make.tol de Sfk.Diary

El programa Sfk.Diary es un conversor de textos en Word, que corresponden a anotaciones de actividades para partes de trabajo, con una anotacion por linea, a un formato que se puede pegar en Excel, de forma que cada concepto (empresa, proyecto, dedicacion, fechas, observaciones, etc.) se encaje en su celda correspondiente. El texto se copia de Word al clipboard, a este texto que esta escrito en un lenguaje natural, aunque con ciertas reglas de escritura, este conversor, le da un formato Ascii separador mediantes tabuladores y le añade fórmulas y ambas cosas permite pegarlo en una plantilla Excel de control de trabajos.

El comportamiento de transformacion de este conversor se realiza mediante un conjunto de reglas de reescritura, unas reglas generales y otras reglas son especificas para los diferentes conceptos de las anotaciones. En el fichero Tol cwx.tol pueden consultarse 2 ejemplos de reglas de reescritura de tipo general. Para que el uso de este conversor sea muy rapido la informacion se lee y escribe en el clipboard de forma que se obtiene copiando de Word, se ejecuta este conversor que lee el clipboard y escribe el resultado en el clipboard y se pega dicho resultado en Excel. 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 de este conversor se 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.Diary conversor de notas escritas de Word a Excel a traves del clipboard
  - ← make.tol programa de conversion con reglas de reescritura .doc a .xls
  - tol directorios de codigo fuente escrito en lenguaje Tol
    - cmm directorio de funciones comunes de texto y para el clipboard
      - ← txt.tol funciones para el manejo de textos
      - ← sfk.tol funciones para manejar sfk169.exe de Swiss File Knife
    - app funciones especificas de aplicacion de la conversion de notas
      - ← cwx.tol funciones de conversion del Control notas en Word a eXcel
    - ← inc.tol fichero para la inclusion de ficheros en lenguaje Tol
  - ../../Bin/sfk directorio para los mandatos de la herramienta Swiss File Knife
    - ← File2Clipboard.bat fichero de mandatos que transfiere de un fichero al clipboard
    - ← Clipboard2File.bat fichero de mandatos que transfiere del clipboard a un fichero
  - → sfk_diary.pdf documento resumen de funciones del programa de conversion

Declaraciones

Constantes

- `Set allTre`
  Reglas particulares y comunes.

Proceso

- `Text cwxWrd`
  Leer el clipboard.
- `Text cwxPre`
  Fijar los tiempos.
- `Text cwxTra`
  Traducir.
- `Text cwxXls`
  Convertir a Excel con tabs.
- `Real cliOut`
  Guarda en el clipboard.

Constantes

Set allTre

```
////////////////////////////////////////////////////////////////////////////
Set  allTre = RulTre << CwxTre;
////////////////////////////////////////////////////////////////////////////
PutDescription("Reglas particulares y comunes.", allTre);
////////////////////////////////////////////////////////////////////////////
```

Text cwxWrd

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

Text cwxPre

```
////////////////////////////////////////////////////////////////////////////
Text cwxPre = CwxTimeFix(cwxWrd);
////////////////////////////////////////////////////////////////////////////
PutDescription("Fijar los tiempos.", cwxPre);
////////////////////////////////////////////////////////////////////////////
```

Text cwxTra

```
////////////////////////////////////////////////////////////////////////////
Text cwxTra = CwxTranslate(cwxPre, allTre);
```

```
////////////////////////////////////////////////////////////////////////////
PutDescription("Traducir.", cwxTra);
////////////////////////////////////////////////////////////////////////////
```

Text cwxXls

```
////////////////////////////////////////////////////////////////////////////
Text cwxXls = CwxExcel(cwxTra);
////////////////////////////////////////////////////////////////////////////
PutDescription("Convertir a Excel con tabs.", cwxXls);
////////////////////////////////////////////////////////////////////////////
```

Real cliOut

```
////////////////////////////////////////////////////////////////////////////
Real cliOut = SfkWriteClipboard(cwxXls);
////////////////////////////////////////////////////////////////////////////
PutDescription("Guarda en el clipboard.", cliOut);
////////////////////////////////////////////////////////////////////////////
```

txt.tol de Sfk.Diary

Funciones de texto.

Declaraciones

Funciones de nombre corto

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

Funciones

- `Set Txt2Set(Text txtInp, Text sepTok)`
  Retorna un conjunto a partir de un texto txtInp, troceandolo por un separador sepTok y dependiendo de ctrFun puede, en este orde, si C compactar, si N eliminar los texto nulos, si U retornar elementos unicos y si S retornar el set ordenado.
- `Set TxtForChr(Text inpTxt, Code funChr)`
  Retorna el conjunto resultado de aplicar la funcion funChr(Text oneChr) a todos los caracteres del texto de entrada txtInp. Retorna un Set a imagen de las funciones basicas For() y EvalSet().
- `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().
- `Text TxtReplaceTree(Text txtInp, Set setTre, Real numCic)`
  Equivalente a la funcion ReplaceTable() que en vez de recibir una tabla de de filas con 2 textos (viejo, nuevo) recibe un conjunto de filas formadas por un conjunto de textos que se cambian por un solo texto de destino.
- `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);
////////////////////////////////////////////////////////////////////////////
```

Set Txt2Set()

```
////////////////////////////////////////////////////////////////////////////
Set Txt2Set(Text txtInp, // Texto de entrada
            Text sepTok) // Elemento separador
////////////////////////////////////////////////////////////////////////////
{
  Set setSep = TxtTokenizer(txtInp, sepTok);
  Set setCmp = EvalSet(setSep, Text(Text eleTxt) { Compact(eleTxt) });
  setCmp
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto a partir de un texto txtInp, troceandolo por un separador
sepTok y dependiendo de ctrFun puede, en este orde, si C compactar, si N
eliminar los texto nulos, si U retornar elementos unicos y si S retornar el
set ordenado.",
Txt2Set);
////////////////////////////////////////////////////////////////////////////
```

Set TxtForChr()

```
////////////////////////////////////////////////////////////////////////////
Set TxtForChr(Text inpTxt, // Texto de entrada
              Code funChr) // Funcion tipo Anything(Text oneChr)
////////////////////////////////////////////////////////////////////////////
{
  Real lenTxt = TextLength(inpTxt);
  For(1, lenTxt, Anything(Real posTxt) { funChr(Sub(inpTxt,posTxt,posTxt)) })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el conjunto resultado de aplicar la funcion funChr(Text oneChr)
a todos los caracteres del texto de entrada txtInp.
Retorna un Set a imagen de las funciones basicas For() y EvalSet().",
TxtForChr);
////////////////////////////////////////////////////////////////////////////
```

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

Text TxtReplaceTree()

```
////////////////////////////////////////////////////////////////////////////
Text TxtReplaceTree(Text txtInp, // Texto de entrada
                    Set  setTre, // Conjunto de filas N:1
                    Real numCic) // Numero de ciclos, 0->hasta fin cambios
////////////////////////////////////////////////////////////////////////////
{
  Set  expAll = EvalSet(setTre, Set(Set oneTre) // Expandir todos los arboles
  {
    Set  setOld = oneTre[1]; // Viejos textos a reemplazar
    Text txtNew = oneTre[2]; // Texto de destino
    Set  expTre = EvalSet(setOld, Set(Text txtOld) { [[txtOld, txtNew]] })
  });

  Set  repTab = BinGroup("<<", expAll); // De conjunto de conjuntos a tabla

  ReplaceTable(txtInp, repTab, numCic)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Equivalente a la funcion ReplaceTable() que en vez de recibir una tabla de
de filas con 2 textos (viejo, nuevo) recibe un conjunto de filas formadas
por un conjunto de textos que se cambian por un solo texto de destino.",
TxtReplaceTree);
////////////////////////////////////////////////////////////////////////////
```

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.Diary

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

cwx.tol de Sfk.Diary

Funciones de conversion del Control actividades en Word a eXcel. Marcadores ¦ 166 (no) y · 183, sin constantes por claridad del codigo.

Declaraciones

Constantes

- `Real CwxWra`
  Caracteres de las observaciones.
- `Real CwxPix`
  Pixeles de las observaciones.
- `Text CwxEtc`
  Indica que se ha trucado el texto al ser largo.
- `Text CwxTab`
  Tabulador para saltas celdas en Excel.
- `Text CwxWee`
  Codigo Excel del nombre del dia de la semana.
- `Text CwxYea`
  Codigo Excel del año con punto y el numero de mes.
- `Text CwxMth`
  Codigo Excel del nombre completo del mes.
- `Text CwxDay`
  Hoy si es tarde o ayer si ahora es temprano.

Variables de control

- `Set CwxTre`
  Reglas de traduccion particularizables.

Funciones

- `Set CwxGetLines(Text inpTxt)`
  Equivalente a Tokenizer() pero retorna los textos compactados y no vacios.
- `Text CwxTimeIni(Text inpTab)`
  Si hay una hora de inicio al principio la elimina.
- `Text CwxClsEndDot(Text inpLin)`
  Si hay un punto al final del texto lo elimina.
- `Text CwxTimeEnd(Text inpLin)`
  Pone un marcador en la duracion de la actividad que hay al final del texto y si no hay duracion asume un cuarto de hora.
- `Text CwxTimeFix(Text inpTxt)`
  Trabaja sobre los tiempos y sobre los signos de puntuacion.

- `Text CwxTranslate(Text inpLin, Set cwxTra)`
  Aplica todas las reglas de traduccion de las anotaciones de actividad.
- `Real CwxArial8CharWidth(Text oneChr)`
  Retorna el ancho en pixels de un caracter en Arial de tamaño 8.
- `Real CwxArial8LineWidth(Text oneLine)`
  Retorna el ancho en pixels de un texto en Arial de tamaño 8.
- `Set CwxLineWrap(Text obsFld)`
  Version de la funcion TxtLineWrap() que realiza un mejor ajuste cuando el tipo de letra no es de ancho fijo sino proporcional como la Arial. Marca el texto con puntos supensivos cuando lo trunca.
- `Set CwxObservations(Text obsFld)`
  Retorna una observacion con la mayuscula inicial y ajustado el texto al ancho en pixels de la columna.
- `Text CwxErrorRow(Text inpLin)`
  Retorna una fila Excel para el caso de error.
- `Text CwxExcel(Text inpTxt)`
  Retorna un texto convertido en filas Ascii con tabs para pegar en Excel.

Constantes

Real CwxWra

```
////////////////////////////////////////////////////////////////////////////
Real CwxWra = 105;
////////////////////////////////////////////////////////////////////////////
PutDescription("Caracteres de las observaciones.", CwxWra);
////////////////////////////////////////////////////////////////////////////
```

Real CwxPix

```
////////////////////////////////////////////////////////////////////////////
Real CwxPix = 533;
////////////////////////////////////////////////////////////////////////////
PutDescription("Pixeles de las observaciones.", CwxPix);
////////////////////////////////////////////////////////////////////////////
```

Text CwxEtc

```
////////////////////////////////////////////////////////////////////////////
Text CwxEtc = "…";
////////////////////////////////////////////////////////////////////////////
PutDescription("Indica que se ha trucado el texto al ser largo.", CwxEtc);
////////////////////////////////////////////////////////////////////////////
```

Text CwxTab

```
////////////////////////////////////////////////////////////////////////////
```

```
Text CwxTab = Char(9);
////////////////////////////////////////////////////////////////////////
PutDescription("Tabulador para saltas celdas en Excel.", CwxTab);
////////////////////////////////////////////////////////////////////////
```

Text CwxWee

```
////////////////////////////////////////////////////////////////////////
Text CwxWee = Replace("=TEXTO(INDIRECTO('D' &FILA());'dddd')",  "'", "\"");
////////////////////////////////////////////////////////////////////////
PutDescription("Codigo Excel del nombre del dia de la semana.", CwxWee);
////////////////////////////////////////////////////////////////////////
"
```

Text CwxYea

```
////////////////////////////////////////////////////////////////////////
Text CwxYea = Replace("=TEXTO(INDIRECTO('D' &FILA());'aa.mm')", "'", "\"");
////////////////////////////////////////////////////////////////////////
PutDescription("Codigo Excel del año con punto y el numero de mes.", CwxYea);
////////////////////////////////////////////////////////////////////////
"
```

Text CwxMth

```
////////////////////////////////////////////////////////////////////////
Text CwxMth = Replace("=TEXTO(INDIRECTO('D' &FILA());'mmmm')",  "'", "\"");
////////////////////////////////////////////////////////////////////////
PutDescription("Codigo Excel del nombre completo del mes.", CwxMth);
////////////////////////////////////////////////////////////////////////
"
```

Text CwxDay

```
////////////////////////////////////////////////////////////////////////
Text CwxDay = FormatDate(If(Hour(Now)>=18, Today, Succ(Today, C, -1)),
                         "%c%d/%m/%Y"); //
////////////////////////////////////////////////////////////////////////
PutDescription("Hoy si es tarde o ayer si ahora es temprano.", CwxDay);
////////////////////////////////////////////////////////////////////////
```

Variables de control

Set CwxTre

```
////////////////////////////////////////////////////////////////////////
Set  CwxTre = [[
   [[ [["realizado¦", "Realizado¦"]], "¦" ]],
   [[ [["¦ ", "¦,", "¦;", "¦.", " ¦", ",¦", ";¦", ".¦"]], "¦" ]] ]];
////////////////////////////////////////////////////////////////////////
PutDescription("Reglas de traduccion particularizables.", CwxTre);
////////////////////////////////////////////////////////////////////////
```

Set CwxGetLines()

```
////////////////////////////////////////////////////////////////////////////
Set CwxGetLines(Text inpTxt) // Texto de entrada
////////////////////////////////////////////////////////////////////////////
{
  Set setTok = Tokenizer(inpTxt, "\n");
  Set cutCic = EvalSet(setTok, Text(Text txtTok)
  {
    Real tabPos = TextFind(txtTok, CwxTab);
    Real cutPos = If(tabPos >= 1, tabPos + 1, 1);
    Text cutTxt = Sub(txtTok, cutPos, TextLength(txtTok));
    Compact(cutTxt)
  });
  Select(cutCic, Real(Text inpTok) { inpTok != "" })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Equivalente a Tokenizer() pero retorna los textos compactados y no vacios.",
CwxGetLines);
////////////////////////////////////////////////////////////////////////////
```

Text CwxTimeIni()

```
////////////////////////////////////////////////////////////////////////////
Text CwxTimeIni(Text inpTab) // Linea de entrada
////////////////////////////////////////////////////////////////////////////
{
  Text inpLin = Sub(inpTab, TextFind(inpTab, CwxTab)+1, TextLength(inpTab));
  Real linLen = TextLength(inpLin);
  Text horSep = Sub(inpLin, 3, 3); // Para controlar si hay hora inicial
  Real cutPos = If(horSep == ":", 7, 1); // Con o sin hora inicial
  Sub(inpLin, cutPos, linLen)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Si hay una hora de inicio al principio la elimina.",
CwxTimeIni);
////////////////////////////////////////////////////////////////////////////
```

Text CwxClsEndDot()

```
////////////////////////////////////////////////////////////////////////////
Text CwxClsEndDot(Text inpLin) // Linea de entrada ya compactada
////////////////////////////////////////////////////////////////////////////
{
  Real linLen = TextLength(inpLin);
  Text dotCtr = Sub(inpLin, linLen, linLen); // Controla punto final
  If(dotCtr != ".", inpLin, Sub(inpLin, 1, linLen-1))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Si hay un punto al final del texto lo elimina.",
CwxClsEndDot);
////////////////////////////////////////////////////////////////////////////
```

Text CwxTimeEnd()

```
////////////////////////////////////////////////////////////////////////////
Text CwxTimeEnd(Text inpLin) // Linea de entrada ya compactada
////////////////////////////////////////////////////////////////////////////
{
  Real linLen = TextLength(inpLin);
  Real hasTim = And(Text Sub(inpLin,linLen,  linLen)   <: [["0","5"]],
                    Text Sub(inpLin,linLen-1,linLen-1) <: [["0","2","5","7"]];
                    Text Sub(inpLin,linLen-2,linLen-2) == ",");
  If(!hasTim, inpLin+"¦0,25",
```

```
     Sub(inpLin, 1, linLen-6)+"¦"+Sub(inpLin, linLen-3, linLen))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Pone un marcador en la duracion de la actividad que hay al final del texto y
si no hay duracion asume un cuarto de hora.",
CwxTimeEnd);
////////////////////////////////////////////////////////////////////////////
```

Text CwxTimeFix()

```
////////////////////////////////////////////////////////////////////////////
Text CwxTimeFix(Text inpTxt)
////////////////////////////////////////////////////////////////////////////
{
  Set setLin = CwxGetLines(inpTxt);
  Set cicLin = EvalSet(setLin, Text(Text inpLin)
  {
    Text timIni = CwxTimeIni(inpLin);   // Quita las horas de inicio
    Text endDot = CwxClsEndDot(timIni); // Quita los puntos finales
    Text timEnd = CwxTimeEnd(endDot);   // Separa y pone el tiempo dedicado

    "·"+timEnd+"\n"
  });
  Replace(SetSum(cicLin), ";", ",") // Une y unifica ; x ,
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Trabaja sobre los tiempos y sobre los signos de puntuacion.",
CwxTimeFix);
////////////////////////////////////////////////////////////////////////////
```

Text CwxTranslate()

```
////////////////////////////////////////////////////////////////////////////
Text CwxTranslate(Text inpLin, // Texto de entrada
                  Set  cwxTra) // Reglas de traduccion
////////////////////////////////////////////////////////////////////////////
{
  Text cwxRep = TxtReplaceTree(inpLin, cwxTra, 0); // Traducir
  Replace(cwxRep, "·", "¦¦¦")   // Poner vacios lo no traducido
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Aplica todas las reglas de traduccion de las anotaciones de actividad.",
CwxTranslate);
////////////////////////////////////////////////////////////////////////////
```

Real CwxArial8CharWidth()

```
////////////////////////////////////////////////////////////////////////////
Real CwxArial8CharWidth(Text oneChr) // Un caracter + su separacion
////////////////////////////////////////////////////////////////////////////
{
  Case(
    oneChr<:[["i","l","|",".",",",";",":","!"]],                          1,
    oneChr<:[["j","í"," "]],                                               2,
    oneChr<:[["[","]","{","}","(",")","/","\\"]],                          3,
    oneChr<:[["f","t","I","Í","r","-","*"]],                               3,
    oneChr<:[["x","y","c","s","J","k","v","=","<",">","+","F","Z","T"]],   5,
    oneChr<:[["a","b","d","e","g","h","n","ñ","o","p","q","u","L"]],       5,
    oneChr<:[["0","1","2","3","4","5","6","7","8","9","?"]],               5,
    oneChr<:[["A","Á","E","É","U","Ú"]],                                   6,
    oneChr<:[["X","B","K","P","S","V","C","D","H","N","Ñ","R"]],           6,
    oneChr<:[["Y","G","O","Ó","Q","M","m",CwxEtc]],                        7,
    oneChr<:[["w","%"]],                                                   9,
```

```
    oneChr<:[[""]],                                                    10,
    oneChr<:[["w"]],                                                   11,
    TRUE, /* á, é, ó, ú, ... */                                        5
  )+1
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el ancho en pixels de un caracter en Arial de tamaño 8.",
CwxArial8CharWidth);
////////////////////////////////////////////////////////////////////////
```

Real CwxArial8LineWidth()

```
////////////////////////////////////////////////////////////////////////
Real CwxArial8LineWidth(Text oneLine) // Una linea
////////////////////////////////////////////////////////////////////////
{ SetSum(TxtForChr(oneLine, CwxArial8CharWidth)) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el ancho en pixels de un texto en Arial de tamaño 8.",
CwxArial8CharWidth);
////////////////////////////////////////////////////////////////////////
```

Set CwxLineWrap()

```
////////////////////////////////////////////////////////////////////////
Set CwxLineWrap(Text obsFld)
////////////////////////////////////////////////////////////////////////
{
  Set  setWra = TxtLineWrap(obsFld, CwxWra, TRUE);
  Text obsWra = setWra[1];
  If(obsWra==obsFld, SetOfText(obsWra, ""), // Cabe
  {
    Real linWid = CwxArial8LineWidth(obsWra); // Ajuste Arial 8 proporcional
    Real addChr = Round(Max(CwxPix - linWid - 7, 0) / 6);
    Set  newWra = TxtLineWrap(obsFld, TextLength(obsWra)+addChr, TRUE);
    SetOfText(newWra[1] + CwxEtc, newWra[2]) // No cabe
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Version de la funcion TxtLineWrap() que realiza un mejor ajuste cuando el
tipo de letra no es de ancho fijo sino proporcional como la Arial.
Marca el texto con puntos supensivos cuando lo trunca.",
CwxLineWrap);
////////////////////////////////////////////////////////////////////////
```

Set CwxObservations()

```
////////////////////////////////////////////////////////////////////////
Set CwxObservations(Text obsFld)
////////////////////////////////////////////////////////////////////////
{
  Text obsTUp = ToUpper(Sub(obsFld,1,1)) + Sub(obsFld,2,TextLength(obsFld));
  CwxLineWrap(obsTUp)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una observacion con la mayuscula inicial y ajustado el texto al
ancho en pixels de la columna.",
CwxObservations);
////////////////////////////////////////////////////////////////////////
```

Text CwxErrorRow()

```
////////////////////////////////////////////////////////////////////////////
Text CwxErrorRow(Text inpLin)
////////////////////////////////////////////////////////////////////////////
{ Repeat(CwxTab, 8)+inpLin+"\n" };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una fila Excel para el caso de error.",
CwxErrorRow);
////////////////////////////////////////////////////////////////////////////
```

Text CwxExcel()

```
////////////////////////////////////////////////////////////////////////////
Text CwxExcel(Text inpTxt)
////////////////////////////////////////////////////////////////////////////
{
  Set setLin = CwxGetLines(inpTxt); // Si se quitan las vacias
  Set setSor = Sort(setLin, Compare);

  Set cicLin = EvalSet(setSor, Text(Text inpLin)
  {
    Set  setFld = Txt2Set(inpLin, "¦"); // No se quitan los vacios
    If(Card(setFld) != 5, CwxErrorRow(inpLin),
    {
      Set  obsSet = CwxObservations(setFld[4]);
      Text obsWra = obsSet[1] + If(obsSet[2]=="", "", CwxTab+obsSet[2]);

      setFld[1] + CwxTab + // Entidad
      setFld[2] + CwxTab + // Codigo
      setFld[3] + CwxTab + // Proyecto
      CwxDay    + CwxTab + // Fecha en formato d/m/y
      setFld[5] + CwxTab + // Horas
      CwxWee    + CwxTab + // Dia de la semana
      CwxYea    + CwxTab + // Año con su mes en numero
      CwxMth    + CwxTab + // Nombre del mes
      obsWra    + "\n"     // Observaciones
    })
  });
  SetSum(cicLin)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un texto convertido en filas Ascii con tabs para pegar en Excel.",
CwxExcel);
////////////////////////////////////////////////////////////////////////////
```

inc.tol de Sfk.Diary

Inclusion de las funciones comunes y de aplicacion

Declaraciones

Inclusiones comunes

- `Set txtInc`
  Funciones de texto.
- `Set sfkInc`
  Funciones de Swiss File Knife.

Inclusiones de aplicación

- `Set pdbCwx`
  De Control en Word a eXcel.

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

Set pdbCwx

```
////////////////////////////////////////////////////////////////////////////
Set  pdbCwx = Include("app/cwx.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("De Control en Word a eXcel.", pdbCwx);
////////////////////////////////////////////////////////////////////////////
```