make.tol de Sfk.Wrap

Sfk.Wrap es un programa formateador de textos Ascii a lineas de 78 caracteres que lee del clipboard dejando su resultado tambien en el clipboard, tambien puede funcionar con otros textos procedentes de procesadores y editores que se convierten a Ascii al salir del clipboard hacia un fichero de texto. Los textos a los que da formato pueden estar en una o en varias lineas. El numero de caracteres blancos que hay al inicio de la primer linea de entrada se conserva en la primera linea de salida y es heredado por las siguientes lineas siendo este la forma de fijar el sangrado con Sfk.Wrap. El programa funciona considerando: a) que varias lineas, aun con sus saltos de linea al final, constituyen un solo parrafo y b) que los parrafos terminan en el primer punto tras el cual haya un salto de linea, esto es, como si se tratara de un punto y a parte. Sfk.Wrap es util, por ejemplo, para dar formato a los textos de contenido que aparecen dentro de las agendas de post, de las paginas Html, etc. También es util para pasar a ficheros de codigo fuente textos que por proceder de Word u otros procesadores ocupan una sola linea, muy larga, por cada parrafo.

Para que el uso de este conversor sea muy rapido los textos se leen y escriben en el clipboard de forma que se obtiene seleccionando y copiando el texto al que hay que dar formato, por ejemplo, de Notepad2, se ejecuta este conversor que lee el clipboard y escribe el resultado en el clipboard y se pega el texto formateado resultante, en el mismo Notepad 2 en el que el texto estaba seleccionado, cambiandose, por tanto, por uno ya con formato. Si bien ha de hacerse notar que internamente la informacion pasa a traves de ficheros temporales que el usuario no observa.

Para simplificar la programacion este conversor emplea un unico fichero temporal y un solo clipboard por lo que este programa no soporta ejecuciones paralelas. Para el manejo del clipboard se utiliza una herramienta de sfk169.exe de las Swiss File Knife, Sfk, que pueden encontrarse en la direccion http://stahlworks.com/dev/swiss-file-knife.html. El programa sfk169.exe no se invoca directamente, se hace a traves de 2 programas de mandatos que permiten fijar un conjunto de caracteres que no de problemas con las eñes y enmascarar el uso de los ficheros temporales.

Árbol de ficheros

- Sfk.Wrap formateador a lineas de 78 caracteres de textos en Ascii
  - ← make.tol formatea textos a lineas de 78 caracteres a traves del clipboard
  - ← wrap.bat mandato de ejecucion del programa de formateo tipo wrap
  - tol directorios de codigo fuente en el lenguaje de programacion Tol
    - cmm funciones comunes de textos y de gestion del clipboard
      - ← txt.tol funciones de textos, por ejemplo, para realizar wrap
      - ← sfk.tol funciones para manejar sfk169.exe de Swiss File Knife
    - ← inc.tol para la inclusion de ficheros en lenguaje Tol
  - ../../Bin/sfk directorio para mandatos y ejecutables de Swiss File Knife
    - ← File2Clipboard.bat transfiere el contenido de un fichero al clipboard

← Clipboard2File.bat transfiere el contenido del clipboard a un fichero

→ sfk_wrap.pdf documento resumen de funciones del programa de formateo

Declaraciones

Constantes

- `Real linLen`
  Longitud de linea que es la que uso para programar.

Proceso

- `Text cliInp`
  Lee clipboard.
- `Text cliWra`
  Da formato al texto.
- `Real cliOut`
  Guarda en el clipboard.

Constantes

Real linLen

```
////////////////////////////////////////////////////////////////////////////////
Real linLen = 78;
////////////////////////////////////////////////////////////////////////////////
PutDescription("Longitud de linea que es la que uso para programar.", linLen);
////////////////////////////////////////////////////////////////////////////////
```

Text cliInp

```
////////////////////////////////////////////////////////////////////////////////
Text cliInp = SfkReadClipboard("error leyendo el clipboard");
////////////////////////////////////////////////////////////////////////////////
PutDescription("Lee clipboard.", cliInp);
////////////////////////////////////////////////////////////////////////////////
```

Text cliWra

```
////////////////////////////////////////////////////////////////////////////////
Text cliWra = TxtParagraphsWrap(cliInp, linLen);
////////////////////////////////////////////////////////////////////////////////
PutDescription("Da formato al texto.", cliWra);
////////////////////////////////////////////////////////////////////////////////
```

Real cliOut

```
////////////////////////////////////////////////////////////////////////////////
Real cliOut = SfkWriteClipboard(cliWra);
////////////////////////////////////////////////////////////////////////////////
```

```
PutDescription("Guarda en el clipboard.", cliOut);
////////////////////////////////////////////////////////////////////////////
```

txt.tol de Sfk.Wrap

Funciones de texto.

Declaraciones

Funciones de nombre corto

- `Text W(Text txtVal)`
  Retorna un camino en formato Unix convertido a formato Windows/DOS.

Funciones

- `Real TxtCountLeftChars(Text txtInp, Text oneChr)`
  Retorna el numero de caracteres iniciales de txtInp que son iguales al caracter de entrada oneChr.
- `Set TxtLineWrap(Text txtInp, Real linMax, Real cmpCtr)`
  Retorna un conjunto de 2 texto el primero con un máximo de linMax caracteres y el segundo con el resto. Es el resultado de cortar txtInp por el primer blanco que permita que el corte cumpla la condición inicial. Si el texto de entrada es mas corte que linMax retorna un conjunto formado por el texto inicial y la tira vacia. Si el corte es imposible busca el mejor corte posible y si no lo encuentra retorna un conjunto formado por el texto inicial y la tira vacia. Si cmpCtr es true los resultados son compactados. Tambien existe en Tol la funcion Wrap() con ciertas semejanzas, aunque mas a TxtParagraphWrap().
- `Set TxtParagraphWrap(Text txtInp, Real linMax, Real cmpCtr)`
  Retorna un conjunto de textos resultado de aplicar recursivamente la funcion TxtLineWrap(), lo que permite recortar párrafos. Tambien existe en Tol la funcion Wrap() con ciertas semejanzas.
- `Text TxtParagraphsWrap(Text txtInp, Real linMax)`
  Retorna un textos resultado de aplicar la funcion TxtParagraphWrap() a todos sus parrafos. Entiende que cada parrafo termina en un punto y salto de linea.
- `Set TxtTokenizer(Text txtInp, Text tagBrk)`
  Retorna un conjunto de textos resultado de cortar el texto de entrada por un unico tag tagBrk no incluyendo el tag tagBrk dentro de los textos. Se soporta en la funcion Tol Tokenizer() que rompe por un unico caracter. Usa como caracter interno de corte el 7 (bell), esperando que no aparezca.

Funciones de nombre corto

Text W()

```
////////////////////////////////////////////////////////////////////////
Text W(Text txtVal) // Text
////////////////////////////////////////////////////////////////////////
{ Replace(txtVal, "/", "\\") };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un camino en formato Unix convertido a formato Windows/DOS.",
W);
////////////////////////////////////////////////////////////////////////
```

Real TxtCountLeftChars()

```
////////////////////////////////////////////////////////////////////////////
Real TxtCountLeftChars(Text txtInp, // Texto de entrada
                       Text oneChr) // Caracteres iniciales a contar
////////////////////////////////////////////////////////////////////////////
{
  Real numChr = 1;                  // Contador de elementos
  Real txtLen = TextLength(txtInp); // Numero de caracteres

  Real While(And(LE(numChr, txtLen), Sub(txtInp, numChr, numChr) == oneChr),
  {
    Real (numChr:= Copy(numChr) + 1);

    TRUE
  });
  If(Sub(txtInp, numChr, numChr) != oneChr, numChr-1, numChr)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el numero de caracteres iniciales de txtInp que son iguales al
caracter de entrada oneChr.",
TxtCountLeftChars);
////////////////////////////////////////////////////////////////////////////
```

Set TxtLineWrap()

```
////////////////////////////////////////////////////////////////////////////
Set  TxtLineWrap(Text txtInp, // Texto de entrada
                 Real linMax, // Maximo numero de caracteres por linea
                 Real cmpCtr) // Si true entonces compacta
////////////////////////////////////////////////////////////////////////////
{
  Text txtCmp = If(cmpCtr, Compact(txtInp), txtInp);
  Text txtRev = Reverse(txtCmp);
  Real txtLen = TextLength(txtCmp);
  Set  cutSet = If(LE(txtLen, linMax), [[txtCmp, ""]], // Ya esta hecho
  {
    Real blkPos = TextFind(txtRev, " ", txtLen-linMax); // Busca para atras

    If(GE(blkPos, 1),
    {
      SetOfText(Sub(txtCmp, 0,               txtLen-blkPos),
                Sub(txtCmp, txtLen-blkPos+1, txtLen))
    },
    {
      // No se puede cortar
      Real blkBad = TextFind(txtCmp, " ", linMax+1); // Busca hacia adelante

      If(LT(blkBad, 0), [[txtCmp, ""]], // No hay corte posible
      {
        SetOfText(Sub(txtCmp, 0,        blkBad-1), // Hay un mal corte
                  Sub(txtCmp, blkBad+1, txtLen))
      })
    })
  });
  If(cmpCtr, SetOfText(Compact(cutSet[1]),Compact(cutSet[2])), cutSet)
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Retorna un conjunto de 2 texto el primero con un máximo de linMax caracteres
y el segundo con el resto.
Es el resultado de cortar txtInp por el primer blanco que permita que el corte
cumpla la condición inicial.
Si el texto de entrada es mas corte que linMax retorna un conjunto formado
por el texto inicial y la tira vacia.
Si el corte es imposible busca el mejor corte posible y si no lo encuentra
retorna un conjunto formado por el texto inicial y la tira vacia.
Si cmpCtr es true los resultados son compactados.
Tambien existe en Tol la funcion Wrap() con ciertas semejanzas,
aunque mas a TxtParagraphWrap().",
TxtLineWrap);
////////////////////////////////////////////////////////////////////////////
```

Set TxtParagraphWrap()

```
////////////////////////////////////////////////////////////////////////////
Set  TxtParagraphWrap(Text txtInp, // Texto de entrada
                      Real linMax, // Maximo numero de caracteres por linea
                      Real cmpCtr) // Si true entonces compacta
////////////////////////////////////////////////////////////////////////////
{
  Set  setTxt = TxtLineWrap(txtInp, linMax, cmpCtr);
  Set  setIni = SetOfText(setTxt[1]);

  If(setTxt[2]=="", setIni,
                    setIni << TxtParagraphWrap(setTxt[2], linMax, cmpCtr))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto de textos resultado de aplicar recursivamente la funcion
TxtLineWrap(), lo que permite recortar párrafos.
Tambien existe en Tol la funcion Wrap() con ciertas semejanzas.",
TxtParagraphWrap);
////////////////////////////////////////////////////////////////////////////
```

Text TxtParagraphsWrap()

```
////////////////////////////////////////////////////////////////////////////
Text TxtParagraphsWrap(Text txtInp, // Texto de entrada
                       Real linMax) // Maximo numero de caracteres por linea
////////////////////////////////////////////////////////////////////////////
{
  Text txtCls = ReplaceTable(txtInp,
  [[
    [[" \n","\n"]] // Limpiar blancos finales
  ]]);

  Set  parSet = TxtTokenizer(txtCls, ".\n"); // Respetar los parrafos puros
  Set  parCic = EvalSet(parSet, Text(Text inpPar)
  {
    If(inpPar=="", "",
    {
      Real iniWhi = TxtCountLeftChars(inpPar, " "); // Indentacion del parrafo
      Set  linSet = TxtParagraphWrap(inpPar+".", linMax - iniWhi, TRUE);
      Text indent = Repeat(" ", iniWhi);
      Set  linCic = EvalSet(linSet, Text(Text linInp) { indent+linInp+"\n" });
      SetSum(linCic)
    })
  });
  SetSum(parCic)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un textos resultado de aplicar la funcion TxtParagraphWrap() a todos
sus parrafos.
Entiende que cada parrafo termina en un punto y salto de linea.",
```

```
TxtParagraphsWrap);
////////////////////////////////////////////////////////////////////////////
```

Set TxtTokenizer()

```
////////////////////////////////////////////////////////////////////////////
Set TxtTokenizer(Text txtInp, // Texto de entrada
                 Text tagBrk) // Tag por el que se corta
////////////////////////////////////////////////////////////////////////////
{ Tokenizer(Replace(txtInp, tagBrk, Char(7)), Char(7)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto de textos resultado de cortar el texto de entrada por un
unico tag tagBrk no incluyendo el tag tagBrk dentro de los textos.
Se soporta en la funcion Tol Tokenizer() que rompe por un unico caracter.
Usa como caracter interno de corte el 7 (bell), esperando que no aparezca.",
TxtTokenizer);
////////////////////////////////////////////////////////////////////////////
```

sfk.tol de Sfk.Wrap

Funciones de Swiss File Knife Sfk, que se usan la herramienta
http://stahlworks.com/dev/swiss-file-knife.html

Declaraciones

Constantes

- `Text SfkDir`
  Camino de localización de la herramienta Sfk.
- `Text SfkTmp`
  Fichero temporal para que Sfk copie o pegue al Clipboard.

Funciones

- `Text SfkReadClipboard(Text errMsg)`
  Retorna el texto que haya en el clipboard en caso de error retorna errMsg.
- `Real SfkWriteClipboard(Text outTxt)`
  Guarda outTxt en el clipboard y retorna true si lo consigue y false si no.

Constantes

Text SfkDir

```
////////////////////////////////////////////////////////////////////////////
Text SfkDir = W("/Asc/Bin/sfk/");
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Camino de localización de la herramienta Sfk.",
SfkDir);
////////////////////////////////////////////////////////////////////////////
```

Text SfkTmp

```
////////////////////////////////////////////////////////////////////////////
Text SfkTmp = "_skf_clipboard_.tmp";
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Fichero temporal para que Sfk copie o pegue al Clipboard.",
SfkTmp);
////////////////////////////////////////////////////////////////////////////
```

Text SfkReadClipboard()

```
////////////////////////////////////////////////////////////////////////////
Text SfkReadClipboard(Text errMsg) // Texto a retornar si hay error
////////////////////////////////////////////////////////////////////////////
{
  Real runSfk = System(SfkDir+"Clipboard2File.bat " + SfkTmp); // Ejecuta Sfk
  Text inpTxt = If(! runSfk, errMsg, ReadFile(SfkTmp)); // Lee el texto
  Real FileDelete(SfkTmp); // Borra el fichero temporal
```

```
  inpTxt
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el texto que haya en el clipboard en caso de error retorna errMsg.",
SfkReadClipboard);
////////////////////////////////////////////////////////////////////////////
```

Real SfkWriteClipboard()

```
////////////////////////////////////////////////////////////////////////////
Real SfkWriteClipboard(Text outTxt) // Texto a inyectar en el clipboard
////////////////////////////////////////////////////////////////////////////
{
  Text WriteFile(SfkTmp, outTxt); // Guarda el texto en el fichero temporal
  Real runSfk = System(SfkDir+"File2Clipboard.bat " + SfkTmp); // Ejecuta Sfk
  Real FileDelete(SfkTmp); // Borra el fichero temporal
  runSfk
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Guarda outTxt en el clipboard y retorna true si lo consigue y false si no.",
SfkReadClipboard);
////////////////////////////////////////////////////////////////////////////
```

inc.tol de Sfk.Wrap

Inclusion de las funciones comunes

Declaraciones

Inclusiones comunes

- `Set txtInc`

  Funciones de texto.

- `Set sfkInc`

  Funciones de Swiss File Knife.

Set txtInc

```
////////////////////////////////////////////////////////////////////////////
Set  txtInc = Include("cmm/txt.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Funciones de texto.", txtInc);
////////////////////////////////////////////////////////////////////////////
```

Set sfkInc

```
////////////////////////////////////////////////////////////////////////////
Set  sfkInc = Include("cmm/sfk.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Funciones de Swiss File Knife.", sfkInc);
////////////////////////////////////////////////////////////////////////////
```