base de dados de uma forma simples.

Apesar da evolução do CDS/ISIS, a linguagem de formatação é sem dúvida um dos grandes trunfos de todos os produtos que exploram este tipo de bases de dados, como por exemplo, o CDS/ISIS para DOS, o WINISIS (versão para Windows do CDS/ISIS da responsabilidade da UNESCO), do WWWISIS, interface WEB e o ISISDLL, que permite desenvolver aplicações independentes noutras linguagens de programação. Estes dois últimos produtos são da responsabilidade da Bireme (Brasil).

O conjunto de instruções que compõem a linguagem de formatação, não está disponível em todos os produtos mencionados. Por exemplo a versão para Windows possui algumas instruções de **hipertexto**, que não podem ser usadas nas versões para DOS, WWWISIS ou ISISDLL.

A facilidade com que se podem imprimir ou mesmo converter os registos de qualquer tipo de base de dados em CDS/ISIS é uma ferramenta muito poderosa e de simples utilização.

A publicação deste manual em simultâneo com os produtos **BIBLIObase**, serve como complemento e auxiliar a todos os utilizadores que pretendam explorar as potencialidades da linguagem de formatação do CDS/ISIS aplicadas a este produto.

Este manual é uma adaptação e tradução livre do **Manual de Referência do CDS/ISIS 3.0** nunca publicado em português, e de alguns documentos relativos à interface WWWISIS da Bireme.

Apesar de "fora de tempo", gostaria de contribuir para uma melhor compreensão e conhecimento das potencialidades desta linguagem, à comunidade de utilizadores do CDS/ISIS.

O Autor

CAPÍTULO 2 : INTRODUÇÃO

A linguagem de formatação permite definir com precisão produtos, quer para visualização quer para impressão, de acordo com vários critérios, a partir dos registos criados numa base de dados CDS/ISIS.

Através desta linguagem, podem ser seleccionados um ou mais elementos de dados pela ordem desejada contidos num registo, a partir de um ou mais campos e/ou subcampos, inserir comentários e atribuir cabeçalhos, como também definir a formatação desses elementos (espaçamentos, indentação, etc.), quer na visualização quer em produtos impressos. Ao conjunto de comandos da linguagem de formatação descritos neste capítulo é dado o nome de **formato**.

Normalmente, um formato define um subconjunto de um registo da base de dados, o qual pode ser utilizado pelo CDS/ISIS, embora um formato seja à partida, criado para definir a forma como os registos são apresentados (formatados) no écran ou na impressora, são também usados com frequência em várias fases da utilização do programa sempre que é necessário trabalhar com informação contida nos registos da base de dados.

Por exemplo, nas tabelas de Selecção de Campos (FST), é utilizado um formato para definir quais os dados a aplicar a uma determinada técnica de indexação. A linguagem de formatação é, por esse motivo, o princípio (centro nervoso) de muitas das operações do CDS/ISIS, e para um eficiente uso desta linguagem é necessário um profundo conhecimento desta técnica.

Para um principiante, alguns formatos podem parecer muito complexos, dando a entender que a linguagem de formatação em si é muito complicada. De facto, todos os formatos, mesmo os mais complexos, são criados a partir de um ou mais comandos ou instruções, separados por vírgulas ou espaços. A aparente complexidade, advém do facto, de poderem existir muitos comandos no mesmo formato. Assim, a maneira mais fácil de compreender um formato é analisar cada comando individualmente.

Se bem que todos os formatos são definidos recorrendo à mesma linguagem, eles podem ser agrupados por categorias de acordo com o uso a que se destinam:

- **Formatos de visualização:** utilizados para apresentar no écran (visualizar) ou para imprimir registos (neste último caso denominados como formatos de impressão);

- **Formatos de extracção:** utilizados nas FST's para definir os dados a extrair ou a indexar.

Quando o CDS/ISIS executa um formato, a sua análise é feita tendo em conta três elementos: o registo da base de dados, o formato e uma área de trabalho onde o resultado produzido pelo formato é armazenado. Os comandos são executados sequencialmente pela mesma ordem como se apresentam no formato. Alguns comandos geram dados (isto é, o conteúdo de um registo) enquanto que outros produzem determinadas acções (como saltar de linha, criar uma linha em branco, etc.). Os dados produzidos são armazenados como linhas de

texto na área de trabalho, sendo então tratados para posterior processamento, como por exemplo, a impressão.

Quando um formato é utilizado para visualização, as linhas de texto produzidas são normalmente restritas a uma determinada dimensão máxima (dimensão da linha). Por exemplo, quando os dados são visualizados em écran, o CDS/ISIS limita automaticamente a dimensão da linha a 80 caracteres, a não ser que novas linhas sejam criadas condicionalmente pelos comandos para o efeito. O CDS/ISIS preenche cada linha o mais possível, até atingir a dimensão máxima da mesma. Quando um campo ultrapassa a dimensão máxima da linha, é repartido em várias linhas; no entanto, sempre que for necessário "partir" uma linha, será sempre entre palavras (isto é, uma palavra nunca será repartida entre duas linhas). Todos os comandos de formatação podem ser digitados em caracteres maiúsculos, minúsculos, ou pela combinação de ambos.

CAPÍTULO 3 : ESTRUTURA DE FICHEIROS DAS BASES DE DADOS CDS/ISIS

Este capítulo descreve a estrutura dos vários ficheiros que compõem uma base de dados CDS/ISIS. Para o utilizador comum esta informação não é relevante, mas pode ser importante para conhecer o comportamento e potencialidades deste tipo de bases de dados.

A ESTRUTURA DOS FICHEIROS

Um registo com uma estrutura ISIS possui duas características especiais que possibilitam com enorme versatilidade a manipulação de informação textual: campos repetíveis e campos de comprimento variável.

Uma vez que os registos não têm uma dimensão predeterminada, assim como os campos não têm um comprimento fixo nem um número predeterminado de repetições, não é possível ter acesso directo a nenhuma parte dos dados de uma base.

O acesso ao registo é feito de um modo indirecto, através de apontadores de um ficheiro auxiliar, com a extensão .XRF, e dentro do registo o acesso é feito através de apontadores existentes num directório. O ficheiro .XRF contém toda a informação necessária para localizar o início de um registo localizado no ficheiro .MST.

A tabela apresenta o registo MFN=1 da base de dados CDS, cujo conteúdo é o seguinte:

Registo de controlo

```
Nxtmfn nxtmfb nxtmfp  t  reccnt mfcxx1 mfcxx2 mfcxx3 RC
152      123      13       0        0        0        0        0        0
```

xref

```
Mfn=     1|comb=  1|comp= 64|      N| ... | 1+000=00000c40
```

Etiqueta de registo

```
Mfn=     1|mfrl=  370|mfbwb=       0|mfbwp= 0|base=  66|nvf=  8|status= 0| 0
```

Directório

```
Mfn=     1|dir=   1|tag=   44|pos=  0|len=   77
Mfn=     1|dir=   2|tag=   50|pos=  77|len=  11
Mfn=     1|dir=   3|tag=   69|pos=  88|len=  78
Mfn=     1|dir=   4|tag=   24|pos=  166|len= 68
Mfn=     1|dir=   5|tag=   26|pos=  234|len= 22
Mfn=     1|dir=   6|tag=   30|pos=  256|len= 20
Mfn=     1|dir=   7|tag=   70|pos=  276|len= 15
Mfn=     1|dir=   8|tag=   70|pos=  291|len= 12
```

Campos de dados

```
Mfn=     1
44  «Methodology of plant eco-physiology: proceedings of the Montpellier
Symposium»
50  «Incl. bibl.»
69  «Paper on: <plant physiology><plant transpiration><measurement and
instruments>»
24  «Techniques for the measurement of transpiration of individual plants»
26  «^aParis^bUnesco^c-1965»
30  «^ap. 211-224^billus.»
70  «Magalhaes, A.C.»
70  «Franco, C.M.»
..
```

FORMATO DO REGISTO DO FICHEIRO MESTRE

O registo do ficheiro mestre é de comprimento variável composto por três segmentos: um cabeçalho de comprimento fixo, um directório e os campos de dados de comprimento variável.

Formato do cabeçalho

O cabeçalho é constituído por 7 números inteiros que se seguem (os campos assinalados com um * são inteiros de 31 bits):

MFN*	Número do registo no ficheiro mestre
MFRL	Dimensão do registo (sempre um número ímpar)
MFBWB*	Apontador anterior - Número do bloco
MFBWP	Apontador anterior - Posição
BASE	Posição do primeiro campo de dados (dimensão conjunta do cabeçalho e do directório do registo, expresso em bytes).
NVF	Número de campos do registo (isto é, número de entradas no directório)
STATUS	Indicador lógico de registo apagado (0= registo activo; 1=registo marcado para eliminação).

Os apontadores **MFBWB** e **MFBWP** são inicialmente definidos com o valor 0 quando o registo é criado. São posteriormente actualizados, de cada vez que um registo é também modificado (ver a seguir).

Formato do directório

O directório é uma tabela que indica o conteúdo dos registos. Existe apenas um directório para cada campo presente no registo (isto é, o directório tem exactamente **NVF** entradas). Cada entrada do directório contém 3 elementos inteiros:

TAG	Etiqueta do campo
POS	Posição do primeiro caracter do campo no segmento do campo de dados (o primeiro campo tem POS=0)
LEN	Dimensão do campo em bytes

A dimensão total em bytes do directório é igual a **6*NVF**; o campo **BASE** do cabeçalho é sempre **18+6*NVF**.

Campos de comprimento variável

Este segmento contém os campos de dados (pela ordem indicada no directório). Os campos de dados são dispostos um após outro, sem qualquer caracter de separação.

Registo de controlo

O primeiro registo do ficheiro mestre é um registo de controlo que o sistema actualiza automaticamente. Este nunca pode ser acedido pelo utilizador. Contém os seguintes campos (os campos assinalados por um * são inteiros de 31 bits):

CTLMFN* é sempre 0

NXTMFN* o MFN a ser atribuído ao próximo registo criado na base de dados

NXTMFB* O último número do bloco ocupado pelo ficheiro mestre (o primeiro bloco é 1)

NXTMFP A localização da próxima posição disponível no último bloco

MFTYPE É sempre 0 para a base de dados de utilizador (1 para a base de dados das mensagens)

RECCNT*

MFCXX1*

MFCXX2*

MFCXX3*

(os últimos quatro campos são utilizados para elaboração de estatísticas durante as operações de salvaguarda e recuperação da base de dados).

Formato do bloco do ficheiro mestre

Os registos do ficheiro mestre são armazenados sequencialmente, um após outro, ocupando cada um **MFRL** bytes. O ficheiro é armazenado em blocos físicos de 512 bytes. Um registo pode iniciar num valor compreendido entre 0 e 498 (nenhum registo começa entre 500 e 510) e pode prolongar-se por dois ou mais blocos.

Como o ficheiro mestre é criado e/ou modificado, o sistema cria um índice indicando a posição de cada registo. Esta informação é armazenada no ficheiro índice de referência .XRF.

Ficheiro índice de referência

O ficheiro .XRF está organizado como sendo uma tabela de apontadores do ficheiro mestre. O primeiro apontador corresponde ao **MFN 1**, o segundo ao **MFN 2**, etc.

Cada apontador é composto por dois campos:

XRFMFB (21 bits) Número do bloco do ficheiro mestre que contém o registo

XRFMFP (11 bits) Posição no bloco do primeiro caracter do ficheiro mestre (o primeiro bloco é a posição 0)

que são armazenados numa variável inteira de 31 bits (4 bytes) como se indica:

apontador = XRFMFB*2048 + XRFMFP

Cada bloco do ficheiro .XRF tem 512 bytes e contém 127 apontadores. O primeiro campo de cada bloco (**XRFPOS**) é uma variável inteira de 31 bits, em que o seu valor absoluto é o número do bloco do XRF. Um valor negativo de **XRFPOS** indica o último bloco.

Os registos apagados são indicados da seguinte forma:

XRFMFB<0	0 registo apagado logicamente (neste caso, ABS(XRFMFB) é o
XRFMFP>	apontador correcto do bloco e XRFMFP é a posição do registo, que pode ainda ser recuperado)
	1
XRFMFB=-1	registo apagado fisicamente
XRFMFP=0	
XRFMFB=0	registo inexistente (todos os registos posteriores ao maior MFN atribuído
XRFMFP=0	na base de dados)

Técnicas de actualização do ficheiro mestre

Criar novos registos

Os novos registos são adicionados no fim do ficheiro mestre, na posição indicada pelos campos **NXTMFP/NXTMFP** do registo de controlo do ficheiro mestre. O MFN a ser atribuído é também obtido a partir do campo **NXTMFN** do registo de controlo.

Após a adição do registo, o campo **NXTMFN** é incrementado de 1 e os campos **NXTMFB/NXTMFP** são actualizados de forma a apontar para a próxima posição disponível. Por outro lado, é criado um novo apontador no ficheiro .XRF e o campo **XRFMFP** correspondente ao registo é incrementado de 1024 para indicar que este é um novo registo a ser invertido (após a inversão do registo, o valor 1024 é subtraído a **XRFMFP**).

Actualizar registos

Sempre que se actualiza um registo (isto é, o registo é editado através da folha de recolha de dados e abandona-se o editor com X) o sistema escreve o registo de novo no ficheiro mestre. A ordem de armazenamento do registo depende do seu estado inicial no momento em que foi lido.

Não há registos pendentes para actualizar o ficheiro inverso.

Esta condição é indicada pelo seguinte:

No XRF XRFMFP<512 e
No MST MFBWB=0 e MFBWP=0

Neste caso, o registo é sempre rescrito no fim do ficheiro mestre (como se de um novo registo se tratasse) como indicam os campos **NXTMFB/NXTMFP** do registo de controlo. Na nova versão do registo, os campos **MFBWB/MFB**WP passam a apontar para a versão antiga do registo, enquanto que no ficheiro .XRF, o apontador aponta para a nova versão. Por outro lado, o valor 512 é adicionado ao campo **XRFMFP** para indicar que se trata de um registo pendente para actualizar no ficheiro inverso. Quando o ficheiro inverso é actualizado, a versão antiga do registo é utilizada para determinar os apontadores a serem apagados, e a nova versão para adicionar os novos apontadores. Após a actualização do ficheiro inverso, o valor 512 é subtraído ao campo **XRFMFP**, e os campos **MFBWB/MFBWP** passam a 0.

Há registos pendentes para actualizar o ficheiro inverso.

Esta condição é indicada pelo seguinte:

No XRF XRFMFP>512 e
No MST MFBWB>0

Neste caso, os campos **NXTMFB/NXTMFP** apontam para a versão do registo que consta do ficheiro inverso. Se possível, se a dimensão do registo não aumentar, o registo é escrito de novo na posição original, caso contrário será escrito no fim do ficheiro mestre. Em ambos os casos, os campos **MFBWB/MFBWP** não são alterados.

Apagar registos

A eliminação de registos é feita como se de uma actualização se tratasse, como a seguir se indica:

No XRF XRFMFB é negativo
No MST STATUS passa a 1

Reorganização do ficheiro mestre

Como anteriormente mencionado, a actualização do ficheiro mestre faz aumentar a sua dimensão e o espaço ocupado, pelas versões anteriores do registo, não pode ser reutilizado. A reorganização permite recuperar este espaço perdido, compactando o ficheiro mestre.

Durante a fase de segurança é criada uma salvaguarda do ficheiro mestre (.BKP). A estrutura e o formato deste ficheiro é a mesma do ficheiro mestre (.MST), à excepção do ficheiro índice de referência (.XRF) que não é necessário, uma vez que todos os registos são adjacentes). Os registos marcados para serem apagados não são salvaguardados. Como é a última versão do registo que é salvaguardada, o sistema só permite esta operação depois da actualização total do ficheiro inverso.

Durante a fase de recuperação da salvaguarda, o ficheiro é lido sequencialmente sendo recriados os ficheiros .MST e .XRF. Nesta fase, todos os registos assinalados como apagados logicamente (antes da salvaguarda) passam a estar apagados fisicamente (através da alteração dos campos **XRFMFB=-1** e **XRFMFP=0**). Os registos apagados são detectados verificando as falhas na numeração dos MFN.

O ficheiro inverso do CDS/ISIS é constituído por 6 ficheiros lógicos distintos, 5 dos quais contêm os termos pesquisáveis do léxico (organizados em índice tipo árvore - B*tree) e o sexto contém a lista dos apontadores associados a cada termo. Com o objectivo de optimizar o espaço em disco, existem dois B*tree separados, um para os termos até 10 caracteres (armazenados nos ficheiros .N01/.L01) e outro com os termos superiores a 10 caracteres(1) (armazenados nos ficheiros .N02/.L02). O ficheiro .CNT contém os campos de controlo para ambas as B*tree. Em cada B*tree, o ficheiro .N0x contém os NfS da árvore e o ficheiro .L0x contém as FOLHAS. Os registos das folhas remetem para o ficheiro de apontadores .IFP.

A relação entre os vários ficheiros é representada esquematicamente na figura seguinte.

As relações físicas entre estes seis ficheiros, é um apontador, que representa o endereço relativo do registo a remeter. O endereço relativo é o número de registo ordinal de um registo de um ficheiro (isto é, o primeiro registo é o número 1, o segundo é o registo número 2, etc.) O ficheiro .CNT aponta para o ficheiro .N0x, o .N0x aponta para o .L0x, e o .L0x aponta para o .IFP. Como o .IFP é um ficheiro compactado, o apontador de .L0x para .IFP tem dois componentes: o número do bloco e a posição dentro do bloco, cada um expresso por um número inteiro.

FICHEIRO INVERSO

Formato do ficheiro .CNT

Este ficheiro contém dois registos de comprimento fixo de 26 bytes (um para cada B*tree), contendo cada um 10 campos inteiros como se indica (os campos marcados com um * são inteiros de 31 bits):

IDTYPE	Tipo da B*tree (1 para .N01 / .L01, 2 para .N02 / .L02)
ORDN	Ordem dos nós (cada registo do .N0x contém no máximo 2*ORDN chaves)
ORDF	Ordem das folhas (cada registo do .L0x contém no máximo 2*ORDF chaves)
N	Número de zonas de memória definidas para os nós
K	Número de zonas de memória definidas para o primeiro nível do índice (K<N)
LIV	Número actual de níveis do índice
POSRX*	Apontador para o registo raiz do ficheiro .N0x

(1) Os termos de pesquisa não podem ultrapassar os 30 caracteres; se o utilizador definir termos superiores, estes serão truncados a 30 caracteres.

NMAXPOS*	Próxima posição disponível no ficheiro .N0x
FMAXPOS*	Próxima posição disponível no ficheiro .L0x
ABNORMAL	Indicador formal de normalidade da B*tree (0 se a B*tree não é normal, 1 se a B*tree é normal). Uma B*tree não é normal se o ficheiro de nós .N0x tiver apenas a raiz.

ORDN, ORDF, N, e K são fixos para um determinado sistema. Em geral, estes valores são os seguintes:

ORDN=5; ORDF=5; N=15; K=5 para ambos os B*trees

Os outros valores são definidos durante a criação das B*trees.

Formato dos ficheiros .N0x

Estes ficheiros contêm o índice dos termos de pesquisa do dicionário (.N01 para os termos até 10 caracteres, e .N02 para os termos superiores a 10 caracteres). Os registos do ficheiro .N0x têm o seguinte formato (os elementos assinalados com um * são inteiros de 31 bits)

POS* um inteiro que indica o número relativo do registo (1 para o primeiro registo, 2 para o segundo registo, etc.)

OCK um inteiro que indica o número das chaves activas no registo (1<=OCK<=2*ORDN)

IT um inteiro que indica o tipo da B*tree (1 para .N01, 2 para .N02)

IDX uma tabela de entradas ORDN (OCK das chaves activas), cada uma com o seguinte formato:

KEY uma cadeia de caracteres de comprimento fixo de LEx caracteres (LE1=10, LE2=30)

PUNT um apontador para o registo de .N0x (se PUNT>0) ou para o registo de .L0x (se PUNT<0) no qual IDX[1].KEY=KEY. PUNT=0 indica uma subdivisão para um índice de nível hierárquico inferior. O índice de nível mais baixo (PUNT<0) aponta para a folha no ficheiro .L0x.

Formato dos ficheiros .L0x

Estes ficheiros contêm todo o índice dos termos de pesquisa (.L01 para os termos até 10 caracteres, e .L02 para os termos superiores a 10 caracteres). Os registos do ficheiro .L0x têm o seguinte formato (os campos marcados com um * são inteiros de 31 bits).

POS* um inteiro que indica o número relativo do registo (1 para o primeiro registo, 2 para o segundo registo, etc.)

OCK um inteiro que indica o número das chaves activas no registo (1<=OCK<=2*ORDF)

IT um inteiro que indica tipo do B*tree (1 para .L01, 2 para .L02)

PS* aponta para o registo seguinte do .L0x (isto é, o registo no qual IDX[1].KEY é o registo imediatamente a seguir a IDX[OCK].KEY deste registo (isto é utilizado para aumentar a velocidade de acesso sequencial ao ficheiro)

IDX uma tabela de entradas ORDN (OCK das chaves activas), cada uma com o seguinte formato:

KEY uma cadeia de caracteres de comprimento fixo de LEx caracteres (LE1=10, LE2=30)

INFO um apontador para o registo de .IFP, onde começa a lista de apontadores associados com KEY. O apontador é composto por dois inteiros de 31 bits, como se indica:

INFO[1]* número relativo do bloco no .IFP

INFO[2]* posição (número da palavra relativo a 0) para a lista de apontadores

Formato do ficheiro .IFP

Este ficheiro contém a lista de apontadores para cada termo do dicionário. Cada lista de apontadores tem o formato indicado de seguida. O ficheiro é estruturado em blocos de 512 caracteres, onde (para um ficheiro carregado inicialmente e compactado) as listas de apontadores para cada termo são adjacentes, salvo para as indicações a seguir mencionadas.

O formato genérico para cada bloco é:

IFPBLK um inteiro de 31 bits, indicando o número do bloco (os blocos são numerados a partir de 1)

IFPREC uma tabela de 127 entradas de 31 bits

IFPREC[1] e **FPREC[2]** do primeiro bloco são um apontador para a próxima posição disponível no ficheiro .IFP.

Os apontadores de .L0x para .IFP e os apontadores no .IFP são dois inteiros de 31 bits; o primeiro inteiro é um número de bloco, e o segundo é uma posição da palavra no **IFPREC** (ou seja, a posição da primeira palavra no **IFPREC** é 0). A lista de apontadores associados ao primeiro termo de pesquisa começa em 1/0.

IFPNXTB* Apontador do próximo segmento (número do bloco)

IFPNXTP* Apontador do próximo segmento (posição)

IFPTOTP* Número total de apontadores (apenas no primeiro segmento)

IFPSEGP* Número de apontadores neste segmento (IFPSEGP<=IFPTOTP)

IFPSEGC* Capacidade do segmento (isto é, o número de apontadores que podem ser armazenados neste segmento)

Cada apontador é uma cadeia de caracteres de 64 bits, composta por:

PMFN (24 bits) Número de registo do ficheiro mestre

PTAG (16 bits) Identificador de campo (atribuído pela FST)

POCC (8 bits) Número de ocorrência

PCNT (16 bits) Número de sequência do termo no campo

Cada campo é armazenado da esquerda para a direita, e se necessário, acrescido de zeros para alinhar à direita a cadeia de caracteres (isto permite comparar dois apontadores como duas cadeias de caracteres com a mesma dimensão).

A lista dos apontadores é armazenada pela ordem ascendente da sequência **PMFN/PTAG/POCC/PCNT**. Quando o ficheiro inverso é carregado sequencialmente (após a inversão total com o serviço ISISINV), cada lista consiste num ou mais segmentos adjacentes. Se **IFPTOT<=32768** então: **IFPNXTB/IFPNXTP=0/0** e **IFPTOT=IFPSEGP=IFPSEGC**.

Sempre que é feita uma actualização, podem ser criados segmentos adicionais sempre que novos apontadores tenham que ser adicionados. Neste caso, é criado um novo segmento com a capacidade de **IFPTOTP** e ligado a outros segmentos (através do apontador **IFPNXTB/IFPNXTP**) de forma a manter a sequência **PMFN/PTAG/POCC/PCNT**. Sempre que ocorre uma quebra, os apontadores do segmento onde o novo apontador deveria ser inserido, são distribuídos de igual forma entre este segmento e o criado de novo. Os novos segmentos são sempre escritos no fim do ficheiro (que é controlado por **IFPREC[1]/ IFPREC[2]** do primeiro bloco do .IFP.

Por exemplo, se o novo apontador Px for inserido entre P2 e P3, como apresenta a lista seguinte:

```
+--------------------------------------+
¦  0  0  5  5  5  |  P1  P2  P3  P4  P5  ¦
+--------------------------------------+
```

após a quebra (e assumindo que a próxima posição disponível no ficheiro .IFP é 3/4) a lista de apontadores passará a ser composta pelos dois segmentos seguintes:

```
+--------------------------------------+
¦  3  4  5  3  5  |  P1  P2  Px  --  --  ¦
+---¦----------------------------------+
+---V----------------------------------+
¦  0  0  5  3  5  |  P1  P2  P5  --  --  ¦
+--------------------------------------+
```

Neste caso, não será criado nenhum segmento novo, até que ambos os segmentos atinjam os limites.

Como anteriormente mencionado, as listas de apontadores são normalmente armazenadas umas após outras. No entanto, para facilitar o acesso ao ficheiro .IFP os segmentos são armazenados da seguinte forma:

- cabeçalho e o primeiro apontador em cada lista (28 bytes) nunca são repartidos entre dois blocos.

- um apontador nunca é repartido entre dois blocos; se não existir espaço suficiente no bloco corrente, todo o apontador será armazenado no bloco seguinte.

CAPÍTULO 4 : ESPECIFICAÇÕES DA LINGUAGEM

Neste capítulo encontra a explicação de todos os comandos da linguagem de formatação. Sempre que possível, é apresentado um pequeno exemplo que ilustra a sua aplicação.

Registo de exemplo

Salvo indicações em contrário, todos os exemplos relativos aos formatos que se seguem, referem-se à figura seguinte, cujo conteúdo de cada campo é o que consta no registo. Este registo, é tirado a partir da base de dados de exemplo CDS distribuída pela UNESCO.

Exemplo

```
MFN = 4
Etq  Conteúdo
---- ------------------------------------------------------------
24     <An> Electric hygrometer apparatus for measuring water-vapour loss from plants in the
       field
26     ^aParis^bUnesco^c1985
30     *^ap. 247-257^billus.
44     Methodology of plant eco-physiology: proceedings of the Montpellier Symposium
50     Incl. bibl.
69     Paper on: <hygrometers><plants transpiration><moisture><water balance>
70     Grieve, B.J.
70     Went, F.W.
```

Os formatos podem ser criados e testados de várias formas e dependem da versão onde são utilizados. Podem ser criados através de um editor de texto (EDIT ou NOTEPAD do Windows). Na versão para DOS devem ser testados através da opção “Seleccionar formato” e “Percorrer ficheiros”. O mesmo acontece na versão para Windows. Em qualquer dos casos podem criar ou modificar formatos e testá-los de imediato, fazendo as correcções em função dos erros e/ou resultados obtidos. No BIBLIObase, pode utilizar uma opção criada especificamente para o efeito, que lhe permite ver, em tempo real, o resultado que se pode obter.

Selector de campo (comando V)

O selector de campo, é um comando usado para extrair um campo específico ou um subcampo de um registo. Um comando especial permite extrair o MFN (número de registo da base de dados) do registo, mesmo que este não seja um campo (o MFN não tem etiqueta e não é definido na FDT).

O conteúdo pode ser seleccionado, limitado, extraído ou alinhado utilizando outros componentes indicados a seguir.

Sintaxe

V<etiqueta do campo>*[outros parâmetros]*

Em que:

<etiqueta do campo> - qualquer valor entre 1 e 32675.

<outros parâmetros> - **^**<*identificador de subcampo*> **[**<*ocorrência*>**[..**<*limite superior*>**]]** *****<*offset*>**.**<*dimensão*> **(**<*primeira linha*>,<*linhas seguintes*>**)**

Identificador de subcampo
Limita o resultado ao conteúdo de um subcampo.

Ocorrência
Limita a saída a uma ocorrência ou a um conjunto de um campo repetível. O parâmetro <*ocorrência*> e <*limite superior*> referem-se à primeira (ou única) e última ocorrência do campo respectivamente. Se o valor indicado em <*ocorrência*> for maior que o número total de ocorrências, não é apresentado nenhum resultado. O mesmo acontece se o campo não for repetível e o valor <*ocorrência*> tiver um valor maior ou igual a 2. Contudo, se <*ocorrência*> for igual a 1 e for usado como num campo não repetível, o seu conteúdo é apresentado.

Este parâmetro deve ser utilizado fora de um grupo repetível; caso contrário <limite superior> é ignorado. Se forem usados dois pontos "**..**" e for omitido o parâmetro <*limite superior*> é assumido **LAST**. Esta chave contém o valor total de ocorrências do campo.

Extracção
Extrai o conteúdo parcial de um campo de dados, subcampo ou ocorrência. O parâmetro <*offset*> é a primeira posição a ser extraída, enquanto que a <*dimensão*> determina o número de caracteres a ser extraídos. Se <*dimensão*> não for indicado ou se for maior que a dimensão do campo, o valor assumido por defeito é a dimensão do campo.

Alinhamento
Alinha o resultado obtido do campo, subcampo, ocorrência ou partes do campo, de acordo com o valor indicado em <*primeira linha*> e <*linhas seguintes*>. Ambos os valores são constantes numéricas. Se a posição na linha corrente for diferente de zero, o alinhamento fica inactivo.

O comportamento deste comando depende dos parâmetros usados. Nada é apresentado quando o campo não existir ou quando um dos componentes ultrapassar uma das restrições impostas.

Exemplo genérico

```
v2/,v3^a| - |,v1/,
v1^n*0.3,
(|; |+v3^s)/,
v20[4],
v10[2..7]/,
v5[3..]/,/* equals to ,v5[3..LAST], */
v1[LAST]*2.7/,
v1(5,5)/,
|Title: |v1^n(5,5)/,
```

Para extrair o conteúdo de um campo de um registo, deve indicar-se a letra ***V*** seguida da etiqueta do campo.

A letra ***V*** (uma mnemónica para definir *Campo de dimensão variável*) é um comando que permite ao CDS/ISIS extrair o conteúdo de um determinado campo. Este código pode ser digitado em maiúsculas ou minúsculas. A figura seguinte apresenta alguns exemplos aplicados ao registo anterior.

Exemplo

```
V24         <An> Electric hygrometer apparatus for measuring water-vapour loss from plants in
            the field
V26         ^aParis^bUnesco^c1985
V30         ^ap. 247-257^billus
V44         Methodology of plant eco-physiology: proceedings of the Montpellier Symposium
```

Comando de subcampo

Para extrair um subcampo a partir de um determinado campo, basta apenas acrescentar o delimitador de subcampo correspondente à etiqueta, tal como é apresentado na figura seguinte. Pode utilizar-se o delimitador de subcampo especial ^ para seleccionar o primeiro subcampo. Neste caso o primeiro subcampo não necessita de ser precedido por um delimitador na fase de digitação.

É indiferente a utilização dos delimitadores de subcampo alfabéticos, maiúsculos ou minúsculos.

Exemplo

```
V26^b       UNESCO
V26^a       p. 247-257
V44^*       Methodology of plant eco-physiology: proceedings of the Montpellier Symposium
v26^*       Paris
```

Extracção de excertos de campos ou subcampos

Pode ser necessário, nalguns casos, extrair uma parte de um campo que não é um subcampo, particularmente quando o campo tem um formato fixo de recolha na folha de recolha de dados, como por exemplo, a inscrição de uma data (AA-MM-DD).

Isto pode ser feito, acrescentando o comando posição / dimensão ("*offset/lenght*") imediatamente a seguir ao comando de campo ou subcampo a que se aplica. Este comando pode ter a seguinte representação: ****posição.dimensão*** ou ****posição*** ou ***.dimensão***, em que:

****posição*** indica a posição do primeiro caracter a ser extraído de um campo ou subcampo (o número de posições é contado a partir do valor 0, isto é, o primeiro caracter, representa a posição 0, o segundo a posição 1, etc.). Se for omitido, o valor da posição é assumido como sendo igual a 0;

.dimensão indica o número de caracteres a serem extraídos; se for omitido a informação restante do campo ou subcampo começa a partir do valor da posição.

São apresentados alguns exemplos na figura seguinte, onde é assumido um registo de exemplo contendo um campo identificado com a etiqueta 1, cujo conteúdo é o seguinte:

```
88-Nov-05
```

Os dois últimos exemplos apresentam uma diferença no tratamento do subcampo: quando se indica um campo (por exemplo V26) a posição zero corresponde à primeira posição do campo, e quando se indica um subcampo (por exemplo V26^b), a posição zero representa o primeiro caracter dos dados logo após o delimitador de subcampo.

Exemplo

```
V1*3.3             Nov
V1.2               88
V1*7               05
V1*7,V1*2.4        05-Nov
V1*7,V1*2.5,V1.2   05-Nov-88
V26.3              ^aP
V26^b*2.4          esco
```

Comando de Indentação

Quando o CDS / ISIS executa um comando de campo ou de subcampo, apresenta o resultado, normalmente, a partir da posição da linha em que se encontra, o que depende do último comando executado.

Se um campo não puder ser apresentado numa única linha, este estender-se-á por tantas linhas, quantas as necessárias, até à sua completa visualização. Normalmente estas linhas começam na posição 1.

Esta posição pode ser alterada através do uso de um comando de indentação (alinhamento), que deve ser acrescentado a seguir a um comando de campo ou subcampo. O comando de indentação pode ser codificado da seguinte forma:

(f,c) ou **(f)**

em que:

f indica o número de espaços a partir da margem esquerda antes de representar a primeira linha do campo. Este comando só actua, de facto, se o campo for formatado a partir do início da linha, doutro modo será ignorado.

c indica o número de espaços a partir da margem esquerda, antes de apresentar todas as linhas restantes de um campo.

O valor 0 pode ser indicado, tanto para f como para c. Se apenas for necessário o **f**, o **c** pode ser omitido (é assumido então, por defeito, valor 0). No entanto, se for necessário o **c**, também será necessário especificar o **f**. São apresentados alguns exemplos na figura seguinte.

Exemplo

```
V44         Methodology of plant eco-physiology: proceedings of the Montpellier Symposium
V44(10)               Methodology of plant eco-physiology: proceedings of the Montpellier
            Symposium
V44(5,9)          Methodology of plant eco-physiology: proceedings
                        of the Montpellier Symposium
V44(0,8)    Methodology of plant eco-physiology: proceedings of
                       the Montpellier Symposium
```

Este comando, quando associado à função **S** (ver "*S(formato)*"), serve para indentar um conjunto de campos, evitando assim a indicação do comando de indentação em cada campo, o que em muitos casos se torna bastante útil, quando a dimensão do formato já é demasiada.

Por exemplo:

S(V10,V20,V30)(3,3)

faz com que o texto gerado pelos campos **10, 20** e **30** seja indentado 3 posições a partir da margem esquerda. No ISIDLL e WWWISIS esta função permite ainda incluir os parâmetros "*****" e "**.**" que permitem limitar a dimensão da cadeia de caracteres resultante.

Comando MFN

Extrai o MFN de um registo da base de dados.

Sintaxe

MFN
MFN*(d)*
Em que:

***d** – indica o número de dígitos apresentados*

Deve indicar-se no formato da seguinte forma:

MFN ou **MFN(d)**

em que **d** é o número de dígitos a serem apresentados. Se **(d)** for omitido, então serão apresentados 6 dígitos por defeito. Ver a figura seguinte:

Exemplo

```
MFN         000004
MFN(3)      004
MFN(2)      04
MFN(1)      4
```

Pode ser utilizada a função **F** (ver "*F(expr-1,expr-2,expr-3)*") para suprimir os zeros não significativos.

COMANDOS DE MODO (M)

O CDS/ISIS permite visualizar os dados de três modos diferentes:

Modo de prova neste modo os campos serão visualizados tal como foram armazenados num registo. O CDS/ISIS não insere qualquer separador entre os campos ou ocorrências de campo repetível. É da responsabilidade do utilizador, recorrer ao uso de comandos de espaçamento, literais ou grupos repetíveis para apresentar da forma mais adequada a separação dos campos. Este modo é normalmente utilizado quando se desejam visualizar registos para efeitos de correcção.

Modo de cabeçalho este modo é normalmente usado para a criação de cabeçalhos quando se imprimem listagens com referências bibliográficas e índices. Todos os caracteres de controlo incluídos no texto, como as directivas de ordenação (ver "Directivas de ordenação") e os delimitadores de palavras-chave (< e >) serão ignorados (exceptuam-se os assinalados de seguida), ao mesmo tempo que os delimitadores de subcampos são substituídos por sinais de pontuação (ver a seguir).

Modo de dados este modo é idêntico ao modo de cabeçalho, mas adiciona no fim de cada campo um ponto "." seguido de 2 espaços (ou apenas 2 espaços se o campo já terminar com o ponto). No entanto, a pontuação automática gerada será suprimida se o selector de campo for seguido por uma literal como sufixo. (ver "Literais").

Quando o CDS/ISIS formata o subcampo de um campo em modo de cabeçalho ou modo de dados, substitui automaticamente os delimitadores de subcampo por sinais de pontuação (o primeiro delimitador de subcampo é sempre ignorado, caso exista). A combinação de caracteres especiais < > é substituída por "**;**", permitindo assim uma forma simples de apresentação das frases contendo as palavras-chave limitadas pelos caracteres especiais < e >.

Os delimitadores de subcampo são substituídos da seguinte forma:

^a	substituído por ';'
^b até ^i	substituído por ','
outros	substituído por '.'

Um comando de modo representa-se da seguinte forma:

M*mc* em que:

m indica o modo como se indica:

P modo de prova
H modo de cabeçalho
D modo de dados

c indica a conversão de caracteres como se indica:

U conversão em maiúsculas
L mantêm-se inalterados

Num formato pode utilizar-se um comando de modo, tantas vezes quantas as necessárias, mantendo-se activo até ser alterado por um outro. Na ausência de um comando de modo no formato, é assumido por defeito MPL (modo de prova sem conversão para maiúsculas).

Exemplos deste comando são apresentados na figura seguinte.

Exemplo

```
mpl,V24    <An> Electric hygrometer apparatus for measuring water-vapour loss from plants in
           the field
mhl,V24    An Electric hygrometer apparatus for measuring water-vapour loss  from plants in the
           field
mdl,V24    An Electric hygrometer apparatus for measuring water-vapour loss from plants in the
           field.
mdu,V24    AN ELECTRIC HYGROMETER APPARATUS FOR MEASURING WATER-VAPOUR OSS FROM PLANTS IN THE
           FIELD
mpl,V26    ^aParis^bUnesco^c1985
mhl,V26    Paris, UNESCO, 1985
mdu,V26    PARIS, UNESCO, 1985.
mpl,V69    Paper on: <hygrometers><plants transpiration><moisture><water balance>
mdl,V69    Paper on: hygrometers; plants transpiration; moisture; water balance.
```

COMANDOS DE ESPAÇAMENTO VERTICAL E HORIZONTAL

A linguagem de formatação permite sete comandos para controlo do espaçamento horizontal e vertical (dois deles aplicam-se apenas na impressão). Resumem-se na figura que se segue.

O comando ***Xn*** insere n espaços em branco antes de formatar o conteúdo do próximo campo ou subcampo. No entanto, se existirem menos espaços livres do que os requeridos na linha, o CDS/ISIS salta para a linha seguinte. Por exemplo, se a próxima posição livre numa linha for 77 e a dimensão máxima da linha for 80, o comando ***X7*** apresentará os dados no início da linha seguinte (e não na terceira posição da linha seguinte).

Xn	Insere n espaços antes de escrever o campo seguinte.
Cn	Posiciona o início do texto na coluna n.
/	salta para a linha seguinte se não existir uma linha em branco na anterior.
#	insere uma nova linha em branco (incondicionalmente)
%	apaga anterior(es) linha(s) em branco caso exista(m).
[1]NP(n)	salta para a página seguinte (condicional ou incondicionalmente)
NC(n)	salta para a coluna seguinte (condicional ou incondicionalmente)

O comando Cn define o texto a ser apresentado a partir da posição n da linha. Se a posição actual for maior do que n, então o texto será formatado a partir da posição n da linha seguinte. Esta facilidade permite definir a tabulação do texto. Se o valor de n for maior do que a dimensão máxima da linha, o valor n será ignorado.

O comando / é idêntico a ser premido um <CR> (mudança de linha) num processador de texto, isto é, obriga à criação de uma nova linha como também, a formatação a continuar no início da linha seguinte. Contudo, ao contrário do <CR>, uma sequência de várias / adjacentes (do ponto de vista sintáctico está correcto) produz o mesmo efeito do que um só comando: este não gera linhas em branco, ou seja, cada / não obriga à criação do mesmo número de linhas. O comando # é o indicado para resolver esta situação. Inicialmente executa o mesmo que o comando **/**, mas o salto para a nova linha é incondicional.

A combinação dos comandos **/#** obrigam a que seja gerada uma linha em branco (e só uma) na formatação de um texto (a combinação de **##** pode provocar a criação de uma ou duas linhas em branco dependendo da posição em que a linha começou a ser formatada quando o primeiro **#** é executado, ou seja, se a linha inicial está em branco ou não).

O uso do comando **#** pode trazer problemas nos casos em que o campo seleccionado não existe. Esta situação é ilustrada com o exemplo que se segue:

/#V10/#V20/#V30 ...

Se todos os campos do registo acima indicados existirem, terá como resultado o espaçamento de uma linha em branco entre cada campo. No entanto, se o campo 20 não existir serão criadas duas linhas em branco entre o campo 10 e o campo 30. Esta situação pode não ser a mais indicada. Neste caso, o que se pretende é uma linha em branco entre cada campo independentemente da existência ou ausência dos campos, não sendo para este caso o formato apresentado o mais apropriado.

[1] Só produz resultado quando utilizado em formatos de impressão

O comando **%** permite solucionar este problema. Tem como finalidade anular todas as linhas em branco (caso existam), que ocorram entre uma linha preenchida e a última linha do texto não vazia, no momento em que o comando é executado. O formato seguinte:

%##V10%##V20%##V30 ...

apresenta apenas uma linha em branco entre cada campo mesmo que um ou mais campos não existam no registo. Outros exemplos são apresentados na figura seguinte:

Exemplo

```
v26^b,x3,v26^a      UNESCO Paris
v26^b/v26^a         UNESCO
                    Paris
v26^b/#v26^a        UNESCO

                    Paris
v26^b,c20,v26^a     UNESCO        Paris
v26^b,####%v26^a    UNESCO




                    Paris
mdl,v26,v30         Paris, Unesco, 1965. p. 247-257, illus.
mdl,v26/v30         Paris, Unesco, 1965.
                    p. 247-257, illus.
```

Comandos de salto de página e coluna[2]

NP*(n)* comando de salto de página

Este comando permite gerar um salto de página. A utilização do parâmetro '(n)' é opcional. Se for mencionado, o comando **NP(n)** só produz efeito se o número de linhas disponíveis na página corrente for menor que ***n***.

Por exemplo, **NP** salta incondicionalmente para a página seguinte, mas NP(4) só faz um salto de página se existirem menos do que 4 linhas na página corrente.

NC*(n)* comando de salto de coluna

Este comando permite gerar um salto de coluna. A utilização do parâmetro '(n)' é opcional. Se for mencionado, o comando **NC(n)** só produz efeito se o número de linhas disponíveis na coluna corrente for menor que **n**.

Por exemplo, NC salta incondicionalmente para a coluna seguinte (ou para a primeira coluna da página seguinte se não existirem mais colunas), mas NC(4) só faz um salto de coluna se existirem menos do que 4 linhas na coluna corrente.

[2] *Estes comandos só produzem efeito quando utilizados em formatos de impressão. Quando utilizados noutro contexto (por exemplo, FST de reformatação), geram caracteres de controlo, e o seu efeito será imprevisível.*

LITERAIS

Uma literal é uma cadeia de caracteres incluída entre os delimitadores respectivos de acordo com o seu tipo. As literais podem ser utilizadas por exemplo, para identificar o conteúdo de um campo (designação, etiqueta, etc.).

Existem três tipos de literais:

Literais condicionais definem o texto que será apresentado apenas se o campo que lhe está associado existir no registo corrente. Se o selector de campo for um comando de subcampo (por exemplo V24^a), o texto da literal só será apresentado se o subcampo indicado existir no campo. Se o selector de campo especificar um campo repetível, o texto só será apresentado apenas uma vez, independentemente do número de ocorrências do campo. As literais condicionais são inscritas entre aspas ("). Por exemplo, "Título:".

Literais repetíveis tal como nas literais condicionais, definem o texto entre delimitadores a ser apresentados apenas se o campo ou subcampo a que está associada existir no registo corrente. Se o campo for repetível, neste caso, a literal será repetida por cada ocorrência do campo. As literais repetíveis são inscritas entre barras verticais (|).

Ex. |Autor:|.

Literais incondicionais definem o texto que será apresentado independentemente da existência de um campo ou subcampo. As literais incondicionais são inscritas entre apóstrofes "'". Por exemplo 'Assuntos:'. As literais incondicionais são sempre apresentadas como sendo um único bloco de texto (isto é, não podem ser repartidas entre duas linhas). A sua dimensão não deve exceder a dimensão da linha disponível, doutra forma, o restante conteúdo será truncado. Para apresentar texto por mais do que uma linha, este deve ser dividido por mais do que uma literal. O alinhamento do texto é possível recorrendo ao comando Cn.

As literais não podem conter os seus próprios delimitadores, isto é, uma literal incondicional não pode conter o caracter ('); no entanto, podem ser incluídas barras verticais (|) ou aspas (").

As literais condicionais e/ou repetíveis são sempre associadas a um campo ou a um subcampo, conforme a sua posição no formato: as literais que antecedem um selector de campo (também chamadas pré-literais) são apresentadas antes do conteúdo do campo, ao passo que as literais que sucedem num selector de campo (também chamados de pós-literais) são apresentadas após o conteúdo do campo.

Se uma **pré-literal** repetível for imediatamente sucedida pelo caracter "+" (isto é |xxx|+), esta é apresentada antes de todas as ocorrências do campo excepto na primeira.

Se uma **pós-literal** repetível for imediatamente precedida pelo caracter "+" (isto é +|xxx|) esta é apresentada depois de todas as ocorrências do campo, excepto na última.

As pré-literais repetíveis e todas as pós-literais são formatadas como se fizessem parte integrante do conteúdo do campo, e se necessário, tomadas em conta no caso de indentação pelos respectivos comandos. As pré-literais condicionais não actuam nos comandos de alinhamento (para tal, caso seja necessário, deve ser usado o comando Cn).

Um determinado campo pode ser associado a mais do que uma literal. Neste caso as várias literais devem obedecer às regras que a seguir se indicam:

Pré-literais

Uma ou mais pré-literais condicionais. Uma pré-literal condicional pode ser seguida por outra, por comandos de espaçamento vertical ou horizontal, comandos de modo, e/ou comandos de escape.

A colocação de comandos entre a primeira pré-literal condicional e o selector de campo associado, transformando-os também em condicionais e só são executados se o campo existir, doutra forma, serão ignorados.

Uma, e apenas uma pré-literal repetível. Caso exista, deve preceder imediatamente o selector de campo associado.

Pós-literais

Uma, e só uma pós-literal repetível, caso exista, deve seguir imediatamente o selector de campo associado. Por exemplo V200^a**|, |**

Uma, e só uma pós-literal condicional, caso exista, deve seguir imediatamente a pós-literal repetível ou o selector de campo associado. Por exemplo V200^a**", "**

As pós-literais não devem ser separadas por vírgulas, e não devem haver vírgulas entre o selector de campo e estas. A vírgula significa o fim da pós-literal associada a um determinado selector de campo.

As literais vazias (isto é, literais sem conteúdo, como por exemplo "" ou | |) são permitidas e podem ser usadas, por exemplo, como pré-literais, para determinar espaçamento vertical condicional, ou como pós-literais, para suprimir a pontuação final que o CDS/ISIS gera quando é activado o modo de dados.

As literais respeitam a conversão de caracteres para maiúsculas se precedidas por um comando de modo.

Exemplos com diferentes tipos de literais são apresentados na figura seguinte.

Selector de campo fictício

Um selector de campo fictício permite definir a apresentação de uma literal condicionada pela presença ou ausência de um determinado campo ou subcampo, sem apresentar o conteúdo do campo associado. Um selector de campo fictício é codificado da seguinte forma:

Dt ou Dt^x ou Nt ou Nt^x

em que:

D ou **N** identificam este como um selector de campo fictício. D indica que todas as literais condicionais associadas devem ser apresentadas (visualizadas ou impressas) apenas se o campo existir.

N indica que estes devem ser apresentados apenas se o campo não existir;

t é o identificador de campo (etiqueta) de controlo para apresentação das literais;

^x é um código delimitador de subcampo opcional. Quando indicado significa que a apresentação das literais é controlada pela existência ou não do subcampo especificado (no caso da inexistência do campo, este implica a ausência do subcampo).

Exemplo

```
'MFN: 'mfn(3)/                  MFN: 004
mdl,"Título: "v24(0,7)          Título: An Electric hygrometer apparatus for measuring water-vapour
                                loss from plants in the field

'MFN: 'mfn(3)/mdl,              MFN: 004
"Título: ",mdu,v24(0,7)         Título: AN ELECTRIC HYGROMETER APPARATUS FOR MEASURING WATER-VAPOUR
                                LOSS FROM PLANTS IN THE FIELD

'MFN: 'mfn(3)/mdu,              MFN: 004
"Título: ",v24(0,7)             TÍTULO: AN ELECTRIC HYGROMETER APPARATUS FOR MEASURING WATER-VAPOUR
                                LOSS FROM PLANTS IN THE FIELD
v70                             Grieve, B.J.Went, F.W.
v70|; |                         Grieve, B.J.; Went, F.W.;
v70+|; |                        Grieve, B.J.; Went, F.W.
|; |v70                        ;Grieve, B.J.; Went, F.W.
|; |+v70                       Grieve, B.J.; Went, F.W.
"Autores"/v70(3,3)+|; |         Autores
                                Grieve, B.J.; Went, F.W.

|(|v70(3,3)|)|                 (Grieve, B.J.)(Went, F.W.)
"(por: ",v70+|; |)"             (por: Grieve, B.J.; Went, F.W.)
mdl,v26                         Paris, Unesco, 1965.
mdl,v26""                       Paris, Unesco, 1965
mdl,v26""/#v99,v30^a            Paris, Unesco, 1965. p. 247-257.
mdl,v26,""/#v44|: |,v30^a       Paris, Unesco, 1965.

                                Methodology of plant eco-physiology: proceedings of the Montpellier
                                Symposium: p. 247-257.
```

Um selector de campo fictício é normalmente precedido pelo menos por uma pré-literal condicional (que pode ser vazia), podendo ser seguida por uma ou várias pré-literais condicionais, comandos de espaçamento horizontal ou vertical, comandos de modo e/ou comandos de escape. Os selectores de campo fictício não podem estar associados a pós-literais.

Alguns exemplos destes comandos são apresentados na seguinte.

Exemplo

```
"[Apenas em Inglês]"n76          [Apenas em Inglês]
"(Anónimo)"n70,v70+|; |          Grieve, B.J.; Went, F.W.
"(Anónimo)"n80,v80+|; |          (Anónimo)
"[Comunicação]"d44               [Comunicação]
"[sem data]"n26^c,v26^c          1965
"[sem data]"n27^c,v27^c          sem data
```

EXPRESSÕES

A linguagem de formatação permite calcular e/ou comparar valores através do uso de expressões. As expressões são definidas para que quando são executadas devolvam um valor que pode ser uma expressão tipo cadeia de caracteres (isto é, o conteúdo de um campo ou de uma literal), e neste caso terá o nome de expressão de cadeia de caracteres; um número, denominada expressão numérica, ou um valor lógico (**Verdadeiro ou Falso**), denominada expressão booleana. O CDS/ISIS dispõe também de um conjunto de funções que, segundo os argumentos definidos, executam uma operação e devolvem um valor. As funções que devolvem um número designam-se funções numéricas, as que devolvem uma cadeia de caracteres são funções de cadeia de caracteres e aquelas que devolvem um valor lógico são funções booleanas. Só as funções de cadeia de caracteres podem ser directamente utilizadas como comandos de formatação. As expressões numéricas podem ser utilizadas nas expressões booleanas ou como argumentos de funções. As expressões e funções booleanas podem ser apenas utilizadas em conjunto com o comando **IF**.

Expressões numéricas

As expressões numéricas são constituídas por operandos que contêm um valor numérico e operadores que determinam os cálculos a executar.

Assim, os operandos que podem ser utilizados neste tipo de expressões são os seguintes:

constantes numéricas tal como 5 18 98.65: as constantes numéricas podem ser representadas como números inteiros, como decimais ou em notação científica como, por exemplo, 1.5E5 (ou seja, 1.5 vezes 10 elevado a 5, isto é, 150000);

funções numéricas tal como VAL(V10), que devolve o valor numérico do campo 10 (ver "Funções numéricas");

MFN devolve o valor do MFN de um registo.

expressão numérica quando utilizada como operando, a expressão deve ser limitada por parênteses curvos, como por exemplo (VAL(V20)-5).

Os operadores válidos são:

- **+** soma (ou + unário)
- **-** subtracção (ou - unário)
- ***** multiplicação
- **/** divisão

Segundo as regras algébricas, na ausência de parênteses, os operadores unários (sinal de + e - respectivamente) são executados em primeiro lugar; as multiplicações e as divisões são executadas antes da adição e da subtracção. Um conjunto de dois ou mais operadores com o mesmo nível, é executado da esquerda para a direita. Podem utilizar-se parênteses para alterar esta ordem. As expressões entre parênteses são calculadas em primeiro lugar antes das expressões externas.

Como os selectores de campo (por exemplo V10 ou V20^a) produzem uma cadeia de caracteres, não podem ser utilizados como operandos numa expressão numérica. A função VAL pode, no entanto, ser utilizada para converter o conteúdo de um campo ou subcampo num valor numérico. Da mesma forma, uma expressão numérica não pode ser directamente visualizada sem primeiro ser convertida numa cadeia de caracteres através da função F.

A versão do CDS/ISIS para IBM PC converte todos os números calculados em vírgula flutuante de precisão simples. Isto permite a definição com uma precisão até a um valor máximo de sete dígitos, como por exemplo, 1.701411E38. A versão para VAX dispõe de números de vírgula flutuante de dupla precisão, o que oferece uma precisão até a um valor máximo de 15 dígitos.

Exemplos de expressões numéricas são apresentados de seguida (em que: MFN=10, v1^a=10, v1^b=20 e v2=30):

Exemplo

```
0.155e+3                        155
1e-3                            0.001
2*3+9                           15
2*(3+9)                         24
10-(4*(2-1))                    6
15*0.001                        0.015
mfn+100                         110
val(v2)+val(v1^a)*7.5           105
(val(v1^a)-val(v1^b))/100       -0.1
```

Expressões de cadeias de caracteres

Este tipo de expressões são constituídas por operandos que são cadeias de caracteres. Como o CDS/ISIS não dispõe de um operador específico para este fim, uma expressão deste tipo consiste sempre num único operando, que pode ser um dos seguintes:

literais incondicionais: tal como 'isto é uma literal incondicional';

selectores de campo: que pode incluir um comando de posição/dimensão (por exemplo: v26^c*2.2);

funções de cadeias de caracteres: tal como S(v24,v25,v26) (ver "Funções de cadeias de caracteres")

Expressões booleanas

As expressões booleanas são utilizadas para determinar se, num conjunto de uma ou mais condições, estas são verdadeiras ou falsas, e determinar o seu valor lógico. Os operandos de uma expressão booleana podem ser os seguintes:

expressão relacional: que compara dois valores e determina se uma relação se verifica (ver a seguir); por exemplo, MFN<10;

função booleana: tal como P(v24), que devolve um valor lógico (ver "Funções booleanas").

Uma expressão relacional permite determinar se uma certa relação entre dois valores se verifica. A fórmula geral de uma expressão relacional é:

expressão-1 *operador-relacional* expressão-2

em que:

expressão-1 pode ser uma expressão quer numérica, quer de cadeia de caracteres.

operador relacional é um dos seguintes:

=	Igual
<>	Diferente de
<	Menor que
<=	Menor ou igual que
>	Maior que
>=	Maior ou igual que
:	Contém (utilizado apenas para as expressões de cadeias de caracteres)

expressão-2 é uma expressão do mesmo tipo da *expressão-1*, isto é, a *expressão-1* e a *expressão-2* devem ser simultaneamente numéricas ou de cadeias de caracteres.

Os operadores relacionais **= <> < <= > >=** têm o seu significado normal quando aplicados a expressões numéricas (dentro dos limites dos valores de precisão numérica definidos em "Expressões numéricas",). Quando se comparam expressões de cadeias de caracteres, aplicam-se as seguintes regras:

1. excepto para o operador **:** (contém), as cadeias de caracteres são comparadas exactamente como ocorrem, isto é, letras maiúsculas e minúsculas são comparadas de acordo com o código ASCII respectivo (por exemplo, A tem um valor menor que a);

2. duas cadeias de caracteres não são consideradas iguais, a não ser que ambas tenham a mesma dimensão. Se duas expressões de cadeias de caracteres tiverem uma dimensão diferente e se os caracteres forem os mesmos, a cadeia mais curta é considerada como inferior em relação à mais longa.

O operador **:** (contém) pesquisa uma cadeia de caracteres (definida pela expressão-2) numa outra cadeia de caracteres (definida pela expressão-1). Se o segundo operando ocorrer em qualquer parte do primeiro, o resultado será considerado verdadeiro. Este operador é indiferente às letras maiúsculas e minúsculas. Uma letra minúscula é considerada de igual valor (código ASCII) à sua equivalente em maiúscula.

Por exemplo, o resultado de:

V10 : 'chemist'

é *Verdadeiro* se, e só se, a palavra *chemist* estiver contida no campo 10, caso contrário o resultado é considerado *Falso.* É de notar que o segundo operando pode ser qualquer conjunto de caracteres sem ordem lógica. Assim a expressão é verdadeira, não apenas se no campo 10 estiver contida a palavra *chemist*, mas também, se existirem as palavras *chemistry*, *biochemist*, *photochemistry*, etc.

Os operadores da expressão booleana podem ser combinados com os seguintes operadores booleanos:

NOT este operador produz um valor **Verdadeiro** se o seu operando for **Falso** e um valor **Falso** se o seu operando for **Verdadeiro**. O operador **NOT** só pode ser utilizado como operador unário, isto significa que se aplica sempre relativamente à expressão booleana que o segue.

AND este operador produz um valor **Verdadeiro** se ambos os operandos forem Verdadeiros. Se um dos operandos for **Falso**, então o resultado é **Falso**.

OR este operador produz um resultado **Verdadeiro** se pelo menos um operador for **Verdadeiro**, caso contrário é **Falso**.

Exemplo

```
mfn=4                          Verdadeiro
not mfn=4                      Falso
not (not mfn=4)                Verdadeiro
v24='plants'                   Falso
v24:'plants'                   Verdadeiro
v24:'PLANTS'                   Verdadeiro
v44.6='method'                 Falso
v44.6='Method'                 Verdadeiro
24:'plants' and v44:'method'   Verdadeiro
```

Durante a execução das expressões booleanas e na ausência de parênteses, o CDS/ISIS executa em primeiro lugar o operador **NOT**, o **AND** e depois o **OR**. No caso da existência de dois ou mais operadores com o mesmo nível, a expressão é executada da esquerda para a direita e neste caso a existência de parênteses determina a prioridade da execução.

Exemplos de expressões booleanas são apresentadas na Figura seguinte:

FUNÇÕES

Uma função processa um valor (chamado valor da função ou valor resultante) que substituirá a função no cálculo da expressão. Uma função pode ter um ou mais argumentos, obrigatoriamente mencionados, que servem para calcular o valor da função. Deste modo, o valor de uma função depende do valor dos argumentos. Estes devem ser separados por vírgulas e limitados por parênteses.

Os argumentos podem ser de três tipos:

formato: um formato do CDS/ISIS, que pode conter qualquer comando válido da linguagem de formatação, excepto para a função REF (ver "REF(expressão, formato)") quando o formato é usado como argumento, é o texto resultante a partir da execução do formato que é passado à função, em vez do formato, propriamente dito.

expressão numérica: quando uma expressão numérica é usada como argumento, é calculado em primeiro lugar, e o valor da expressão é então passado à função.

selector de campo: um selector de campo usado como argumento também pode ser um comando de campo ou subcampo. Não pode conter o comando de posição/dimensão.

As funções disponíveis e os argumentos ou expressões correspondentes, são descritos de seguida e classificados segundo o tipo de valor resultante, que pode ser numérico, uma cadeia de caracteres ou booleano.

FUNÇÕES NUMÉRICAS

Val(<*formato*>)

A função **VAL** devolve como resultado o valor numérico dos seus argumentos. O formato convertido em argumento é um formato do CDS/ISIS que pode conter qualquer comando válido da linguagem de formatação. O CDS/ISIS executa este formato e produz uma cadeia de caracteres. Este é então processado da esquerda para a direita até que um valor numérico seja encontrado (pode ser em notação exponencial). A função VAL devolve o valor numérico convertido para a representação numérica interna do computador destinada à execução de cálculos. Se nenhum valor numérico for identificado, o valor da função será igual a zero. Se o texto produzido tiver mais do que um valor numérico, apenas o primeiro será considerado. Por exemplo, (assumindo que v1^a=10, v1^b=20, v2=30):

Exemplo

```
val('15.79')                    15.79
val(v1)                         10
val(v1^a)                       10
val(v2)                         30
val("19"v1^b)                   1920
val('xxxx7yyyy8zzzz')           7
val('abs. 5.e-4 ml')            0.00058
val('água')                     0
val('Jul-Ago 1985')             0
```

O último exemplo devolve o valor 0 (e não 1985) porque o CDS/ISIS toma em consideração o hífen entre Jul e Ago como sendo o início de um valor numérico negativo, e o A de Ago como o fim. No entanto, o valor extraído é apenas '-', que dá origem a 0. É importante na fase do carregamento dos dados, definir regras para a sua digitação nos campos ou subcampos sobre os quais se pretendem fazer cálculos numéricos.

Rsum(<*formato*>)

A função **RSUM** devolve a soma de um ou mais valores numéricos. O texto produzido pelo argumento é interpretado da esquerda para a direita como na função VAL e todos os valores numéricos contidos são adicionados. O resultado final da soma é o valor da função. Os valores individuais (isolados) devem ser separados por um ou mais caracteres não numéricos e adicionados ao formato indicado como argumento. RSUM pode ser usado para determinar a soma de valores numéricos contidos em todas as ocorrências de um campo repetível. Por exemplo, assumindo que o campo V1 tem 4 ocorrências cujos valores são: 10,20,30,40

Exemplo

```
rsum('10,20,30')                60
rsum(v1|;|)                     10
rsum(v1|,|,'48,3.5')            61.5
```

Rmin(<*formato*>)

A função **RMIN** devolve o valor mínimo de um ou mais valores numéricos. O texto produzido pelo argumento é interpretado da esquerda para a direita como na função VAL, e todos os valores numéricos são considerados. O menor valor numérico é o valor da função. Os valores individuais devem ser separados por um ou mais caracteres não numéricos, inseridos no formato indicado como argumento. RMIN pode ser usado para determinar o valor mínimo contido em todas as ocorrências de um campo repetível. Por exemplo, assumindo o campo 1 com 4 ocorrências; (10,20,30,40):

Exemplo

```
rmin('1,2,-3')                -3
rmin(v1|;|)                   10
rmin(v1|,|,'48,3.5')          3.5
```

Rmax(<*formato*>)

A função **RMAX** devolve o valor máximo de um ou mais valores numéricos. O texto produzido pelo argumento é processado da esquerda para a direita, tal como na função VAL, e todos os valores numéricos são considerados. O maior valor numérico é o valor da função. Os valores individuais devem ser separados por um ou mais caracteres não numéricos, inseridos no formato indicado como argumento. RMAX pode ser usado para determinar o maior número contido em todas as ocorrências de um campo repetível. Por exemplo, assumindo o campo 1 com 4 ocorrências: 10,20,30,40:

Exemplo

```
rmax('1,2,-3')                2
rmax(v1|;|)                   40
rmax(v1|,|,'48,3.5')          48
```

Ravr(<*formato*>)

A função **RAVR** devolve o valor médio (forma aritmética) de um ou mais valores numéricos. O texto produzido pelo argumento é processado da esquerda para a direita, como na função VAL, e todos os valores numéricos são considerados. O valor médio é então determinado e apresentado como sendo o valor da função. Os valores individuais devem ser separados por um ou mais caracteres não numéricos inseridos no formato indicado como argumento. RAVR pode ser usado para determinar o valor médio de todas as ocorrências de um determinado campo repetível.

Por exemplo, assumindo o campo 1 com 4 ocorrências contendo 10,20,30,40:

Exemplo

```
ravr('1,2,-3')                0
ravr(v1|;|)                   25
ravr(v1|,|,'48,3.5')          25.25
```

L(*<formato>***)**
L(*[<format do ficheiro inverso>]<formato da chave >***)**

A função **L** utiliza o texto produzido pelo argumento como um termo de pesquisa no ficheiro inverso e determina o MFN do primeiro apontador (caso exista). Antes de verificar a existência do termo no ficheiro inverso, este é automaticamente convertido para maiúsculas. Se o termo não for encontrado, o valor da função será 0. A função L é normalmente usada em conjunto com a função REF a fim de possibilitar a consulta de tabelas (ver "REF(expressão, formato)", para exemplos acerca do uso da função L).

O formato convertido em argumento é executado tendo em conta o modo de visualização. (MHL, MPL, MDL, etc.). Isto é importante porque a utilização de um modo de visualização incorrecto pode resultar numa pesquisa não sucedida do termo. Como regra geral deve ser utilizado do mesmo modo que na definição da FST do ficheiro inverso.

Devolve o MFN do primeiro apontador (caso exista) utilizando a chave gerada pelo <formato da chave> utilizada para pesquisar no ficheiro inverso. Também se pode obter num ficheiro inverso alternativo indicado em <*formato do ficheiro inverso*>

As chaves são convertidas para caracteres maiúsculos antes de serem pesquisados. O modo de apresentação por defeito é o Mpl. Se for indicado um modo diferente no ficheiro FST, a chave deverá utilizar o modo apropriado. Se a chave não for encontrada, a função devolve **0**. O parâmetro <*formato do ficheiro inverso*> deve ser uma cadeia de caracteres possuindo um termo válido do ficheiro inverso, caso contrário será gerado um erro de sintaxe. Esta função é utilizada juntamente com a função **Ref** de forma a obter o conteúdo dos campos de um registo diferente.

Exemplo

```
if l(v15)<> 0 then |Term: |v15, fi,
ref(l(['books']v1,'-',v2),v10/),
```

FUNÇÕES DE CADEIAS DE CARACTERES

As funções de cadeias de caracteres podem ser usadas, quer como operandos de expressões de cadeias de caracteres, quer como comandos de formatação. Quando usadas como comandos, o valor da função é formatado como se fosse um campo de um registo.

F(*expr-1,expr-2,expr-3*)

A função **F** converte um valor numérico a partir da sua representação interna da vírgula flutuante numa cadeia de caracteres. Os três argumentos são todos expressões numéricas. O primeiro argumento (**expr-1**) é o número a ser convertido. O segundo argumento (**expr-2**) é o número mínimo de caracteres a ser apresentado pela **expr-1** e o terceiro argumento (**expr-3**) é o número de casas decimais. Os segundo e terceiro argumentos são opcionais. No entanto, o segundo argumento (**expr-2**) não pode ser omitido se o terceiro (**expr-3**) estiver presente.

expr-2 define a dimensão mínima, isto é, uma cadeia de caracteres com pelo menos **expr-2** caracteres, e se o valor numérico convertido exigir **expr-2** caracteres ou menos, o resultado será alinhado à direita tendo em consideração essa dimensão. Se o número de caracteres necessários para representar o valor de **expr-1** for maior do que a dimensão exigida, o CDS/ISIS utilizará as posições adicionais necessárias. Neste caso a cadeia de caracteres resultante será maior do que a indicada por **expr-2**. A dimensão do número inclui um caracter à esquerda reservado a um sinal (-).

expr-3 define o número de casas decimais depois da vírgula. Se omitida, o resultado será em notação científica, e do mesmo modo se **expr-2** for omitida, o valor da dimensão assumido por defeito, será de 16 caracteres. Para a conversão de números inteiros e de ponto fixo, se a parte inteira de um número for maior que a conversão em número inteiro, o resultado será substituído por um conjunto de asteriscos (*).

A função **F** pode ser utilizada para alinhar uma coluna de números pelo ponto decimal indicando a dimensão da expressão de saída mais apropriado.

Alguns exemplos da função **F** são apresentados de seguida:

Exemplo

```
f(1)                    1.000000000E+00
f(1,10)                 1.000E+00
f(-1,10,2)              -1.00
f(1,5,2)                1.00
f(1,8,2)                1.00
f(mfn,1,0)              4
f(mfn,2,0)              4
f(mfn,3,0)              4
```

Ref(<*expressão*>,<*formato*>**)**
Ref([<*base de dados*>]<*expressão*>,<*formato*>**)**

Executa o <*formato*> no registo seleccionado pelo parâmetro <*expressão*>. Se o parâmetro <*base de dados*> for definido, outra (ou a mesma) base de dados pode ser referenciada e seleccionado um registo diferente. O parâmetro <*expressão*> pode ser um formato qualquer que devolva o MFN do registo a ser formatado. A função **L** pode ser utilizada para executar uma pesquisa e devolver o MFN.

Exemplo

```
ref(l(v3),v1/,v2/,v3/),
if ref(['USERS']l(['user']v2),v4)='OK' then |Nome: |v10/, fi,
(if p(v99) then ref([v99]1,v30/), fi),
```

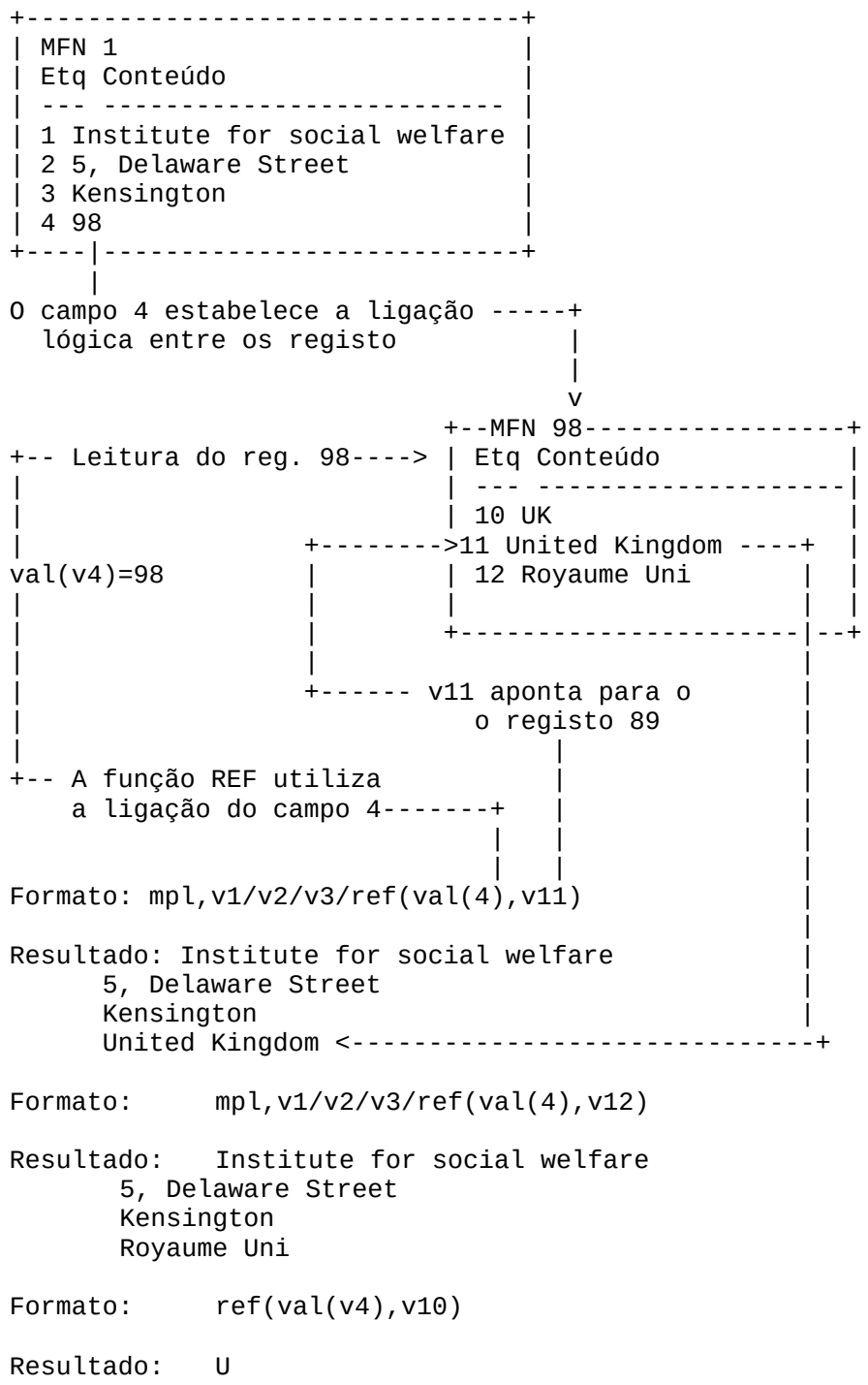
A função **REF** permite extrair dados a partir da indicação do número de registo (MFN) da base de dados, associado ao ficheiro mestre. O primeiro argumento é uma expressão numérica contendo a indicação do MFN do registo a ser seleccionado e o segundo argumento é o formato aplicado ao registo. Se o valor da expressão não corresponder ao MFN de um registo da base de dados, então a função REF devolve um valor nulo (isto é, não são visualizados quaisquer dados). O processamento executado pela função REF é representado na figura seguinte onde o registo a ser formatado é o número 1.

Este exemplo mostra que a função REF é um dispositivo poderoso, que permite ligar entre si dados armazenados em registos diferentes da mesma base de dados, fazendo com que ao serem apresentados os dados pareçam pertencer ao mesmo registo. No primeiro exemplo, os registos 1 e 98 são ligados, pela indicação no campo 4 do registo número 1 do MFN do registo relacionado; este contém o nome do país em inglês e em francês. Pode seleccionar-se a língua de edição especificando no formato a etiqueta pertinente do registo relacionado.

Nalguns casos, a ligação de registos por indicação do MFN pode ser inconveniente do ponto de vista da recolha dos dados. Assim, qualquer erro na digitação do MFN do registo a ligar, resulta na incorrecta visualização dos dados podendo levar algum tempo a determinar qual o registo correcto a que se deve ligar. No exemplo da figura seguinte a ligação dos registos pode ser feita a partir de outro tipo de dados digitados no registo, como por exemplo, um código de país (neste caso 'UK'). Usando o MFN como ligação ao registo do país, este processamento exige a consulta de tabelas ou a pesquisa na base de dados para encontrar os MFN's correspondentes ao código 'UK'; é mais conveniente a digitação do código do que o número de MFN (é mais fácil encontrar 'UK' do que '98'). É possível obter o mesmo resultado na saída dos dados, como ilustra a figura seguinte, organizando a base de modo a ser possível explorar esta facilidade recorrendo à função L (ver "L(formato)").

A função L pesquisa o termo e devolve o MFN respectivo. Pode assim, ser usada para converter uma cadeia de caracteres (por exemplo 'UK') num MFN. Para utilizar a função L é necessário, estabelecer de antemão, uma relação única entre uma cadeia de caracteres e o MFN correspondente. O ficheiro inverso dispõe de um mecanismo que permite esta facilidade (ver "Ficheiro Inverso"). No exemplo, é suficiente inverter o campo 10 dos registos 'país' para estabelecer uma relação única entre o código do país e o MFN correspondente (o conceito de unicidade é importante, porque a função L assume que o termo pesquisado deve remeter apenas para um registo). É preciso pois, construir uma relação única indicando, se necessário, um prefixo.

A figura seguinte ilustra esta técnica, assumindo que o campo 10 do registo 'país' foi invertido com o prefixo 'CP='.

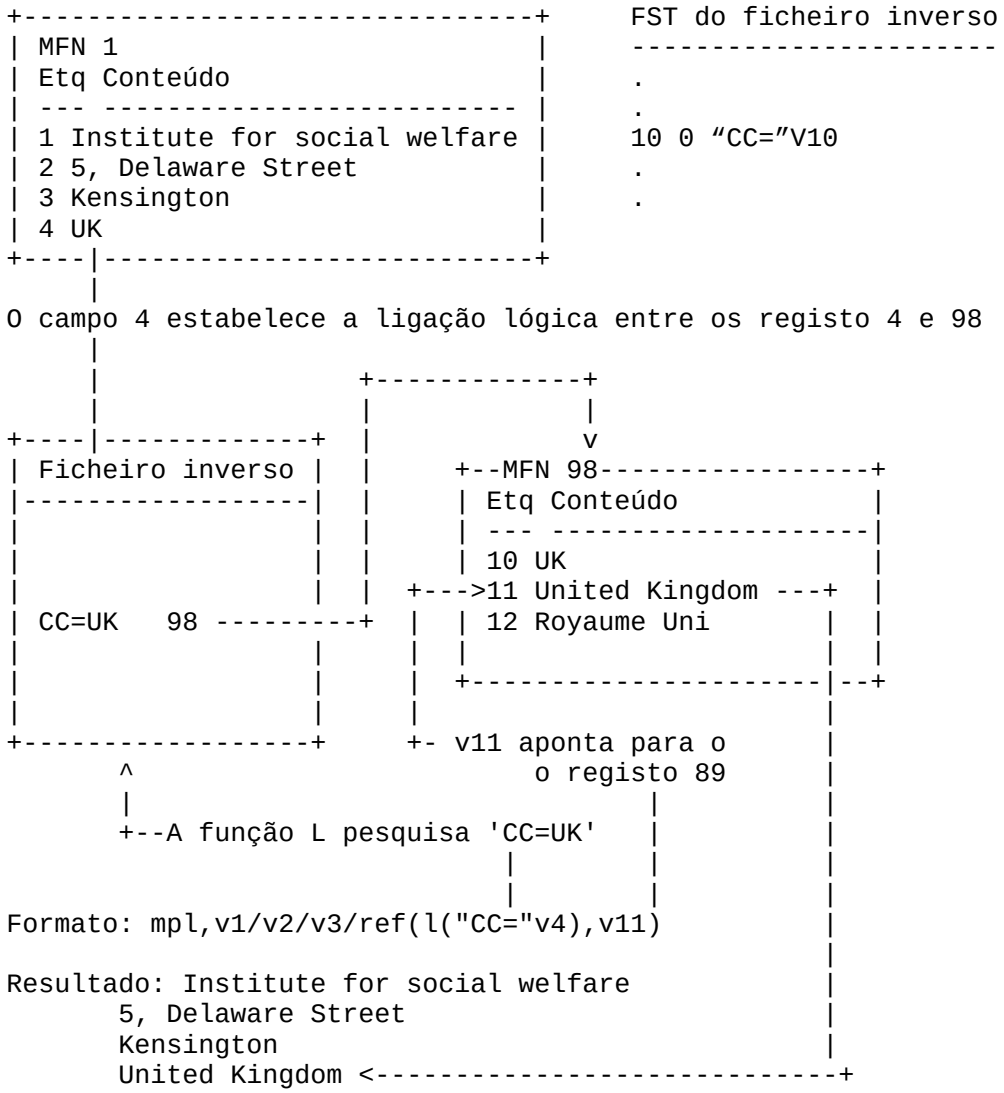



```
Formato:     mpl,v1/v2/v3/ref(val(4),v12)

Resultado:   Institute for social welfare
       5, Delaware Street
       Kensington
       Royaume Uni

Formato:     ref(val(v4),v10)

Resultado:   U
```




```
Formato:     mpl,v1/v2/v3/ref(l("CC="v4),v12)

Resultado:   Institute for social welfare
       5, Delaware Street
       Kensington
       Royaume Uni
```

O CDS/ISIS não faz nenhuma distinção sobre a natureza da relação entre dois registos. Dispõe simplesmente de um mecanismo para ligar registos. Toda as aplicações que se criem devem ser concebidas em função do significado de uma relação, ainda que, disponha de uma utilização correcta da linguagem de formatação e uma boa definição da base de dados. Por exemplo, se um registo bibliográfico tiver que ser ligado a dois registos que por algum motivo lhe estejam associados, é necessário dispor de dois campos diferentes, um para armazenar os dados entre o primeiro e outro para armazenar a ligação entre o segundo. Estas relações são, com efeito, de natureza diferente.

Por outro lado, como o segundo argumento da função REF é um formato, esta função possibilita o uso recursivo para estabelecer relações hierárquicas, como por exemplo, aquelas que seriam necessárias para visualizar as relações hierárquicas de um Thesaurus.

Um formato pode conter tantas chamadas à função REF quantas as exigidas, não ultrapassando o limite da área de trabalho que é de 8000 caracteres. Esta limitação é uma restrição de execução tanto mais que é uma restrição da linguagem de formatação. No entanto, utilizando a memória expandida, esta limitação pode ser ultrapassada a partir da versão 3.0

S(*<formato>***)**

A função **S** devolve o texto gerado pelo argumento. Como anteriormente mencionado, o CDS/ISIS não prevê operadores explícitos para expressões de cadeias de caracteres. A função **S** pode ser usada para implementar a concatenação de cadeias de caracteres. É particularmente útil em expressões booleanas, onde pode ser utilizada para uso de um **OR** (OU) implícito que é mais eficiente (e mais conciso) do que usar o **OR** explícito. Por exemplo, as duas expressões booleanas que se seguem:

s(mdl,v10,v20,v30) : 'água'

v10 : 'água' OR v20 : 'água' OR v30 : 'água'

são equivalentes (são ambas Verdadeiras, se em qualquer um dos campos estiver contida a palavra 'água') mas, a primeira é executada com mais rapidez do que a segunda.

EXECUÇÃO DE PROGRAMAS EXTERNOS CHAMADOS POR UM FORMATO

Num formato pode ser executado qualquer programa em CDS/ISIS Pascal, para executar funções de formatação especiais, necessárias para uma determinada aplicação que não pode ser obtida a partir da linguagem de formatação e das funções já existentes. Estes programas são designados Programas externos (executados por um menu ou formato). O desenvolvimento deste tipo de programas tem como objectivo satisfazer necessidades específicas, embora a sua descrição não pertença ao conjunto dos comandos da linguagem de formatação. No entanto, o CDS/ISIS dispõe de uma forma normalizada para interagir entre os programas externos e a linguagem de formatação.

Do ponto de vista da linguagem de formatação, um programa externo é uma função de cadeia de caracteres tendo um formato como argumento. O argumento é o primeiro a ser executado e o resultado é passado à função. Um programa externo devolve uma expressão de cadeia de caracteres tratada pelo CDS/ISIS, como se fosse um campo de um registo a ser formatado.

Do ponto de vista do CDS/ISIS Pascal, um programa externo é um programa declarado com o atributo [FORMAT] (para mais detalhes ver Manual CDS/ISIS Pascal). Antes de utilizar um programa externo num formato este deve ser previamente compilado.

Exemplo

```
Program SAMPLE(arg: string; lw,occ: real; str: string) [FORMAT];
 begin
    str:=arg;
 end.


&sample('xxx')              xxx
&sample(v26^a)              Paris
&sample(mhl,v24)            An Electric hygrometer apparatus for measuring water-vapour loss
                            from plants in the field
&sample(mhu,v24)(0,5)       AN ELECTRIC HYGROMETER APPARATUS FOR
                                 MEASURING WATER-VAPOUR LOSS FROM
                                 PLANTS IN THE FIELD
```

Uma chamada a um programa externo deve ser indicado num formato da seguinte forma:

&nome(*formato*)

em que:

& indica uma chamada a um programa externo;

nome é o nome do programa CDS/ISIS Pascal a ser executado;

formato é o argumento.

Um comando de indentação pode ser associado durante a chamada do programa externo, o qual o CDS/ISIS aplica ao resultado produzido. A figura seguinte apresenta um exemplo deste tipo de programa, que devolve apenas o argumento como valor da função.

FUNÇÕES BOOLEANAS

P(*<selector de campo>*)

A função P devolve um valor Verdadeiro, se no registo em processamento existir pelo menos uma ocorrência no campo ou subcampo indicado pelo argumento. Por exemplo:

Exemplo

```
p(v24)                          Verdadeiro
p(v26^d)                        Falso
p(v80)                          Falso
```

A(*<selector de campo>*)

A função A devolve um valor Verdadeiro, se no registo em processamento não existir nenhuma ocorrência de campo ou subcampo indicado pelo argumento.

A ausência do campo implica a ausência de todos os subcampos. No entanto, se o selector de campo especificado como argumento indicar um subcampo, a função A devolve um valor verdadeiro se o campo existir, mas se o subcampo estiver ausente, ou se o próprio campo não existir. Por exemplo:

Exemplo

```
a(v24)                          Falso
a(v24^s)                        Verdadeiro
a(v26^d)                        Verdadeiro
a(v80)                          Verdadeiro
```

Comando IF

O comando **IF** permite executar formatos relativamente ao contexto em que se inserem, isto é, apresentar um resultado que pode variar de acordo com o conteúdo do registo formatado.

É representado da seguinte forma:

IF *condição* THEN *formato-1* ELSE *formato-2* FI

em que:

condição	é uma expressão booleana como a definida em "Expressões booleanas";
formato-1	é um formato do CDS/ISIS que será executado se e só se a expressão booleana for verdadeira;
formato-2	é um formato do CDS/ISIS que será executado se e só se a expressão booleana for falsa.

A expressão **ELSE** *formato-2* é opcional e pode ser omitida. As palavras **IF, THEN** e **FI**, são sempre obrigatórias, se bem que o *formato-1* possa ser omitido quando é seguido pela expressão **ELSE** (por exemplo, sempre que não haja nada para ser apresentado se a condição for Verdadeira). Um comando **IF** pode ser ainda representado das seguintes formas:

IF *condição* **THEN** *formato-1* **FI**

IF *condição* **THEN ELSE** *formato-2* **FI**

Não há restrição quanto ao número de comandos a utilizar no formato-1 ou formato-2. O comando IF pode ser utilizado na criação de equações mais complexas, tal como a conjugação de vários IF's encadeados. A palavra FI deve, neste caso, ser usada para fechar cada comando IF (pode equivaler-se cada IF e FI a um par de parênteses). Por exemplo:

```
if p(v1) then v24 else if p(v2) and a(v3) then v5 fi fi
^                     ^                         ^  ^
¦                     +-------------------------+  ¦
+--------------------------------------------------+
```

O comando **IF** é particularmente útil para desenvolver formatos de visualização gerais para bases de dados integradas que incluam diferentes tipos de registos. Para tal, a cada tipo de registo é associada uma marca distintiva (por exemplo, um campo contendo um código de identificação do tipo de registo). Deste modo, fazendo a verificação do registo com o comando IF, podem ser seleccionados através dum simples formato, os dados específicos a apresentar segundo o seu tipo.

Select <*expressão*>
case <*opção-1*>**:** <*formato-1*>
case <*opção -2*>**:** <*formato-2*>
case <*opção-n*>**:** <*formato-n*>
[elsecase <*formato-0*>**]**
endsel

Avalia a <*expressão*> e compara o resultado com cada opção **case** (<opção-1>, <opção-2>…<opção-n>). Se alguma opção for igual ao valor de <*expressão*>, o bloco relativo à linguagem de formatação a que respeita é executado (<*formato-1*>, <*formato-2*>…<*formato-n*>), caso contrário a cláusula ***elsecase*** (caso seja definida) é executada (*formato-0*).

O parâmetro <*expressão*> deve gerar uma cadeia de caracteres ou um valor numérico. Se If <*expressão*> devolver uma *string*, todas os valores das opções nas cláusulas **case** devem ser do mesmo tipo, caso contrário, se a <*expressão*> for numérica, os valores das opções também devem ser numéricos.

Exemplo

```
select s(v5)
  case '1': ,f(val(v5)/2,2,2)/,
  case '2': ,v5/,
  case '3': ,v6,'-',v1/,
  elsecase ,|Error in field v5 = |v5/,
endsel,

select nocc(v7)
  case 0: 'absent'/,
  case 1: 'one occurrence'/,
  case 2: 'two occurrences'/,
  elsecase 'more than 2 occurrences'/,
endsel,
```

Grupos repetíveis

Um grupo repetível consiste num conjunto de comandos de formatação incluídos entre parênteses. O significado de cada comando é idêntico ao descrito anteriormente, salvo para os campos repetíveis usados de uma forma especial.

Um exemplo do tratamento de um campo repetível pode ser indicado da seguinte forma:

```
(“Assuntos: “V606^a+| ; |)
```

Conhecer como o CDS/ISIS processa os campos repetíveis permite compreender melhor o conceito de grupo repetível. Na ausência de qualquer outra indicação, trata todas as ocorrências de um campo repetível de acordo com a ordem de digitação, como sendo uma cadeia de caracteres simples.

Um grupo repetível altera a forma normal como o CDS/ISIS normalmente trata as ocorrências de um campo repetível, processando uma ocorrência a um tempo superior do que todas de uma só vez.

Quando o CDS/ISIS encontra um parênteses aberto de um grupo repetível, procede da seguinte forma:

Um contador de ocorrências é reiniciado a 1;

O formato dentro dos parênteses é então executado de forma a que todos os comandos dentro do grupo apresentem o resultado correspondente ao contador de ocorrência corrente;

Caso não seja apresentado nenhum resultado (isto é, se não houverem mais ocorrências de nenhum dos campos repetíveis dentro do grupo) então o processamento do grupo repetível terminará.

Doutra forma o contador de ocorrências será incrementado de 1 e os passos 2 e 3 são repetidos.

```
+-- MFN 1 ----------------------------+   +-- MFN 2 ----------------------------+
|                                     |   |                                     |
|  Etq             Conteúdo           |   |  Etq             Conteúdo           |
|  ---  ----------------------------- |   |  ---  ----------------------------- |
|    1  INST                          |   |    1  PERS                          |
|   11  Institute for peace research  |   |   21  ^aBrown^bJohn^cMr             |
|   12  12, Moon Street               |   |   22  45-11-23                      |
|   13  Arlington                     |   |   23  <Química><Física>             |
|   14  USA                           |   |   24  1                             |
+-------------------------------------+   +-------------------------------------+
```

```
Formato:     v1,'-',f(mfn,1,0) / mdl,
if v1='INST' then v11/v12(4,4)/v13(4,4)/v14(4,4)/
else v21^c" ",v21^b" ",v21^c" "/"Data de nascimento: "
v22*6.2,'.',v22*3.2,'.19',v22.2/
"Campos de interesse: ",v23/"Empregado: "/d24,
ref(val(v24),v11(4,4)/v12(8,8)/v13(8,8)/v14(8,8)/) fi
```

Resultado (quando formatado o registo 1):

```
INST-1
Institute for peace research
12, Moon Street
Arlington, VA
USA
```

Resultado (quando formatado o registo 2):

```
PERS-2
Mr. John Brown
Data de nascimento: 23.11.1945
Campos de interesse: Química; Física
Empregado:
Institute for peace research
12, Moon Street
Arlington, VA
USA
```

EXTENSÃO DA LINGUAGEM DE FORMATAÇÃO

Os comandos que a seguir se apresentam estão apenas disponíveis no **WWWISIS** e aplicações que tenham como suporte o **ISISDLL**. Os comandos que se seguem podem ser utilizados nos formatos de todos os produtos **BIBLIObase**.

@<nome_do_ficheiro>

Inclui o formato especificado no formato corrente, armazenado no ficheiro indicado. O parâmetro <*nome_do_ficheiro*> pode incluir localização física do ficheiro (unidade e/ou directoria). A sintaxe dos comandos do ficheiro incluído é analisada durante a sua execução. É obrigatório separar o <*nome_do_ficheiro*> por vírgula.

Exemplo

```
@test.pft,v20,
if v1='L' then @large.pft, fi,
if  s(v921,v922):'aa' then @tdocaa.pft, else @isbd.pft, fi
```

Break

Interrompe a execução da formatação de um grupo repetível. Quando utilizado fora de um grupo repetível, interrompe a execução do formato corrente.

A execução do formato é retomada após o fim do grupo repetível. Quando utilizada dentro da função **REF**, a execução continua com o formato a seguir à função.

Exemplo

```
(if iocc > 10 then '10+ occurrences'/,break else v5^n|-|,v5^s,/, fi,),
```

Cat(*<formato>*)

Devolve o conteúdo do ficheiro gerado pelo argumento *formato.*

Exemplo

```
mfn,cat('myfile.html'),
cat('current document'/, ,if v10='c' then 'firstdoc.txt' else 'default.doc' fi),
cat(v101),
```

Comentário /* */

É possível inserir um comentário num formato, limitando o texto à esquerda por **/*** e à direita por ***/**. O texto indicado não é interpretado e pode ser útil para anotar partes do formato mais complexas.

Exemplo

```
/* Título*/    V200^a/
/* esta é um comentário numa linha simples */,
/* este comentário  estende-se  por mais do que uma linha */,
if a(v10) /*and p(v20) */ then v20/ fi,
```

Continue

Processa a ocorrência seguinte de um campo repetível, caso exista.

Exemplo

```
(if iocc = 1 then continue else v10/ fi),
(f(iocc,1,0),'=',v70,continue/),
```

Date, Date(<*palavra_chave*>)

Devolve a data corrente do sistema. Se não forem indicados parâmetros, devolve uma cadeia de caracteres do tipo:

yyyymmdd hhmmss w nnn

que representa:

aaaa	ano
mm	mês
dd	dia
hh	hora
mm	minuto
ss	segundo
w	dia da semana (0-6)
nnn	número de dias desde o 1º de Janeiro do ano corrente

O parâmetro <*palavra_chave*> pode ser uma das seguintes:

DATETIME apresenta a data do sistema no formato europeu mais a hora corrente (**dd/mm/aa hh:mm:ss**)

DATEONLY apresenta o mesmo resultado mas sem a informação relativa à hora

Exemplo

```
date,
'Data: ',date(DATEONLY)/, ,'Hora: ‘,mid(date(DATETIME),10,8)/
```

Getenv(*<formato>*)

Devolve o conteúdo de uma variável de ambiente. Caso o argumento gerado pelo *<formato>* não produza nenhuma variável de ambiente válida, não é devolvido nenhum valor.

Exemplo

```
'Caminho actual: ',getenv('PATH'),
(v1|=|,getenv(v1)/),
```

Iocc

Devolve o número da ocorrência (a partir de 1), caso contrário devolve o valor 0.

Exemplo

```
("Author: "v1/, ,if iocc > 3 then 'et all',break, fi),
(f(iocc,3,0),|.|v10/),
```

Instr(*<formato-1>,<formato-2>*)

Devolve um número indicando a posição inicial de uma cadeia de caracteres gerada pelo formato *<formato-2>* , encontrada na cadeia de caracteres gerada pelo *<formato-1>*. Se esta não for encontrada o valor é **0**. Ambos os argumentos *<formato-1>* e <formato-2> devem gerar cadeias de caracteres, caso contrário provocará um erro de sintaxe. A utilização da função **S** pode ajudar nas situações onde é necessário gerar uma cadeia de caracteres mais complexa.

Exemplo

```
if instr(v5,'ab')>0 then v5/, fi,
if instr(s(|'|v1|'|),v5)>0 then v1, fi,
left(v18,instr(v18,'.')-1),
```

Left(*<formato-1>,<formato-2>*)

Devolve uma nova cadeia de caracteres contendo o conjunto de caracteres mais à esquerda da cadeia original gerada pelo *<formato-1>*. O *<formato-2>* especifica o número actual de caracteres a serem lidos de *<formato-1>* da esquerda para a direita.

Se a cadeia de caracteres gerada por *<formato-2>* for maior que a dimensão definida por *<formato-1>* a função devolve o conteúdo de *<formato-1>*. Se o *<formato-2>* for zero (0) o for um número negativo, a função devolve uma cadeia de caracteres vazia.

Exemplo

```
if left(v1^n,2)='Ma' then v1^n/, fi,
left(v1,instr(v1,'.')-1),
```

Lw(*<int>*)

Define a dimensão da linha de formatação com o valor indicado em *<int>*.

Exemplo

```
if size(v10) > 76 then lw(254), fi,
lw(70),v20/,lw(10),v30/,
```

Mid(*<formato-1>*,*<formato-2>*,*<formato-3>*)

Devolve uma nova cadeia de caracteres contendo um número de caracteres da cadeia original *<formato-1>*. O parâmetro *<formato-3>* indica o número de caracteres a serem lidos de *<formato-1>* e o parâmetro *<formato-2>* indica a posição na cadeia de caracteres onde a extracção deve começar.

Exemplo

```
mid(v2,2,80),
mid(v1,instr(v1,'key'),size(v1))/,
```

Mstname

Devolve o nome da base de dados corrente (caminho completo).

Exemplo

```
'Base de dados corrente: ',mstname/,
ref(['names']l(['names']'ABC'), ,'Base de dados:',mstname/),
```

Newline(<*formato*>)

Define e/ou assume os caracteres por defeito **CR/LF** resultantes do <*formato*>. O parâmetro <*formato*> pode ainda ter sequências de escape tais como:

\r	Carriage return
\n	Line feed

Todos os comandos de salto de linha "****"serão automaticamente substituídos pelo resultado do formato até a uma nova definição do mesmo.

Exemplo

```
newline(if v151='unix' then '\n' else '\r\n' fi,
newline(v301),
newline('<BR>'),
```

Nocc(<*seleccionador de campo*>)

Devolve o número de ocorrências do campo de dados ou do subcampo definido pelo parâmetro <*seleccionador de campo*>. Este parâmetro deve ser apenas um campo e/ou subcampo e nunca o resultado de uma função. Todos os outros componentes associados ao <*seleccionador de campo*>, quando utilizados provocam erros de sintaxe.

Exemplo

```
if nocc(v3)> 10 then 'Demasiadas ocorrências.'/, fi,
'Existem ',f(nocc(v20),2,0),' autors.'/,
```

Npost(<*formato da chave*>)
Npost(<*[formato],chave*>)

Devolve o número total de apontadores de uma chave (indicada pelo parâmetro <*formato da chave*>) no ficheiro inverso. O parâmetro <*formato*> deve gerar uma cadeia de caracteres contendo o nome do ficheiro inverso. O parâmetro <*formato da chave*> contém a chave a ser pesquisada no ficheiro inverso.

Exemplo

```
if npost(v1)> 1 then 'chaves duplicadas ',v1/, fi,
'Existem ',f(npost(v20),3,0),'chaves para ',v20,'. '/,
```

Replace(<*formato-1*>,<*formato-2*>, <*formato-3*>**)**

Devolve uma nova cadeia de caracteres, depois de substituir o <*formato-2*> pelo <*formato-3*>. Se o <*formato-2*> for uma cadeia de caracteres vazia ou não existir no <*formato-1*>, a função devolve a cadeia de caracteres <*formato-1*>.

Se <*formato-3*> for vazio, a cadeia de caracteres <*formato-2*> será excluída do <*formato-1*>. A função **Replace** é sensível às maiúsculas / minúsculas em <*formato-2*> e <*formato-3*>.

Exemplo

```
replace('Mary And John','And','and')/,
if replace(v1^a,'01x','01X')= '894501X' then v1^n/, fi,
replace(s(v304,v333),',',', ')/,
replace(s(if v415='spanish' then v299 else 'none'
fi),v1,v759)/,
```

Right (<*formato-1*>,<*formato-2*>**)**

Devolve uma nova cadeia de caracteres contendo a parte mais à direita da cadeia de caracteres original (<*formato-1*>). O parâmetro <*formato-2*> devolve o número de caracteres a serem lidos de <*formato-1*> da direita para a esquerda.

Se o <*formato-2*>) tiver uma dimensão superior ao <*formato-1*>, a função devolve <*formato-1*>. Se o <*formato-2*>) for 0 ou um número negativo, é devolvida uma cadeia de caracteres vazia.

Exemplo

```
if right(v1^n,1) = 'r' then v1^n/, fi,
right(v65,4)/,
```

Size(<*formato*>**)**

Devolve a dimensão da cadeia de caracteres produzida pelo parâmetro <*formato*>. Este parâmetro deve obrigatoriamente devolver uma cadeia de caracteres, caso contrário será produzido um erro de sintaxe.

Exemplo

```
if size(v10)> 76 then lw(254), fi,
f(size(v10,v20),1,0),
```

Type(<*formato*>)

Devolve uma cadeia de caracteres como se indica:

A se a cadeia de caracteres tiver apenas caracteres alfabéticos (de acordo com a tabela de caracteres alfabéticos, como por exemplo ISISAC.TAB ou espaços.
N se a cadeia de caracteres tiver apenas números (0-9)
X para todos os tipos de caracteres (alfanuméricos)

O parâmetro <*formato*> deve obrigatoriamente devolver uma cadeia de caracteres, caso contrário será produzido um erro de sintaxe.

Exemplo

```
if type(v1)='N' then f(val(v1),3,2)/ else v1/, fi,
if s(type(v1),type(v2),type(v3))<>'AAA' then 'Invalid character type detected'/, fi,
```

Proc(<*formato*>)

Adiciona ou substitui campos de dados no registo corrente. O parâmetro <*formato*> gera o conjunto de comandos específicos que devem ser executados pela função.

As instruções que especificam a actualização são uma cadeia de caracteres contendo os comandos **d**(delete), **a**(add) e **h**(add) que se aplicam ao registo corrente. Os camandos **d**(delete) devem preceder todos os outros comandos indicados.

Os comandos disponíveis são os seguintes:

d* - apaga todos os campos do registo
d<etiqueta do campo> - apaga todas as ocorrências do campo indicado
d<etiqueta do campo>**/**<ocorrência> - apaga a ocorrência indicada do campo
a<etiqueta do campo>**#<**cadeia de caracteres>**#** - adiciona **<**cadeia de caracteres> como uma nova ocorrência do campo indicado
h<etiqueta do campo> <n> <cadeia de caracteres> - adiciona a <cadeia de caracteres>, com <n> bytes de comprimento, como uma nova ocorrência do campo

O delimitador **#** pode ser um caracter qualquer não numérico.

O espaço deve ser utilizador entre os parâmetros <etiqueta do campo><n> <cadeia de caracteres> quando utilizados no comando **h.**

Exemplo

```
proc('d70',|a10#|v70|#|),
proc(if v24*0.4 = 'Tech' then 'd*', fi),
```

Putenv(<*formato*>)

Define uma variável de ambiente ao nível do sistema operativo correspondendo ao valor definido pelo parâmetro <*formato*>. Esta variável é válida apenas no âmbito do processo corrente.

Exemplo

```
putenv('TEST=test'),getenv('TEST'),
```

System(<*formato*>)

Executa o argumento produzido pelo parâmetro <*formato*> como sendo um comando do sistema operativo. O parâmetro <*formato*> deve gerar uma cadeia de caracteres contendo o código a ser executado. O possível resultado obtido pelo comando é enviado directamente para o *standard output*.

Exemplo

```
system('dir'),
if p(v2) then system('type ',v2), fi,
```