make.tol de iForense

Programa constructor del sitio web del dominio forense.info dedicado a contenidos de formación en informática forense en donde se presentan 3 cursos de experto, especialista y máster en informática forense y pericial. Los contenidos que emplea son posts con 3 niveles de importancia (status), que se organizan en un directorio, que se denomina agenda, este directorio tiene contiene varios ficheros, cada fichero contiene varios posts y cada post pertenece a una o varias clases (categorías) de posts. Dentro de este directorio de agenda, sin pertenecer a ella, hay un glosario de términos de informática forense que permiten ilustrar los contenidos del sitio de forma no determinista.

Este programa para la creación del sitio web forense.info: a) está basado en un macro-expansor de Tol en Html a doble nivel, b) crea índices de artículos e ilustraciones de forma automática, c) rellena automáticamente la descripción y la lista de palabras clave de cada página de una forma personalizada para ella (campos meta) y d) sus funciones principales son crear todos los artículos de contenido, crear todas las páginas por categorías, crear la homepage y las páginas absolutas y de control de errores, crear el mapa del sitio web en XML y poner los contenidos online.

Los posts de la agenda de contenidos pueden ser de los 3 siguientes niveles: a) anulados que no salen (se le denomina nivel A), b) bajos que se publican sólo dentro de sus clases, pero no con árticulo propio (nivel B) y c) comunes que se publican dentro de su clase y con en su propio artículo (nivel C), la mayoría de estos post tienen este nivel, por lo que es el nivel por defecto.

Este programa para la construcción del sitio web forense.info utiliza un directorio de agenda de posts, dentro de este directorio los posts se estructuran en varios ficheros, usualmente un fichero para cada clase de posts, lo que permite organizar los post por su tipo de contenido, por ejemplo, toda la bibliografía esta en el mismo fichero. Estos ficheros de posts tienen de extensión .age. En este mismo directorio se guarda un glosario de términos de informática forense y pericial con la que se complementa la información publicada en los posts. Los términos publicados junto con los posts bien pueden ser generales o bien ser relativos a la información concreta que proporciona en cada post.

Los posts pueden pertenecer a múltiples clases, también llamadas categorías, y se crean páginas Html de posts, artículos, en un directorio llamado articulos y de conjuntos de post para cada clase, en un directorio llamado categorias. Las páginas de artículos se generan en este programa con la opción art, las páginas con los artículos agrupados por categorías con la opción cat, las páginas por defecto de cada directorio, las absolutas y las de control de errores con la opción rot y el mapa del sitio web en Xml con la opción xsm, siglas de Xml site map.

Este programa y la creación de este sitio web sobre informática forense está basado en un macro-expansor a doble nivel de Tol embebido en Html, donde la semilla de Html contiene Tol embebido y los post también pueden contener Tol embebido, por lo que dentro de la primera expansión, la de la semilla, se pueden realizar otras expansiones, que son las contenida en los post. La macro-espansión de tol a doble nivel permite que los post contengan código en lenguaje Html y código en lenguaje Tol, ello permite, por ejemplo, crear índices o incluir definiciones de los términos del glosario de informática forense dentro de cada post. Para el manejo de los términos del glosario de informática forense del que dispone este programa, en su libreria glo.tol, se emplea de forma básica el álgebra de conjuntos de Tol, para evitar, en la medida de lo posible, la repetición de la ocurrencia de la definición de los términos.

Este programa sólo escribe los ficheros de páginas Html que son diferentes a los ya creados en ejecuciones anteriores, de forma que no haya que enviar y poner online todo el conjunto de páginas sino las modificadas en fecha más reciente al último log de envío. Este control de recencia lo realiza la opción fup, que son las siglas de ftp update, frente a la opción ftp que genera ficheros de mandatos de envío con todo el contenido del web. Nótese que las opciones fup y ftp sólo generan ficheros de mandatos ftp de envío y luego se pueden ejecutar estos ficheros con la opcion snd. Aunque también ha de observarse que la inclusión de términos del glosario es tan variada que en casi todas las ejecuciones las páginas Html generadas son diferentes a las generadas en la anterior ejecución por lo que se produce su envío.

Los comentarios del código de este programa están realizados utilizando unas veces el español no acentuado dentro del código y otras veces el inglés. Se ha comprobado el funcionamiento de este programa para las versiones de Tol 1.1.5, 1.1.6 y 2.0.1. Con la version 1.1.1 da problemas, por ejemplo, por el uso que se hace del tercer parámetro de la función de texto TextReplace(texto, tabla, numero de ciclos), ya que la versión de Tol 1.1.1 no se contemplaba el número de ciclos.

Árbol de ficheros

iForense construye las paginas del sitio sobre informática forense Forense.Info

- ← make.tol proceso principal de generacion del sitio web y todas su páginas
- tol directorios de código en lenguaje Tol
  - cmm funciones comunes
    - ← txt.tol código de funciones de textos
    - ← dte.tol código de funciones de fechas
    - ← set.tol código de funciones de conjuntos
    - ← fil.tol código de funciones de ficheros
    - ← dir.tol código de funciones de directorios
    - ← tme.tol funciones del macro-expansor doble de Tol en Html
    - ← img.tol funciones para imagenes declaradas en Html
    - ← ftp.tol para generar mandatos para hacer Ftp

- ← xsm.tol para construir sitemaps en Xml
- app funciones específicas de aplicación
  - ← glo.tol de términos del glosario forense e informatico
  - ← pdb.tol de manejo de los posts de una agenda
  - ← pht.tol funciones auxiliares para el Html de los post
- ← inc.tol para la inclusión de ficheros en lenguaje Tol
- agenda directorio destinado a la agenda de post
  - ← 01.presentacion.age ejemplo de los primeros posts del sitio Forense.Info
  - ← 20.glosario.txt glosario de ejemplo con algunos términos de informática forense
- web directorio destinado a las páginas web generadas y a contenido
  - css directorio para ficheros de estilo de tipo Css
    - ← common.css fichero Cascading Style Sheets para las paginas Html
  - seed directorio para ficheros semilla
    - ← seed.htm semilla de página Html, template, con Tol embebido
  - src directorio para ficheros con código Javascript
    - ← common.js funciones Javascript de redes sociales y multimedia
  - imagenes directorio de ilustraciones del sitio web
    - ← Perito.cpb.CrackingPasswordsFuerzaBruta.png ejemplo de imagen ilustrativa de Forense.Info
    - ← Perito.pmm.MapaModulo.png ejemplo de esquema ilustrativo de Forense.Info
  - articulos directorio para las páginas de artículos de informática forense
    - → peritojudicial.html ejemplo de página Html de artículo generada
  - categorias directorio para paginas de categorias de informática forense
    - → presentacion.html ejemplo de página Html de categoría generada
  - → sitemap.xml mapa del sitio web generado automáticamente en Xml
- → resultado.html muestra del web de informática forense del dominio Forense.Info
- → iforense.pdf documento resumen de funciones del programa creador del web

Declaraciones

Inclusiones

- `Set allInc`
  Inclusion de las funciones comunes y de aplicacion.

Constantes

- `Text makSep`

Linea horizontal para separar fases de operacion.

- `Text CtrAge`
  All text files inside this directory.
- `Text CtrFil`
  Para el volcado de estadisticas de posts.
- `Real CtrPxC`
  Number of posts per category page.
- `Real CtrPxA`
  Number of posts per article page.
- `Real CtrTit`
  Number of posts per title.
- `Real CtrDes`
  Number of posts per description.
- `Set CatMnu`
  Categorias del menu principal.
- `Set CatAre`
  Areas de conocimiento.
- `Set CatMod`
  Módulos de las areas de conocimiento.
- `Set CatBib`
  Referencias bibliograficas.
- `Set CatCon`
  Datos de contacto y privacidad.
- `Set CatTop`
  Todas las categorias de alto nivel.
- `Set CatNot`
  Not to top category list.
- `Set CatAll`
  Todas las categorias de cualquier nivel.

Proceso

- `Text ctrExe`
  Argumento validado para la ejecucion del make.
- `Real makHlp`
  Es cierto si se ha visualizado la ayuda.

Read agenda

- `Set CtrPdb`
  Read all posts.
- `Set CtrCmn`
  Posts comunes.

- `Set CtrAre`
  Post de areas de conocimiento.
- `Set CtrMod`
  Post de los modulos de las areas de conocimiento.
- `Set CtrBib`
  Post de referencias bibliograficas.
- `Real makCat`
  Build the category pages for CatAll set.
- `Real makArt`
  Build all articles pages.
- `Real makRot`
  Build root absolute page.
- `Real makXsm`
  Build Xml site map.
- `Real makFtp`
  Crear los ficheros de mandatos completos ftp o de actualizacion fup.
- `Real makSnd`
  Send files via ftp.
- `Real makRep`
  Massive file replacement.
- `Real gloSta`
  Ver los terminos más frecuentes del glosario.

Set allInc

```
////////////////////////////////////////////////////////////////////////////
Set  allInc = Include("tol/inc.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Inclusion de las funciones comunes y de aplicacion.", allInc);
////////////////////////////////////////////////////////////////////////////
```

Estructuras de datos

```
// Status:
// A: Anulado, aunque este en la agenda no se publica
// B: Baja, solo se publica en su categoria, no como articulo independiente
// C: Comun, se publica en su categoria y como articulo independiente
// D: Destacado, se publica en su categoria y como articulo independiente y
//    si hay un menu de acceso a articulos importantes se incluye en el.
// Otros make web mios no terminan de implementar esto en toda su dimension
// En este (y probablemente en otros) no funcionaban los anulados porque en
// pdb.tol ponia a->pstSta != "-" en vez de a->pstSta != "A"
////////////////////////////////////////////////////////////////////////////

Struct PdbSt   // Posts database
{
  Set  pstCla, // Conjunto de tipos de post
  Text pstSta, // Status A(nulado), B(ajo) y C(comun, por defecto)
  Text pstCod, // Identificador del post
  Text pstTh1, // Titulo del post en Html, h1, h2
```

```
  Text pstTit, // Titulo del post sin Html
  Date pstDte, // Fecha de ocurrencia
  Text pstAut, // Autor
  Text pstTxt  // Codigo de texto del post en Html+Javascript+Tol
};
```

Constantes

Text makSep

```
////////////////////////////////////////////////////////////////////////////
Text makSep = "\n"+Repeat("_", 72)+"\n";
////////////////////////////////////////////////////////////////////////////
PutDescription("Linea horizontal para separar fases de operacion.", makSep);
////////////////////////////////////////////////////////////////////////////
```

Text CtrAge

```
////////////////////////////////////////////////////////////////////////////
Text CtrAge = "agenda";
////////////////////////////////////////////////////////////////////////////
PutDescription("All text files inside this directory.", CtrAge);
////////////////////////////////////////////////////////////////////////////
```

Text CtrFil

```
////////////////////////////////////////////////////////////////////////////
Text CtrFil = "poststat.csv";
////////////////////////////////////////////////////////////////////////////
PutDescription("Para el volcado de estadisticas de posts.", CtrFil);
////////////////////////////////////////////////////////////////////////////
```

Real CtrPxC

```
////////////////////////////////////////////////////////////////////////////
Real CtrPxC =  20;
////////////////////////////////////////////////////////////////////////////
PutDescription("Number of posts per category page.", CtrPxC);
////////////////////////////////////////////////////////////////////////////
```

Real CtrPxA

```
////////////////////////////////////////////////////////////////////////////
Real CtrPxA =   6;
////////////////////////////////////////////////////////////////////////////
PutDescription("Number of posts per article page.", CtrPxA);
////////////////////////////////////////////////////////////////////////////
```

Real CtrTit

```
////////////////////////////////////////////////////////////////////////////
Real CtrTit =   2;
////////////////////////////////////////////////////////////////////////////
PutDescription("Number of posts per title.", CtrTit);
```

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

Real CtrDes

```
////////////////////////////////////////////////////////////////////////////
Real CtrDes =  20;
////////////////////////////////////////////////////////////////////////////
PutDescription("Number of posts per description.", CtrDes);
////////////////////////////////////////////////////////////////////////////
```

Set CatMnu

```
////////////////////////////////////////////////////////////////////////////
Set CatMnu =
[[
  "Presentación",
  "Derecho e informática",
  "Informática forense",
  "Investigación básica",
  "Investigación avanzada",
  "Base documental"
]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Categorias del menu principal.", CatMnu);
////////////////////////////////////////////////////////////////////////////
```

Set CatAre

```
////////////////////////////////////////////////////////////////////////////
Set CatAre = [["Área"]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Areas de conocimiento.", CatAre);
////////////////////////////////////////////////////////////////////////////
```

Set CatMod

```
////////////////////////////////////////////////////////////////////////////
Set CatMod = [["Módulo"]];        //
////////////////////////////////////////////////////////////////////////////
PutDescription("Módulos de las areas de conocimiento.", CatMod);
////////////////////////////////////////////////////////////////////////////
```

Set CatBib

```
////////////////////////////////////////////////////////////////////////////
Set CatBib = [["Bibliografía"]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Referencias bibliograficas.", CatBib);
////////////////////////////////////////////////////////////////////////////
```

Set CatCon

```
////////////////////////////////////////////////////////////////////////////
Set CatCon = [["Contacto"]];      // Contacto y privacidad
////////////////////////////////////////////////////////////////////////////
PutDescription("Datos de contacto y privacidad.", CatCon);
```

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

Set CatTop

```
////////////////////////////////////////////////////////////////////////////
Set CatTop = CatMnu << CatAre << CatMod << CatBib << CatCon;
////////////////////////////////////////////////////////////////////////////
PutDescription("Todas las categorias de alto nivel.", CatTop);
////////////////////////////////////////////////////////////////////////////
```

Set CatNot

```
////////////////////////////////////////////////////////////////////////////
Set CatNot =
[[
  "Tema",
  "Resumen",
  "Multimedia", "Documentación PDF", "Vídeo MP4",
  "Mapa del web"
]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Not to top category list.", CatNot);
////////////////////////////////////////////////////////////////////////////
```

Set CatAll

```
////////////////////////////////////////////////////////////////////////////
Set CatAll = CatTop << CatNot;
////////////////////////////////////////////////////////////////////////////
PutDescription("Todas las categorias de cualquier nivel.", CatAll);
////////////////////////////////////////////////////////////////////////////
```

Text ctrExe

```
////////////////////////////////////////////////////////////////////////////
Text ctrExe = If(Not(ObjectExist("Text","ctrBat")), "hlp",
              If(ctrBat="",                         "hlp",
                                                    ToLower(ctrBat)));
////////////////////////////////////////////////////////////////////////////
PutDescription("Argumento validado para la ejecucion del make.", ctrExe);
////////////////////////////////////////////////////////////////////////////
```

Real makHlp

```
////////////////////////////////////////////////////////////////////////////
Real makHlp = If(ctrExe!="hlp", FALSE,
{
  Text WriteLn(
  makSep+"help:
  Usage: make [OPTION]
  Builds Forense.Info site
  OPTION
    cat: build all category pages
    art: build all articles pages
    rot: build root absolute page
    xsm: build sm site maps
    ftp: build ftp files (go to ftp dir and run manually)
    fup: build a file update protocol (newer than ftp.log file)
    snd: send file via ftp
```

```
    all: do all works: xml, fup...
    rep: massive file replacement (internal order)
    hlp: show this help
    tst: test some functions");
  TRUE
});
////////////////////////////////////////////////////////////////////////
PutDescription("Es cierto si se ha visualizado la ayuda.", makHlp);
////////////////////////////////////////////////////////////////////////
```

Set CtrPdb

```
////////////////////////////////////////////////////////////////////////
Set CtrPdb = If(! (ctrExe <: [["all", "cat", "art"]]), Empty,
                PdbRead(CtrAge)); // Read all posts
////////////////////////////////////////////////////////////////////////
PutDescription("Read all posts.", CtrPdb);
////////////////////////////////////////////////////////////////////////
```

Set CtrCmn

```
////////////////////////////////////////////////////////////////////////
Set CtrCmn = Select(CtrPdb, Real(Set a) { a->pstSta >= "C" });
////////////////////////////////////////////////////////////////////////
PutDescription("Posts comunes.", CtrCmn);
////////////////////////////////////////////////////////////////////////
Text WriteLn("Status __C "+FormatReal(Card(CtrCmn),"%3.0lf")+" registers");
```

Set CtrAre

```
////////////////////////////////////////////////////////////////////////
Set CtrAre = Select(CtrCmn, Real(Set a) { And(  CatAre[1] <: a->pstCla,
                                              !("Resumen" <: a->pstCla)) });
////////////////////////////////////////////////////////////////////////
PutDescription("Post de areas de conocimiento.", CtrAre);
////////////////////////////////////////////////////////////////////////
```

Set CtrMod

```
////////////////////////////////////////////////////////////////////////
Set CtrMod = Select(CtrCmn, Real(Set a) { CatMod[1] <: a->pstCla });
////////////////////////////////////////////////////////////////////////
PutDescription("Post de los modulos de las areas de conocimiento.", CtrMod);
////////////////////////////////////////////////////////////////////////
```

Set CtrBib

```
////////////////////////////////////////////////////////////////////////
Set CtrBib = Select(CtrCmn, Real(Set a) { CatBib[1] <: a->pstCla });
////////////////////////////////////////////////////////////////////////
PutDescription("Post de referencias bibliograficas.", CtrBib);
////////////////////////////////////////////////////////////////////////
```

Real makCat

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

```
Real makCat = If(And(ctrExe!="all",ctrExe!="cat"), FALSE,
{
  Text WriteLn(makSep+"building all category pages...");
  Text WriteFile(CtrFil, "Posts; Class\n");

  Real CtrArt = FALSE; // Categories, not articles

  // Make category pages
  Set  cicCat = EvalSet(CatAll, Real(Text CtrPag)
  {
    Text filNam = PhtFileName(CtrPag); // Output file
    Text WriteLn("["+CtrPag+"|"+filNam+"]");

    Set  SelPdb = // Selected post from database for this page
    {
      Set selSet = Select(CtrPdb, Real(Set pdbObj) { CtrPag <: pdbObj->pstCla
      Real selCrd = Card(selSet);
      Text txtCrd = FormatReal(selCrd,"%.0lf");
      Text WriteLn("  "+txtCrd+" posts selected");
      Text AppendFile(CtrFil, txtCrd+"; "+CtrPag+"\n");

      selSet
    };

    TmeFile("web/seed/seed.htm", "web/"+filNam)
  });

  // Make special category completo, all the posts
  Real  allCat =
  {
    Text CtrPag = "Completo";
    Text filNam = PhtFileName(CtrPag); // Output file
    Text WriteLn("["+CtrPag+"|"+filNam+"]");
    Set  SelPdb = CtrPdb; // Todos los posts
    TmeFile("web/seed/seed.htm", "web/"+filNam)
  };


  Card(cicCat)+allCat
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build the category pages for CatAll set.", makCat);
////////////////////////////////////////////////////////////////////////////
```

Real makArt

```
////////////////////////////////////////////////////////////////////////////
Real makArt = If(And(ctrExe!="all",ctrExe!="art"), FALSE,
{
  Text WriteLn(makSep+"building all article pages...");

  Real CtrArt = TRUE; // Articles

  // Make article pages
  Set  cicArt = For(1, Card(CtrCmn), Real(Real pstNum)
  {
    Set  SelPdb = SetSubCicle(CtrCmn, pstNum, CtrPxA);
    Text CtrPag = TxtOutside2Tag(CtrCmn[pstNum]->pstTit,"<",">");

    Text filNam = PhtFileName(CtrPag); // Output file
    Text WriteLn("["+CtrPag+"\n  |"+filNam+"]");
    TmeFile("web/seed/seed.htm", "web/"+filNam)
  });

  Card(cicArt)
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build all articles pages.", makArt);
////////////////////////////////////////////////////////////////////////////
```

Real makRot

```
////////////////////////////////////////////////////////////////////////////
Real makRot = If(And(ctrExe!="all",ctrExe!="rot"), FALSE,
{
  Text WriteLn(makSep+"building root absolute page...");

  Text WriteLn("-> creación del index.html, homepage");
  Text inxHtm = ReadFile("web/categorias/presentacion.html");
  Text inxRot = ReplaceTable(inxHtm,
  [[
     [[ "\"../",                              "\""]]
  ]]);
  Real FilWriteIfDiff("web/index.html", inxRot);

  Text WriteLn("-> creación del indice de enlaces absolutos");
  // Save external references
  Text absSav = Replace(inxHtm, " href=\"http", " _XX_=\"_XX_");
  Text absRep = ReplaceTable(absSav,
  [[
     [[ " src=\"../", "  src=\"http://www.forense.info/"]],
     [[" href=\"../", " href=\"http://www.forense.info/"]]
  ]],1);

  // Recall external references
  Text absRec = Replace(absRep, " _XX_=\"_XX_", " href=\"http");
  Real FilWriteIfDiff("web/absoluto.html", absRec);

  Text WriteLn("-> creación de la pagina 404");
  Text errRec = Replace(absRec,
    "Formación en el",
    "La página no existe |");
  Real FilWriteIfDiff("web/error404.html", errRec);

  Text WriteLn("-> protección de directorios con absolutos");
  Real FilWriteIfDiff("web/articulos/index.html",  absRep);
  Real FilWriteIfDiff("web/categorias/index.html", absRep);
  Real FilWriteIfDiff("web/css/index.html",        absRep);
  Real FilWriteIfDiff("web/imagenes/index.html",   absRep);
  Real FilWriteIfDiff("web/practicas/index.html",  absRep);
  Real FilWriteIfDiff("web/src/index.html",        absRep);
  Real FilWriteIfDiff("web/tablas/index.html",     absRep);

  Real FilWriteIfDiff("web/seed/index.htm", absRep); // No en sitemap.xml

  Text WriteLn("-> common directory page, utf-8");
  FilCopy("../common/dir.html", "web/categorias/comxidir.html", TRUE)
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build root absolute page.", makRot);
////////////////////////////////////////////////////////////////////////////
```

Real makXsm

```
////////////////////////////////////////////////////////////////////////////
Real makXsm = If(And(ctrExe!="all",ctrExe!="xsm"), FALSE,
{
  Text WriteLn(makSep+"building xml site map...");
  XsmDir("web/sitemap.xml", "web", "http://www.forense.info/")
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build Xml site map.", makXsm);
////////////////////////////////////////////////////////////////////////////
```

Real makFtp

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

```
Real makFtp = If(!(ctrExe <: [["ftp","fup","all"]]), FALSE,
{
  Text msgTxt = If(ctrExe=="ftp", "ftp (all files)", "fup (update)");
  Text absPth = Replace(GetSourcePath(ctrExe),"/make.tol","");     // Absoluto
  Text locPth = absPth+"/web";                    // Ha de ser una ruta absoluta
  Text webNam = GetFileName(absPth);             // Nombre del directorio actual
  Text dtePth = If(ctrExe=="ftp", "", "ftp/"+webNam+".log");       // Relativo

  Text WriteLn(makSep+webNam+": building "+msgTxt+" from "+locPth+"...");
  FtpAll(
    webNam,                       // Web name
    "www.forense.info",           // Host remoto
    locPth,                       // Directorio local
    dtePth,                       // Fichero señal de fecha de actualizacion
    [[//dir        extension    type    binary or ascii
      ["",         "html",     "html",  FALSE]],  // Ascii Html
      ["",         "ico",      "ico",   TRUE]],   // Binary favicon.ico
      ["",         "xml",      "xml",   FALSE]],  // Ascii site map, rss
      ["",         "js",       "js",    FALSE]],  // Javascript
      ["",         "css",      "css",   FALSE]],  // Css
      ["",         "pdf",      "pdf",   TRUE]],   // Pdf documents
      ["",         "png",      "png",   TRUE]]    // Png solo de css
    ]])
});
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Crear los ficheros de mandatos completos ftp o de actualizacion fup.",
makFtp);
////////////////////////////////////////////////////////////////////////////
```

Real makSnd

```
////////////////////////////////////////////////////////////////////////////
Real makSnd = If(And(ctrExe!="all",ctrExe!="snd"), FALSE,
{
  Text WriteLn(makSep+"sending files using ftp...");
  System("ftp\\iForense.bat")
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Send files via ftp.", makSnd);
////////////////////////////////////////////////////////////////////////////
```

Real makRep

```
////////////////////////////////////////////////////////////////////////////
Real makRep = If(ctrExe!="rep", FALSE,
{
  Text WriteLn(makSep+"massive file replacement...");
  Set  repTab = [[ [["\n<Pst.Sta> C\n",               "\n"]],
                   [["\n<Pst.Typ> post\n",            "\n"]],
                   [["\n<Pst.Aut> Antonio Salmerón\n","\n"]] ]];
  Text WriteLn(Replace(F(repTab),"\n",""));

  Set  filSet = DirExtAll(CtrAge, "age", FALSE, TRUE);
  Set  filCic = EvalSet(filSet, Real(Text filPth)
  {
    Text oldTxt = ReadFile(filPth);
    Text oldPth = Replace(filPth, ".age", ".seg");
    Text WriteLn(filPth+"->"+oldPth);
    Text WriteFile(oldPth, oldTxt);

    Text newTxt = ReplaceTable(oldTxt, repTab, 1); // Only one loop
```

```
    Text WriteFile(filPth, newTxt);

    TRUE
  });
  Card(filCic)
});
////////////////////////////////////////////////////////////////////////
PutDescription("Massive file replacement.", makRep);
////////////////////////////////////////////////////////////////////////
```

Real gloSta

```
////////////////////////////////////////////////////////////////////////
Real gloSta = GloStats(2); // Terminos del glosario con mas de 1 ocurrencias
////////////////////////////////////////////////////////////////////////
PutDescription("Ver los terminos más frecuentes del glosario.", gloSta);
////////////////////////////////////////////////////////////////////////
```

txt.tol de iForense

Text functions.

Declaraciones

Funciones de nombre corto

- `Text Q(Text txtVal)`
  Retorna un texto entre dobles comillas. Equivalente a la funcion Tol Qt().
- `Text W(Text txtVal)`
  Retorna un camino en formato Unix convertido a formato Windows/DOS.
- `Text F(Anything anyVal)`
  Retorna numeros, fechas, textos, conjuntos como un texto de formato simple.

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 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.
- `Text TxtBetween2Tag(Text inpTxt, Text tagIni, Text tagEnd, Real cmpFlg)`
  Retorna un subtexto entre la primera ocurrencia de tagIni y tagEnd. Si tagIni o tagEnd no aparecen retorna la tira vacia. Si cmpFlg es cierto entonces aplica la funcion Compact() al texto que retorna. Por ejemplo: TxtBetween2Tag('a b [[c]] d [[e]] f', '[[', ']]', TRUE) retorna 'c'.
- `Text TxtInside2TagPlus(Text txtInp, Text tagIni, Text tagEnd)`
  Returns all the texts between 2 tags plus theis tags (tagIni and tagEnd) in txtInp. Is diferente that the classic TxtInside2Tag(). For example: <aaa(::)bbb(---)ccc>, <(>, <)> -> <(::)(---)>.
- `Text TxtOutside2Tag(Text txtInp, Text tagIni, Text tagEnd)`
  Returns all the texts outside 2 tags (tagIni and tagEnd) in txt. For example: <aaa(::)bbb(:::)ccc>, <(>, <)> -> <aaabbbccc>.
- `Text TxtOutHtmTag(Text htmTxt)`
  Retorna todos el texto fuera de los tags de html.
- `Text TxtOutHtmScr(Text htmTxt)`
  Retorna todos el texto fuera de los tags de html y de los scripts. Sirve para extraer texto limpio del que sacar palabras clave.

- `Text TxtReplaceSecuence(Text txtInp, Set repTab)`

  Retorna un texto resultado de aplicar la tabla de reemplazamientos repTab al texto de entrada txtIno. Es una version de ReplaceTable(txtInp, repTab, 1) que garantiza la secuencia de los reemplazamientos del primero al ultimo de los reemplazamientos. Es una funcion recursiva. Cada reemplazamiento solo se efectua una vez.

Funciones de nombre corto

Text Q()

```
////////////////////////////////////////////////////////////////////////////
Text Q(Text txtVal) // Text
////////////////////////////////////////////////////////////////////////////
{ Char(34) + txtVal + Char(34) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un texto entre dobles comillas. Equivalente a la funcion Tol Qt().",
Q);
////////////////////////////////////////////////////////////////////////////
```

Text W()

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

Text F()

```
////////////////////////////////////////////////////////////////////////////
Text F(Anything anyVal)
////////////////////////////////////////////////////////////////////////////
{
  Text graVal = Grammar(anyVal);
  Case
  (
    graVal=="Text", anyVal, // Ya es texto

    graVal=="Real", If(EQ(anyVal, Round(anyVal)),
                       FormatReal(anyVal, "%.0lf"),  // Entero sin decimales
                       FormatReal(anyVal, "%.2lf")), // 2 decimales

    graVal=="Date", If(Hour(anyVal),
                       FormatDate(anyVal, "%c%Y/%m/%d %h:%i:%s"), // Tiempo
                       FormatDate(anyVal, "%c%Y/%m/%d")),         // Fecha

    graVal=="Set",
    {
      Real crdSet = Card(anyVal);
      Case(
      EQ(crdSet,0), "[]",
      EQ(crdSet,1), "["+F(anyVal[1])+"]",
      TRUE,         "["+F(anyVal[1])+SetSum(For(2,crdSet,Text(Real setPos)
                  { "|"+F(anyVal[setPos]) }))+"]")
```

```
    },

    TRUE,           "Not basic type"
  )
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna numeros, fechas, textos, conjuntos como un texto de formato simple.",
F);
////////////////////////////////////////////////////////////////////////
```

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

Text TxtBetween2Tag()

```
////////////////////////////////////////////////////////////////////////
Text TxtBetween2Tag(Text inpTxt, // Texto de entrada
                    Text tagIni, // Tag inicial
                    Text tagEnd, // Tag final
                    Real cmpFlg) // Si true aplica Compact()
////////////////////////////////////////////////////////////////////////
{
  Real posIni = TextFind(inpTxt, tagIni);
  Text result = If(LE(posIni,0), "",
  {
    Real lenIni = TextLength(tagIni);
    Real posSub = posIni + lenIni;
    Real posEnd = TextFind(inpTxt, tagEnd, posSub);
    If(LE(posEnd, 0), "", Sub(inpTxt, posSub, posEnd-1))
```

```
  });
  If(cmpFlg, Compact(result), result)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un subtexto entre la primera ocurrencia de tagIni y tagEnd.
Si tagIni o tagEnd no aparecen retorna la tira vacia.
Si cmpFlg es cierto entonces aplica la funcion Compact() al texto que retorna.
Por ejemplo:
  TxtBetween2Tag('a b [[ c ]] d [[ e ]] f', '[[', ']]', TRUE)
  retorna 'c'.",
TxtBetween2Tag);
////////////////////////////////////////////////////////////////////////////
```

Text TxtInside2TagPlus()

```
////////////////////////////////////////////////////////////////////////////
Text TxtInside2TagPlus(Text txtInp, // Input text
                       Text tagIni, // Initial tag
                       Text tagEnd) // End tag)
////////////////////////////////////////////////////////////////////////////
{
  Real posIni = TextFind(txtInp, tagIni);
  If(LE(posIni,0), "",
  {
    Real lenIni = TextLength(tagIni);
    Real posSub = posIni + lenIni;
    Real posEnd = TextFind(txtInp, tagEnd, posSub);
    Real txtLen = TextLength(txtInp);

    If(And(EQ(posIni,1),LE(posEnd,0)), txtInp,
    If(And(GT(posIni,1),LE(posEnd,0)), Sub(txtInp,posIni, txtLen),
       Sub(txtInp,posIni,posEnd+TextLength(tagEnd)-1)+ // Recursion
       TxtInside2TagPlus(Sub(txtInp, posEnd+TextLength(tagEnd), txtLen),
                         tagIni, tagEnd)))
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns all the texts between 2 tags plus theis tags (tagIni and tagEnd)
in txtInp.
Is diferente that the classic TxtInside2Tag().
For example: <aaa(::)bbb(---)ccc>, <(>, <)> -> <(::)(---)>.",
TxtInside2TagPlus);
////////////////////////////////////////////////////////////////////////////
```

Text TxtOutside2Tag()

```
////////////////////////////////////////////////////////////////////////////
Text TxtOutside2Tag(Text txtInp, // Input text
                    Text tagIni, // Initial tag
                    Text tagEnd) // End tag)
////////////////////////////////////////////////////////////////////////////
{
  Set  txtSet = TxtTokenizer(tagIni + tagEnd + txtInp, tagIni);
  Set  txtCic = EvalSet(txtSet, Text(Text txtTok)
  { TxtBetween2Tag(txtTok + tagIni, tagEnd, tagIni, FALSE) });
  SetSum(txtCic)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns all the texts outside 2 tags (tagIni and tagEnd) in txt.
For example: <aaa(::)bbb(:::)ccc>, <(>, <)> -> <aaabbbccc>.",
TxtOutside2Tag);
////////////////////////////////////////////////////////////////////////////
```

Text TxtOutHtmTag()

```
////////////////////////////////////////////////////////////////////////
Text TxtOutHtmTag(Text htmTxt)
////////////////////////////////////////////////////////////////////////
{ TxtOutside2Tag(htmTxt, "<", ">") };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna todos el texto fuera de los tags de html.",
TxtOutHtmTag);
////////////////////////////////////////////////////////////////////////
```

Text TxtOutHtmScr()

```
////////////////////////////////////////////////////////////////////////
Text TxtOutHtmScr(Text htmTxt)
////////////////////////////////////////////////////////////////////////
{ TxtOutHtmTag(TxtOutside2Tag(htmTxt, "<script", "</script>")) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna todos el texto fuera de los tags de html y de los scripts.
Sirve para extraer texto limpio del que sacar palabras clave.",
TxtOutHtmScr);
////////////////////////////////////////////////////////////////////////
```

Text TxtReplaceSecuence()

```
////////////////////////////////////////////////////////////////////////
Text TxtReplaceSecuence(Text txtInp, // Texto de entrada
                        Set  repTab) // Tabla de reemplazamientos
////////////////////////////////////////////////////////////////////////
{
  Real crdTab = Card(repTab); // Numero de cambios a realizar
  If(!crdTab, txtInp, // Nada que hacer
  {
    Text txtNew = Replace(txtInp, repTab[1][1], repTab[1][2]);  // Un cambio
    TxtReplaceSecuence(txtNew, SetLastN(repTab,crdTab-1)) // Resto de cambios
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un texto resultado de aplicar la tabla de reemplazamientos repTab al
texto de entrada txtIno.
Es una version de ReplaceTable(txtInp, repTab, 1) que garantiza la secuencia
de los reemplazamientos del primero al ultimo de los reemplazamientos.
Es una funcion recursiva.
Cada reemplazamiento solo se efectua una vez.",
TxtReplaceSecuence);
////////////////////////////////////////////////////////////////////////
```

dte.tol de iForense

Date functions.

Declaraciones

Funciones

- `Text Dte2Spa(Date dteHor)`
  Returns a date as a spanish text.

Text Dte2Spa()

```
////////////////////////////////////////////////////////////////////////////
Text Dte2Spa(Date dteHor) // Spanish
////////////////////////////////////////////////////////////////////////////
{
  Real yeaNum = Year(dteHor);
  Text yeaTxt = FormatReal(yeaNum, "%.0lf");

  Real mthNum = Month(dteHor);
  Text mthTxt = DteSpanishMonth[mthNum];

  Real dayNum = Day(dteHor);
  Text dayTxt = FormatReal(dayNum, "%.0lf");

  dayTxt+" de "+mthTxt+" de "+yeaTxt
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a date as a spanish text.",
Dte2Spa);
////////////////////////////////////////////////////////////////////////////
```

set.tol de iForense

Set functions.

Declaraciones

Funciones

- `Text Set2Txt(Set valSet, Text iniTxt, Text endTxt, Text sepTxt, Text sepLst, Text txtDet, Text datFmt, Text dteDet, Text dteFmt)`
  Returns a text like a list with all the elements of valSet converted in a text format (elements types: Text, Real or Date). Arguments: - iniTxt initial text, for example: '(', '[[', '', etc. - endTxt end text, for example ')', ']]', '', etc. - sepTxt elements separator, for example '; ', ',', '', etc. - sepLst two last elements separator, for example, ' & ', ' and ', etc., you can specify the same as sepTxt - txtDet text delimiters, for exmple, quotes for TOL, single quote for SQL, nothing, etc. - datFmt real numbers format, for explame, '%.0lf' for integers, if none then uses the default TOL real number format. - dteDet date delimiters, for example, simgle quote for SQL. - dteFmt date format, for example, '%c%Y%m%d', if none then uses the default TOL dates format. Only works with TOL types Text, Real or Date, when find a Set type then works in a recursive way with the same arguments.

- `Set Set2Keyword(Set valSet, Real minChr, Real a2zOrd, Real maxKey)`
  Returns a set with all the elements of valSet converted in a text format, ordered and without repetitions. Remove all word with LE(TextLength(), minChr). Returns the maxKey elements more occurrencies.
- `Text Set2TxtKeyword(Set valSet, Real minChr, Real a2zOrd, Real maxKey)`
  Returns a text list with all the elements of valSet converted in a text format with commas like a keywords list, ordered and without repetitions. Remove all word with LE(TextLength(), minChr). Returns the maxKey elements more occurrencies.
- `Text Set2TxtCommaAmp(Set valSet)`
  Return a text list with all the elements of valSet converted in a text format with commas and & in Html style. For example Set2TxtCommaAmp([['a','b','c','d']]) returns: a, b, c & d -> in html -> a, b, c & d.
- `Set SetFirstN(Set setInp, Real numEle)`
  Retorna un subconjunto de un conjunto con los primeros numEle elementos. Si el conjunto tiene menos de numEle elementos los retorna todos. Si numEle es 0 o negativo retorna el conjunto vacio.
- `Set SetFirstNByFun(Set set, Real num, Code funSor)`
  Returns a subset with the first num elements of a set ordered by funSor. If the set does not have num elements returns the set ordered by funSor.
- `Set SetLastN(Set setInp, Real numEle)`
  Retorna un subconjunto de un conjunto con los ultimos numEle elementos. Si el conjunto tiene menos de numEle elementos los retorna todos.
- `Set SetSubCicle(Set setInp, Real iniPos, Real lenRet)`
  Retorna un subconjunto de un conjunto con los primeros numEle elementos. Si el conjunto tiene menos de numEle elementos los extrae por el principio. Por ejemplo: SetSubCicle([a,b,c,d],3,8) -> [c,d,a,b,c,d,a,b]
- `Anything SetGetRand(Set setInp)`
  Retorna un elemento al azar del conjunto de entrada setInp. Si setInp es Empty retorna FALSE.

Text Set2Txt()

```
////////////////////////////////////////////////////////////////////////
Text Set2Txt(Set  valSet, // Set of elements
             Text iniTxt, // Initial text for list
             Text endTxt, // End text for list
             Text sepTxt, // Element separators
             Text sepLst, // 2 last elements separator
             Text txtDet, // Delimiter for texts
             Text datFmt, // Format for real numbers
             Text dteDet, // Delimiter for dates
             Text dteFmt) // Format for dates
////////////////////////////////////////////////////////////////////////
{
  Real card = Card(valSet);
  Text body =
    If(EQ(card,0), "",
    If(EQ(card,1), F(valSet[1]),
    If(EQ(card,2), F(valSet[1])+sepLst+F(valSet[2]),
```

```
    {
      Set txtVal = For(2,card,Text(Real p)
      {
        If(EQ(p,card),sepLst,sepTxt) + F(valSet[p])
      });
      F(valSet[1]) + BinGroup("+",txtVal)
    }));
  iniTxt+body+endTxt
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a text like a list with all the elements of valSet converted in a
text format (elements types: Text, Real or Date).
Arguments:
 - iniTxt initial text, for example: '(', '[[', '', etc.
 - endTxt end text, for example ')', ']]', '', etc.
 - sepTxt elements separator, for example '; ', ', ', '', etc.
 - sepLst two last elements separator, for example, ' & ', ' and ', etc.,
   you can specify the same as sepTxt
 - txtDet text delimiters, for exmple, quotes for TOL, single quote for
   SQL, nothing, etc.
 - datFmt real numbers format, for explame, '%.0lf' for integers, if none
   then uses the default TOL real number format.
 - dteDet date delimiters, for example, simgle quote for SQL.
 - dteFmt date format, for example, '%c%Y%m%d', if none
   then uses the default TOL dates format.
Only works with TOL types Text, Real or Date, when find a Set type then
works in a recursive way with the same arguments.",
Set2Txt);
////////////////////////////////////////////////////////////////////////////
```

Set Set2Keyword()

```
////////////////////////////////////////////////////////////////////////////
Set Set2Keyword(Set  valSet, // Set of texts
                Real minChr, // Minimum number of chars
                Real a2zOrd, // If true ordered az else by ocurrences
                Real maxKey) // Maximun number of keywords
////////////////////////////////////////////////////////////////////////////
{
  Set cmmWrd =
  [[
     "ambos",
     "desde",
     "dentro",
     "entre",
     "esta", "estas", "esto", "estos",
     "junto",
     "mucho", "muchos",
     "otra", "otras", "otro", "otros",
     "parte",
     "porque",
     "puede", "pueden", "pudiendo",
     "siguiente", "siguientes",
     "sobre",
     "sumamente",
     "también",
     "tanto",
     "zulú"
  ]];

  // Convert all to text
  Text lstTxt = ToLower(Set2Txt(valSet,"", "", " ", " ", "", "", "", ""));

  Text htmNot = TxtOutHtmScr(lstTxt);

  Text lstCls = Compact(ReplaceTable(htmNot, // _ must NOT be changed
    [[ [[Char(34), " "]], [["'",      " "]],
       [[".",      " "]], [[":",      " "]],
       [[";",      " "]], [[",",      " "]],
       [["[",      " "]], [["]",      " "]],
       [["(",      " "]], [[")",      " "]],
```

```
/*      [["<",      " "]], [[">",      " "]],    Sobra */
        [["«",      " "]], [["»",      " "]], // Titulos de libros
        [["¡",      " "]], [["!",      " "]],
        [["¿",      " "]], [["?",      " "]],
        [["/",      " "]], [["|",      " "]],
        [["&amp",   " "]],
        [[" ", " "]],
        [[" none ", " "]],
        [[" the ",  " "]],
        [[" and ",  " "]],
        [[" or ",   " "]],
        [[" & ",    " "]],
        [[" y ",    " "]],
        [[" o ",    " "]]
    ]]);

// Split in words
  Set  setTxt = Tokenizer(lstCls, " ");

// Select word with more than minChr characters that locks like words
  Set  setSel = Select(setTxt, Real(Text wrdTxt)
  {
    Real wrdLen = TextLength(wrdTxt);
    Text iniLet = Sub(wrdTxt, 1,      1);
    Text lstLet = Sub(wrdTxt, wrdLen, wrdLen);
    And(GT(wrdLen, minChr), iniLet >= "a", iniLet <= "z",
                            lstLet >= "a", lstLet <= "z")
  });

// Classify words
  Set  setCla = Classify(setSel, Real(Text a, Text b)
                         { Compare(ToLower(a),ToLower(b)) });

// Make a frecuencies table [ word, number of occurrencies ]
  Set  tabFrq = EvalSet(setCla, Set(Set cla)
                         { [[ cla[1], Card(cla) ]] });

// Sort (with most ocurrencies first)
  Set  srtFrq = Sort(tabFrq, Real(Set a, Set b)
                      { Compare(Real(b[2]), Real(a[2])) });

// Project by word column (the first column)
  Set  keySet = EvalSet(srtFrq, Text(Set a) { Text(a[1]) });

// Minus common words
  Set  keyMin = keySet - cmmWrd;

// Select the first maxKey or Card(keySet) or all if maxKey=0
  Set  keyFst = If(maxKey, SetFirstN(keyMin, maxKey), keyMin);

// Sort if is needed
  If(! a2zOrd, keyFst,
     Sort(keyFst, Real(Text a, Text b)
       { Compare(ToLower(a),ToLower(b)) })) // Order
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set with all the elements of valSet converted in a
text format, ordered and without repetitions.
Remove all word with LE(TextLength(), minChr).
Returns the maxKey elements more occurrencies.",
Set2Keyword);
////////////////////////////////////////////////////////////////////////////
```

Text Set2TxtKeyword()

```
////////////////////////////////////////////////////////////////////////////
Text Set2TxtKeyword(Set  valSet, // Set of texts
                    Real minChr, // Minimum number of chars
                    Real a2zOrd, // If true ordered az else by ocurrences
                    Real maxKey) // Maximun number of keywords
////////////////////////////////////////////////////////////////////////////
```

```
{
  Set2Txt(Set2Keyword(valSet, minChr, a2zOrd, maxKey),
          "", "", ", ", ", ", "", "", "", "")
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a text list with all the elements of valSet converted in a
text format with commas like a keywords list, ordered and without repetitions.
Remove all word with LE(TextLength(), minChr).
Returns the maxKey elements more occurrencies.",
Set2TxtKeyword);
////////////////////////////////////////////////////////////////////////////
```

Text Set2TxtCommaAmp()

```
////////////////////////////////////////////////////////////////////////////
Text Set2TxtCommaAmp(Set  valSet) // Set of elements
////////////////////////////////////////////////////////////////////////////
{ Set2Txt(valSet, "", "", ", ", " &amp; ", "", "", "", "") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Return a text list with all the elements of valSet converted in a
text format with commas and & in Html style.
For example Set2TxtCommaAmp([['a','b','c','d']]) returns:
 a, b, c &amp; d -> in html -> a, b, c & d.",
Set2TxtCommaAmp);
////////////////////////////////////////////////////////////////////////////
```

Set SetFirstN()

```
////////////////////////////////////////////////////////////////////////////
Set SetFirstN(Set  setInp, // Set de entrada
              Real numEle) // Numero de elementos a retornar
////////////////////////////////////////////////////////////////////////////
{
  If(LE(numEle, 0), Empty,
     For(1, Min(Card(setInp), numEle), Anything(Real setPos)
         { setInp[setPos] }))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un subconjunto de un conjunto con los primeros numEle elementos.
Si el conjunto tiene menos de numEle elementos los retorna todos.
Si numEle es 0 o negativo retorna el conjunto vacio.",
SetFirstN);
////////////////////////////////////////////////////////////////////////////
```

Set SetFirstNByFun()

```
////////////////////////////////////////////////////////////////////////////
Set SetFirstNByFun(Set set, Real num, Code funSor)
////////////////////////////////////////////////////////////////////////////
{ SetFirstN(Sort(set, funSor), num) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a subset with the first num elements of a set ordered by funSor.
If the set does not have num elements returns the set ordered by funSor.",
SetFirstNByFun);
////////////////////////////////////////////////////////////////////////////
```

Set SetLastN()

```
////////////////////////////////////////////////////////////////////////////
Set SetLastN(Set  setInp, // Set de entrada
             Real numEle) // Numero de elementos a retornar
////////////////////////////////////////////////////////////////////////////
{
  If(LE(numEle, 0), Empty,
     For(Max(1, 1+Card(setInp)-numEle), Card(setInp), Anything(Real setPos)
         { setInp[setPos] }))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un subconjunto de un conjunto con los ultimos numEle elementos.
Si el conjunto tiene menos de numEle elementos los retorna todos.",
SetLastN);
////////////////////////////////////////////////////////////////////////////
```

Set SetSubCicle()

```
////////////////////////////////////////////////////////////////////////////
Set SetSubCicle(Set  setInp, // Conjunto de elementos
                Real iniPos, // Posicion inicial
                Real lenRet) // Numero de elementos a retornar
////////////////////////////////////////////////////////////////////////////
{
  Real modCic(Real setPos, Real crdSet)
  {
    Real modPos = setPos % crdSet;
    If(LE(modPos, 0), crdSet, modPos)
  };

  Real crdSet = Card(setInp);

  For(0, lenRet-1, Anything(Real setPos)
  {
    Real posCic = modCic(iniPos + setPos, crdSet);
    setInp[posCic]
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un subconjunto de un conjunto con los primeros numEle elementos.
Si el conjunto tiene menos de numEle elementos los extrae por el principio.
Por ejemplo: SetSubCicle([a,b,c,d],3,8) -> [c,d,a,b,c,d,a,b]",
SetSubCicle);
////////////////////////////////////////////////////////////////////////////
```

Anything SetGetRand()

```
////////////////////////////////////////////////////////////////////////////
Anything SetGetRand(Set setInp)
////////////////////////////////////////////////////////////////////////////
{
  Real setCrd = Card(setInp);
  If(LE(setCrd, 0), FALSE,
  {
    Real     rndPos = Min(setCrd, Max(1, Round(Rand(0, setCrd)+0.5)));
    Anything rndVal = setInp[rndPos];
    rndVal
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un elemento al azar del conjunto de entrada setInp.
Si setInp es Empty retorna FALSE.",
SetGetRand);
////////////////////////////////////////////////////////////////////////////
```

fil.tol de iForense

Funciones de ficheros.

Declaraciones

Funciones

- `Real FilWriteIfDiff(Text filPth, Text txtNew)`
  Escribe txtOut en filPth si filPth es un nuevo fichero o si su contenido actual es diferente de txtOut y en otro caso no escribe nada. Retorna true si lo ha escrito y false si no ha sido necesario hacerlo. Evita que los ficheros adquiran una fecha de actualizacion reciente cuando su contenido no ha cambiando realmente, evitando, por ejemplo, retransmisiones por ftp de fichero que realmente no han cambiado.
- `Real FilCheckExtension(Text filPth, Text txtExt, Real casSen)`
  Retorna cierto si el fichero de ruta filPth tiene la extension txtExt. Si casSen es cierto distingue entre mayusculas de minusculas.
- `Real FilCopy(Text oldPth, Text newPth, Real overwrite)`
  Returns TRUE in the file oldNam can be copied over newNam.

Real FilWriteIfDiff()

```
////////////////////////////////////////////////////////////////////////////
Real FilWriteIfDiff(Text filPth, // Fichero de salida
                    Text txtNew) // Texto para escribir
////////////////////////////////////////////////////////////////////////////
{
  If(! FileExist(filPth),        { Text WriteFile(filPth, txtNew); TRUE },
  If(txtNew != ReadFile(filPth), { Text WriteFile(filPth, txtNew); TRUE },
                                   FALSE ))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Escribe txtOut en filPth si filPth es un nuevo fichero o si su contenido
actual es diferente de txtOut y en otro caso no escribe nada.
Retorna true si lo ha escrito y false si no ha sido necesario hacerlo.
Evita que los ficheros adquiran una fecha de actualizacion reciente
cuando su contenido no ha cambiando realmente, evitando, por ejemplo,
retransmisiones por ftp de fichero que realmente no han cambiado.",
FilWriteIfDiff);
////////////////////////////////////////////////////////////////////////////
```

Real FilCheckExtension()

```
////////////////////////////////////////////////////////////////////////////
Real FilCheckExtension(Text filPth, // Ruta del fichero
                       Text txtExt, // Extension para comprobar
                       Real casSen) // Si distingue mayus/minus en extension
////////////////////////////////////////////////////////////////////////////
```

```
{
  Text filExt = If(casSen, GetFileExtension(filPth),
                   ToLower(GetFileExtension(filPth)));
  Text chkExt = If(casSen, txtExt,
                   ToLower(txtExt));
  filExt==chkExt
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna cierto si el fichero de ruta filPth tiene la extension txtExt.
Si casSen es cierto distingue entre mayusculas de minusculas.",
FilCheckExtension);
////////////////////////////////////////////////////////////////////////
```

Real FilCopy()

```
////////////////////////////////////////////////////////////////////////
Real FilCopy(Text oldPth,    // Ruta del fichero origen
             Text newPth,    // Ruta del fichero destino
             Real overwrite) // Si cierto sobre-escribe
////////////////////////////////////////////////////////////////////////
{
  Text cpyTxt = "copy "+oldPth+" "+newPth;
  Text cpyCmd = Replace(cpyTxt, "/", "\\");
  Case(
    Not(FileExist(oldPth)),     FALSE,           // Original doesn't exist
    oldPth==newPth,             FALSE,           // Auto-copy ?
    Not(FileExist(newPth)),     System(cpyCmd),  // Execute copy
    overwrite,                  System(cpyCmd),  // Copy and overwrite
                                FALSE)           // Don't overwrite
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns TRUE in the file oldNam can be copied over newNam.",
FilCopy);
////////////////////////////////////////////////////////////////////////
```

dir.tol de iForense

Directories functions.

Declaraciones

Funciones

- `Set DirExtAll(Text dirPth, Text chkExt, Real toLowe, Real casSen)`
  Returns a set of relative paths of files with the extension chkExt that are inside the directory dirPth and inside all of its directories. If toLowe then all names are changed to lower case. If casSen then are case sensitive in the extension match. Is a version of DirAll() function that only check extensions and not uses TextMatch() that writes warnings at Tol 2.0.1.
- `Text DirReadFiles(Text dirPth, Text chkExt, Text filSep)`
  Returns as a text the contents of all files in dirPth that theirs file names match with file pattern filPat. In this text, the contenst of each file will be separated with filSep.

Set DirExtAll()

```
////////////////////////////////////////////////////////////////////////////
Set DirExtAll(Text dirPth, // Directory path
              Text chkExt, // File extension
              Real toLowe, // If true all paths are changed to lower case
              Real casSen) // If true the extension match is case sensitive
////////////////////////////////////////////////////////////////////////////
{
  If(Not(DirExist(dirPth)), Empty,
  {
    Set  getDir = GetDir(dirPth);
    Set  filSet = getDir[1];
    Set  dirSet = getDir[2];

    Set  filFnd = EvalSet(filSet, Text(Text filNam)
    {
      If(!FilCheckExtension(filNam, chkExt, casSen), "",
         If(toLowe, ToLower(dirPth+"/"+filNam), dirPth+"/"+filNam))
    });

    Set filSel = Select(filFnd, Real(Text filNam) { filNam != "" });

    Set  dirFnd = EvalSet(dirSet, Set(Text subDir)
      { DirExtAll(dirPth+"/"+subDir, chkExt, toLowe, casSen) });

    Real dirCar = Card(dirFnd);

    If(EQ(dirCar,0), filSel,
    If(EQ(dirCar,1), filSel << dirFnd[1],
                     filSel << BinGroup("+", dirFnd)))
  })
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Returns a set of relative paths of files with the extension chkExt that
are inside the directory dirPth and inside all of its directories.
If toLowe then all names are changed to lower case.
If casSen then are case sensitive in the extension match.
Is a version of DirAll() function that only check extensions and not uses
TextMatch() that writes warnings at Tol 2.0.1.",
DirExtAll);
////////////////////////////////////////////////////////////////////////////
```

Text DirReadFiles()

```
////////////////////////////////////////////////////////////////////////////
Text DirReadFiles(Text dirPth, // Directory path
                  Text chkExt, // File extension
                  Text filSep) // File separator
////////////////////////////////////////////////////////////////////////////
{
  Set pthSet = DirExtAll(dirPth, chkExt, FALSE, TRUE);
  Set txtSet = EvalSet(pthSet, Text(Text filPth) { ReadFile(filPth) });
  Set2Txt(txtSet, "", "", filSep, filSep, "", "", "", "")
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns as a text the contents of all files in dirPth that theirs file names
match with file pattern filPat.
In this text, the contenst of each file will be separated with filSep.",
DirReadFiles);
////////////////////////////////////////////////////////////////////////////
```

tme.tol de iForense

Macro-expansor potenciado de Tol para ser inyectado en Html o en otros lenguajes de programacion.

Por defecto utiliza tags que combinan < y { el inicial y el final } y > como Php utiliza <? y ?> y Asp <% y %>.

Esta es la version potenciada que permite inyectar Tol dentro de semillas Html del tipo seed.htm, este codigo Tol embebido se expande en el codigo Html de las agendas de post u otras cosas, que a su vez pueden contener codigo Tol embebido que se puede volver a expandir. Existe una version de este codigo que no admite esta capacidad.

Esta es una version especialmente depurada de este codigo e incluye la modificacion TxtReplaceSecuence(codTxt, repTab) en vez de la clasica ReplaceTable(codTxt, repTab, 1).

Declaraciones

Constantes

- `Text TmeIni`
  Default initial tag. Other tags can be used as <% like Asp or <? like Php.
- `Text TmeEnd`
  Default ending tag. Other tags can be used as %> like Asp or ?> like Php.
- `Text TmeSep`
  For internal use. Lets use Tokenizer() replacing initial tags by TmeSep.
- `Text TmeEmpty`
  All code must return some text. If there are only definitions and replace with nothing can be ended with an empty string of with a TmeEmpty.

Funciones

- `Set TmeGetMacros(Text tagIni, Text tagEnd, Text codTxt)`
  Returns a set with all codes inside the initial en the ending tags.
- `Text TmeFormat(Anything retVal)`
  Intenta retornar un texto a partir de otros tipo, es una funcion equivalente a la funcion de nombre corto de textos F().
- `Set TmeEvalMacros(Set codSet)`
  Returns a set with the text results of all macros. This secuential evaluation does not let to share functions and variables between macros.
- `Set TmeShareMacros(Set codSet, Real codPos)`

Returns a set with the text results of all macros. This recursive evaluation lets to share definitions, functions and variables between macros.

- `Text TmeSubMacros(Text tagIni, Text tagEnd, Text codTxt, Real share)`
  Returns the result of full macro expansion. The original remmark with the old code ReplaceTable(codTxt, repTab, 1) was: Be aware if you uses duplicated macros and assumes something about your order definitions. Now the function TxtReplaceSecuence(codTxt, repTab) make the replacements in a strick order. Inside this code always is called with share=TRUE.
- `Text TmePst(Text codTxt)`
  Easy way to call TmeSubMacros() with the default values and behaviour. Returns the text generated.
- `Text TmeExpandFile(Text tagIni, Text tagEnd, Text inpFil, Text outFil)`
  Returns the result of full macro expansion in inpFil. Update the file remmark from classic seed.htm to the output file name outFil. If outFil is not empty then writes the result in outFil.
- `Real TmeFile(Text inpFil, Text outFil)`
  Easy way to call TmeExpandFile() with the default values and behaviour. Returns true if some text was generated.

Constantes

Text TmeIni

```
////////////////////////////////////////////////////////////////////////////
Text TmeIni = "<"+"{";
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Default initial tag. Other tags can be used as <% like Asp or <? like Php.",
TmeIni);
////////////////////////////////////////////////////////////////////////////
```

Text TmeEnd

```
////////////////////////////////////////////////////////////////////////////
Text TmeEnd = "}"+">";
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Default ending tag. Other tags can be used as %> like Asp or ?> like Php.",
TmeIni);
////////////////////////////////////////////////////////////////////////////
```

Text TmeSep

```
////////////////////////////////////////////////////////////////////////////
Text TmeSep = Char(7);
////////////////////////////////////////////////////////////////////////////
PutDescription(
"For internal use. Lets use Tokenizer() replacing initial tags by TmeSep.",
TmeIni);
////////////////////////////////////////////////////////////////////////////
```

Text TmeEmpty

```
////////////////////////////////////////////////////////////////////////
Text TmeEmpty = "";
////////////////////////////////////////////////////////////////////////
PutDescription(
"All code must return some text. If there are only definitions and replace
with nothing can be ended with an empty string of with a TmeEmpty.",
TmeEmpty);
////////////////////////////////////////////////////////////////////////
```

Set TmeGetMacros()

```
////////////////////////////////////////////////////////////////////////
Set TmeGetMacros(Text tagIni, // Initial tag
                 Text tagEnd, // Ending tag
                 Text codTxt) // Other programing language code
////////////////////////////////////////////////////////////////////////
{
  Text repTxt = Replace(codTxt, tagIni, TmeSep);
  Set  linSet = Tokenizer(repTxt, TmeSep);
  Real lenSet = Card(linSet);
  If(lenSet <= 1, Empty, // There are not macros
  {
    Set For(2, lenSet, Text(Real posLin)
    {
      Text linTxt = linSet[posLin];
      Real posEnd = TextFind(linTxt, tagEnd, 1);
      If(posEnd <= 1, "", // Without ending (0) or not code inside (1)
                      Sub(linTxt, 1,posEnd-1))
    })
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set with all codes inside the initial en the ending tags.",
TmeGetMacros);
////////////////////////////////////////////////////////////////////////
```

Text TmeFormat()

```
////////////////////////////////////////////////////////////////////////
Text TmeFormat(Anything retVal)
////////////////////////////////////////////////////////////////////////
{ F(retVal) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Intenta retornar un texto a partir de otros tipo, es una funcion equivalente
a la funcion de nombre corto de textos F().",
TmeFormat);
////////////////////////////////////////////////////////////////////////
```

Set TmeEvalMacros()

```
////////////////////////////////////////////////////////////////////////
Set TmeEvalMacros(Set codSet) // A set with Tol code that returs text
////////////////////////////////////////////////////////////////////////
{
  EvalSet(codSet, Text(Text codEle)
  {                                // All the evaluations at the same level,
    Anything retVal = Eval(codEle); // can't share the definitions
    TmeFormat(retVal)               // Try to convert to text
```

```
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set with the text results of all macros. This secuential evaluation
does not let to share functions and variables between macros.",
TmeEvalMacros);
////////////////////////////////////////////////////////////////////////
```

Set TmeShareMacros()

```
////////////////////////////////////////////////////////////////////////
Set TmeShareMacros(Set  codSet, // Set of tol codes
                   Real codPos) // Position inside codSet (recursion counter)
////////////////////////////////////////////////////////////////////////
{
  Real lstPos = Card(codSet);
  If(codPos > lstPos, Empty, // Nothing to do
  {                                  // Evaluations at different stack levels,
    Anything retVal = Eval(codSet[codPos]); // can inherit definitions
    Text txtFmt = TmeFormat(retVal);        // Try to convert to text
    [[ txtFmt ]] << TmeShareMacros(codSet, codPos+1) // Recursion
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set with the text results of all macros. This recursive evaluation
lets to share definitions, functions and variables between macros.",
TmeShareMacros);
////////////////////////////////////////////////////////////////////////
```

Text TmeSubMacros()

```
////////////////////////////////////////////////////////////////////////
Text TmeSubMacros(Text tagIni, // Initial tag
                  Text tagEnd, // Ending tag
                  Text codTxt, // Other programing language code
                  Real share)  // If true share definitions between macros
////////////////////////////////////////////////////////////////////////
{
  Set macSet = TmeGetMacros(tagIni, tagEnd, codTxt);
  Set txtSet = If(share, TmeShareMacros(macSet, 1), TmeEvalMacros(macSet));

  Set repTab = For(1, Card(macSet), Set(Real macPos)
               { [[ Text(tagIni+macSet[macPos]+tagEnd), txtSet[macPos] ]] });
  TxtReplaceSecuence(codTxt, repTab) // ReplaceTable(codTxt, repTab, 1)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the result of full macro expansion.
The original remmark with the old code ReplaceTable(codTxt, repTab, 1) was:
Be aware if you uses duplicated macros and assumes something about your order
definitions. Now the function TxtReplaceSecuence(codTxt, repTab) make the
replacements in a strick order.
Inside this code always is called with share=TRUE.",
TmeSubMacros);
////////////////////////////////////////////////////////////////////////
```

Text TmePst()

```
////////////////////////////////////////////////////////////////////////
Text TmePst(Text codTxt) // Other programing language code
////////////////////////////////////////////////////////////////////////
{ TmeSubMacros(TmeIni, TmeEnd, codTxt, TRUE) };
////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Easy way to call TmeSubMacros() with the default values and behaviour.
Returns the text generated.",
TmePst);
////////////////////////////////////////////////////////////////////////////
```

Text TmeExpandFile()

```
////////////////////////////////////////////////////////////////////////////
Text TmeExpandFile(Text tagIni, // Initial tag
                   Text tagEnd, // Ending tag
                   Text inpFil, // Input file
                   Text outFil) // Output file
////////////////////////////////////////////////////////////////////////////
{
  Text codTxt = ReadFile(inpFil);
  Text outTxt = TmeSubMacros(tagIni, tagEnd, codTxt, TRUE); // Sharing

  Text outRep = Replace(outTxt, "// FILE    : seed.htm",
                                "// FILE    : "+GetFileName(outFil));

  Real wriFil = If(outFil != "", FilWriteIfDiff(outFil, outRep), FALSE);
  outTxt
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the result of full macro expansion in inpFil.
Update the file remmark from classic seed.htm to the output file name outFil.
If outFil is not empty then writes the result in outFil.",
TmeExpandFile);
////////////////////////////////////////////////////////////////////////////
```

Real TmeFile()

```
////////////////////////////////////////////////////////////////////////////
Real TmeFile(Text inpFil, // Input file
             Text outFil) // Output file
////////////////////////////////////////////////////////////////////////////
{ If(TmeExpandFile(TmeIni, TmeEnd, inpFil, outFil) != "", TRUE, FALSE) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Easy way to call TmeExpandFile() with the default values and behaviour.
Returns true if some text was generated.",
TmeFile);
////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
Text TmePut(Text codTol) // Tol programing language code
////////////////////////////////////////////////////////////////////////////
{ TmeIni+" "+codTol+" "+TmeEnd };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a Tol code inside Tme brackets.",
TmePut);
////////////////////////////////////////////////////////////////////////////
```

img.tol de iForense

Image functions.

Declaraciones

Constantes

- `Text ImgTag`
  Tag inicial para imagenes.

Funciones

- `Set ImgGetSet(Text htmCod)`
  Retorna el conjunto de imagenes de una pieza de codigo Html.
- `Text ImgNormalize(Text imgCod)`
  Retorna el codigo Html de una imagen normalizado.
- `Text ImgGetSrc(Text imgCod)`
  Retorna de un codigo Html de una imagen la url de la imagen.
- `Text ImgGetAlt(Text imgCod)`
  Retorna de un codigo Html de una imagen el texto alt de la imagen.
- `Text ImgGetTit(Text imgCod)`
  Retorna de un codigo Html de una imagen el texto title de la imagen.
- `Text Img2Htm(Text imgSrc, Text imgAlt, Text imgTit, Text imgCla, Text imgDef)`
  Retorna el codigo Html de una imagen con sus atributos completos rellenando aquellos que faltan.
- `Text Img2Div(Text imgSrc, Text imgAlt, Text imgTit, Text imgLnk, Text imgLst, Text imgCla, Text imgDef)`
  Retorna el codigo Html de una imagen con sus atributos completos rellenando aquellos que faltan y todo dentro de una estructura Html div.

Constantes

Text ImgTag

```
////////////////////////////////////////////////////////////////////////////
Text ImgTag = "<img ";
////////////////////////////////////////////////////////////////////////////
PutDescription("Tag inicial para imagenes.", ImgTag);
////////////////////////////////////////////////////////////////////////////
```

Set ImgGetSet()

```
////////////////////////////////////////////////////////////////////////////
Set ImgGetSet(Text htmCod)
////////////////////////////////////////////////////////////////////////////
```

```
{
  Text imgTxt = Compact(TxtInside2TagPlus(htmCod, ImgTag, ">"));
  If(imgTxt == "", Empty,
  {
    Text imgRep = Replace(imgTxt, "><", ">"+Char(7)+"<");
    Tokenizer(imgRep, Char(7))
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el conjunto de imagenes de una pieza de codigo Html.",
ImgGetSet);
////////////////////////////////////////////////////////////////////////
```

Text ImgNormalize()

```
////////////////////////////////////////////////////////////////////////
Text ImgNormalize(Text imgCod)
////////////////////////////////////////////////////////////////////////
{
  Text imgCmp = Compact(imgCod);
  ReplaceTable(imgCmp,
  [[
    [["= ",      "="]],
    [[" =",      "="]],
    [[". '",     "'"]], // Elimina los puntos finales
    [[Char(34), "'"]]
  ]])
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html de una imagen normalizado.",
ImgNormalize);
////////////////////////////////////////////////////////////////////////
```

Text ImgGetSrc()

```
////////////////////////////////////////////////////////////////////////
Text ImgGetSrc(Text imgCod)
////////////////////////////////////////////////////////////////////////
{ TxtBetween2Tag(ImgNormalize(imgCod), "src='", "'", TRUE) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna de un codigo Html de una imagen la url de la imagen.",
ImgGetSrc);
////////////////////////////////////////////////////////////////////////
```

Text ImgGetAlt()

```
////////////////////////////////////////////////////////////////////////
Text ImgGetAlt(Text imgCod)
////////////////////////////////////////////////////////////////////////
{ TxtBetween2Tag(ImgNormalize(imgCod), "alt='", "'", TRUE) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna de un codigo Html de una imagen el texto alt de la imagen.",
ImgGetAlt);
////////////////////////////////////////////////////////////////////////
```

Text ImgGetTit()

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

```
Text ImgGetTit(Text imgCod)
////////////////////////////////////////////////////////////////////////////
{ TxtBetween2Tag(ImgNormalize(imgCod), "title=", "'", TRUE) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna de un codigo Html de una imagen el texto title de la imagen.",
ImgGetTit);
////////////////////////////////////////////////////////////////////////////
```

Text Img2Htm()

```
////////////////////////////////////////////////////////////////////////////
Text Img2Htm(Text imgSrc, // Image path
             Text imgAlt, // Alternate text, mandatory
             Text imgTit, // Image title
             Text imgCla, // Image class
             Text imgDef) // Para cuando no hay textos
////////////////////////////////////////////////////////////////////////////
{
  Text htmSrc =   " src="+Q(imgSrc);

  Text htmCla = " class="+Q(imgCla);

  Text htmAlt = If(imgAlt!="", " alt="+Q(imgAlt),     // Hay alt
                If(imgTit!="", " alt="+Q(imgTit),     // Sin alt pero si title
                               " alt="+Q(imgDef)));   // Sin nada

  Text htmTit = If(imgTit!="", " title="+Q(imgTit),   // Hay title
                If(imgAlt!="", " title="+Q(imgAlt),   // Sin title pero si alt
                               " title="+Q(imgDef)); // Sin nada

  Text htmImg = "\n  <img" + htmSrc +
                "\n     "   + htmCla +
                "\n       " + htmAlt +
                "\n    "    + htmTit + " />";

  htmImg
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html de una imagen con sus atributos completos rellenando
aquellos que faltan.",
Img2Htm);
////////////////////////////////////////////////////////////////////////////
```

Text Img2Div()

```
////////////////////////////////////////////////////////////////////////////
Text Img2Div(Text imgSrc, // Image path
             Text imgAlt, // Alternate text, mandatory
             Text imgTit, // Image title
             Text imgLnk, // Main link
             Text imgLst, // List of links
             Text imgCla, // Image class
             Text imgDef) // Para cuando no hay textos
////////////////////////////////////////////////////////////////////////////
{
  Text htmImg = Img2Htm(imgSrc, imgAlt, imgTit, imgCla, imgDef);

  Text htmTxt = Case
  (
    And(imgAlt != "", imgTit == ""),             imgAlt,
    And(imgAlt == "", imgTit != ""),             imgTit,
    And(imgAlt == "", imgTit == ""),             imgDef,
        imgAlt == imgTit,                        imgAlt,
        TRUE,                          imgTit+": "+imgAlt
  )+". ";
```

```
  Text htmDes = "<p>"+htmTxt+imgLst+"</p>";

  "\n<div class="+Q(imgCla)+">" +
  "\n  "    + imgLnk + htmImg + "</a>" +
  "\n  "    + htmDes +
  "\n</div><div style="+Q("clear:both")+"></div>\n"
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html de una imagen con sus atributos completos rellenando
aquellos que faltan y todo dentro de una estructura Html div.",
Img2Div);
////////////////////////////////////////////////////////////////////////////
```

ftp.tol de iForense

Funciones para realizar Ftp (versión independiente del web). No realiza el Ftp sino que crea todos los ficheros de mandatos que permiten realizarlo de forma automatica.

Declaraciones

Funciones

- `Real FtpDir(Text hstDir, Text locDir, Text extPat, Text cmdFil, Real toLowe, Real casSen, Text dtePth)`
  Agrega de formar recursiva a un fichero de nombre cmdFil las instrucciones necesarias para enviar todos los ficheros con extension extPat de un directorio local (locDir) y de sus subdirectorios. Hay que proporcionarle el directorio remoto (hstDir) bajo el cual se requiere hacer la copia. Si toLowe es minuscula todos los nombres de ficheros y subdirectorios son enviados en minusculas, esto es clasico para sistemas Unix o Linux. Si casSen es TRUE el math de fichero se realizar sensible a mayusculas y minusculas. Si se le proporciona el nombre de un fichero que indica una fecha dtePth, entonces solo transmite los ficheros mas recientes que dicho fichero, este fichero señal de fecha puede ser el fichero de log del ftp.
- `Real FtpBuild(Text hstDir, Text locDir, Text extPat, Text cmdFil, Text hstNam, Real toLowe, Real casSen, Real binCtr, Text dtePth)`
  Crea un fichero de nombre cmdFil con las instrucciones necesarias para enviar todos los ficheros de extension extPat de un directorio local (locDir) y de sus subdirectorios. Hay que proporcionar el nombre del host y el nombre del directorio remoto (hstDir) bajo el cual se requiere hacer la copia: locDir hstDir / | \ -> / | \ a b c a b c Si toLowe es minuscula todos los nombres de ficheros y subdirectorios son enviados en minusculas, esto es clasico para sistemas Unix o Linux. Si casSen es TRUE el math de fichero se realizar sensible a mayusculas y minusculas. Sin binCtr es TRUE la transmision es en modo binario y en otro caso en Ascii. Si se le proporciona el nombre de un fichero que indica una fecha dtePth, entonces solo transmite los ficheros mas recientes que dicho fichero, este fichero señal de fecha puede ser el fichero de log del ftp. En Windows se ejecuta como ftp -n -s:cmdFil. Retorna el numero de ordenes de transmision dadas.
- `Real FtpAll(Text webNam, Text hstDom, Text locDir, Text dtePth, Set ftpSet)`
  Funcion principal que crea todos los mandatos para la realizacion del ftp. Realiza un ciclo creando ficheros de mandatos para todos los elementos de la tabla ftpSet.

Real FtpDir()

```
////////////////////////////////////////////////////////////////////////
Real FtpDir(Text hstDir, // Directorio remoto
            Text locDir, // Directorio local
            Text extPat, // Patron de extension de fichero
            Text cmdFil, // Fichero de salida con mandatos ftp
            Real toLowe, // Envio de nombre en minusculas
            Real casSen, // Equiparacion sensible a mayus/minusculas
            Text dtePth) // File path to obtain a update date
////////////////////////////////////////////////////////////////////////
```

```
{
  Set  getDir   = GetDir(locDir);
  Set  filSet   = getDir[1];
  Set  dirSet   = getDir[2];

  Set  filSnd  = EvalSet(filSet, Text(Text filNam)
  {
    If(!FilCheckExtension(filNam, extPat, casSen), "", // Does not match
    {
      Real filNew = If(dtePth=="", TRUE, // There is not a file to compare
                       FileTime(locDir+"/"+filNam) >= FileTime(dtePth));
      If(! filNew, "", // Is not a recent update file
      {
        Text filLow = If(toLowe, ToLower(filNam), filNam);
        "send "+filNam+" "+filLow+"\n"
      })
    })
  });

  Text filCmd = If(Card(filSnd)==0, "", BinGroup("+",filSnd));
  Text allCmd = If(filCmd=="", "",
  {
    Text WriteLn("ftp> "+locDir);

    Text AppendFile(cmdFil, "mkdir "+hstDir+"\n");
    Text AppendFile(cmdFil, "cd "+hstDir+"\n");
    Text AppendFile(cmdFil, "lcd "+locDir+"\n");
    Text AppendFile(cmdFil, filCmd)
  });

  Set  dirSnd  = EvalSet(dirSet, Real(Text subDir)
  {
    Text dirName = If(toLowe, ToLower(subDir), subDir);
    FtpDir(hstDir+"/"+dirName, locDir+"/"+dirName, extPat,
           cmdFil, toLowe,  casSen, dtePth)
  });

  Card(filSnd)+Card(dirSnd)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Agrega de formar recursiva a un fichero de nombre cmdFil las instrucciones
necesarias para enviar todos los ficheros con extension extPat de un
directorio local (locDir) y de sus subdirectorios.
Hay que proporcionarle el directorio remoto (hstDir) bajo el cual se requiere
hacer la copia.
Si toLowe es minuscula todos los nombres de ficheros y subdirectorios son
enviados en minusculas, esto es clasico para sistemas Unix o Linux.
Si casSen es TRUE el math de fichero se realizar sensible a mayusculas y
minusculas.
Si se le proporciona el nombre de un fichero que indica una fecha dtePth,
entonces solo transmite los ficheros mas recientes que dicho fichero,
este fichero señal de fecha puede ser el fichero de log del ftp.",
FtpDir);
////////////////////////////////////////////////////////////////////////////
```

Real FtpBuild()

```
////////////////////////////////////////////////////////////////////////////
Real FtpBuild(Text hstDir, // Directorio remoto
              Text locDir, // Directorio local
              Text extPat, // Patron de extension de fichero
              Text cmdFil, // Fichero de salida con mandatos ftp
              Text hstNam, // Nombre del host remoto
              Real toLowe, // Envio de ficheros en minusculas
              Real casSen, // Equiparacion sensible a mayusculas/minusculas
              Real binCtr, // Si TRUE envio en binario, si FALSE en Ascii
              Text dtePth) // Fichero de señal de fecha de actualizacion
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("ftp> call as: ftp -n -s:"+cmdFil);
  Text WriteLn("ftp> searching...");
```

```
  Text WriteFile (cmdFil, "open "+hstNam+"\n");
  Text AppendFile(cmdFil, "user "+FtpHUs+" "+FtpHPa+"\n");
  Text AppendFile(cmdFil, "cd "+hstDir+"\n");
  Text AppendFile(cmdFil, "lcd "+locDir+"\n");

  Text If(binCtr, AppendFile(cmdFil, "binary"+"\n"),
                  AppendFile(cmdFil, "ascii" +"\n"));

  Set  getDir   = GetDir(locDir);
  Set  filSet   = getDir[1];
  Set  dirSet   = getDir[2];

  Set  filSnd  = EvalSet(filSet, Text(Text filNam)
  {
    If(!FilCheckExtension(filNam, extPat, casSen), "", // Doesn't match
    {
      Real filNew = If(dtePth=="", TRUE, // There is not a file to compare
                       FileTime(locDir+"/"+filNam) >= FileTime(dtePth));
      If(! filNew, "", // Is not a recent update file
      {
        Text filLow = If(toLowe, ToLower(filNam), filNam);
        AppendFile(cmdFil,"send "+filNam+" "+filLow+"\n")
      })
    })
  });

  Set  dirSnd  = EvalSet(dirSet, Real(Text subDir)
  {
    Text dirNam = If(toLowe, ToLower(subDir), subDir);
    FtpDir(hstDir+"/"+dirNam, locDir+"/"+dirNam, extPat,
           cmdFil, toLowe,  casSen, dtePth)
  });

  Text AppendFile(cmdFil,"close"+"\n");
  Text AppendFile(cmdFil,"quit"+"\n");
  Text WriteLn("ftp> done");

  Card(filSnd)+Card(dirSnd)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Crea un fichero de nombre cmdFil con las instrucciones necesarias para enviar
todos los ficheros de extension extPat de un directorio local (locDir)
y de sus subdirectorios.
Hay que proporcionar el nombre del host y el nombre del directorio remoto
(hstDir) bajo el cual se requiere hacer la copia:
         locDir          hstDir
       /   |   \  ->  /   |   \
      a    b    c    a    b    c
Si toLowe es minuscula todos los nombres de ficheros y subdirectorios son
enviados en minusculas, esto es clasico para sistemas Unix o Linux.
Si casSen es TRUE el math de fichero se realizar sensible a mayusculas y
minusculas.
Sin binCtr es TRUE la transmision es en modo binario y en otro caso en Ascii.
Si se le proporciona el nombre de un fichero que indica una fecha dtePth,
entonces solo transmite los ficheros mas recientes que dicho fichero, este
fichero señal de fecha puede ser el fichero de log del ftp.
En Windows se ejecuta como ftp -n -s:cmdFil.
Retorna el numero de ordenes de transmision dadas.",
FtpBuild);
////////////////////////////////////////////////////////////////////////////
```

Real FtpAll()

```
////////////////////////////////////////////////////////////////////////////
Real FtpAll(Text webNam, // Web name
            Text hstDom, // Dominio del host remoto
            Text locDir, // Directorio local
            Text dtePth, // Fichero de señal de fecha de actualizacion
            Set  ftpSet) // Tabla de transporte ftp
////////////////////////////////////////////////////////////////////////////
```

```
{
  Text cmdFtp = "ftp/"+webNam+".bat";
  Text WriteFile(cmdFtp, W("cd "+Replace(locDir,"/web","/ftp")+"\n")+
                         "copy "+webNam+".bat "+webNam+".log\n");

  Real ftp(Text dirNam, Text extPat, Text filTyp, Real binCtr)
  {
    Text cmdFil = webNam + filTyp + ".ftp";
    Real bldFtp = FtpBuild(
                    If(dirNam!="", FtpHDi+"/"+dirNam, FtpHDi),
                    If(dirNam!="", locDir+"/"+dirNam, locDir),
                    extPat,
                    "ftp/"+cmdFil,
                    hstDom,                  // Host domain
                    FALSE,                   // To lower all
                    TRUE,                    // Case sensitive
                    binCtr ,                 // binary or Ascii
                    dtePth);                 // File to obtain a update date

    Text AppendFile(cmdFtp, "ftp -n -s:"+cmdFil+" >> "+webNam+".log\n");
    bldFtp
  };

  Set ftpCic = EvalSet(ftpSet, Real(Set ftpReg)
  {
    ftp(ftpReg[1], ftpReg[2], ftpReg[3], ftpReg[4])
  });

  Card(ftpCic)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Funcion principal que crea todos los mandatos para la realizacion del ftp.
Realiza un ciclo creando ficheros de mandatos para todos los elementos de
la tabla ftpSet.",
FtpAll);
////////////////////////////////////////////////////////////////////////////
```

xsm.tol de iForense

Funciones para realizar una descripcion Xsm del sitemap de un sitio web utilizando las especificaciones de Google. Pone la misma prioridad para todas las paginas, ya que la prioridad es un valor relativo entre las paginas del sitio web, no frente a otros webs. Xsm son las siglas de Xsm Site Map, originalmente a las funciones de esta libreria se las identifico por Xml, pero se cambio por ser Xml un termino muy general.

Declaraciones

Variables de control

- `Text XsmPri`
  Prioridad, valor relativo entre paginas.
- `Text XsmFrq`
  Frecuencia de actualizacion del sitio web.
- `Set XsmTyp`
  Tipos de ficheros que se transmiten.
- `Text XsmExc`
  Directorio en el que no se busca, el de semillas.

Constantes

- `Text XsmHea`
  Semilla para la cebecera del sitemap en Xml.
- `Text XsmEnd`
  Texto final de un sitemap, etiqueta de cierre Xml.
- `Text XsmUrl`
  Semilla para la url de un fichero Xml de sitemap.

Funciones

- `Set XsmDateRepTab(Date dteFil)`
  Retorna una tabla de reemplazamientos para las fechas.
- `Real XsmDir(Text xsmFil, Text dirPth, Text urlDom)`
  Crea un sitemap con el contenido de un directorio, retorna el numero de ficheros includos en el sitemap.

Variables de control

Text XsmPri

```
////////////////////////////////////////////////////////////////////////
Text XsmPri = "0.5"; // Da igual 1 que 0 es relativo entre las paginas
////////////////////////////////////////////////////////////////////////
PutDescription("Prioridad, valor relativo entre paginas.", XsmPri);
////////////////////////////////////////////////////////////////////////
```

Text XsmFrq

```
////////////////////////////////////////////////////////////////////////////
Text XsmFrq = "weekly"; // daily, weekly, monthly, ...
////////////////////////////////////////////////////////////////////////////
PutDescription("Frecuencia de actualizacion del sitio web.", XsmFrq);
////////////////////////////////////////////////////////////////////////////
```

Set XsmTyp

```
////////////////////////////////////////////////////////////////////////////
Set  XsmTyp = [["html", "pdf"]]; // Tipos Htm y Pdf
////////////////////////////////////////////////////////////////////////////
PutDescription("Tipos de ficheros que se transmiten.", XsmTyp);
////////////////////////////////////////////////////////////////////////////
```

Text XsmExc

```
////////////////////////////////////////////////////////////////////////////
Text XsmExc = "/practicas/"; // Exclusion
////////////////////////////////////////////////////////////////////////////
PutDescription("Directorio en el que no se busca, el de semillas.", XsmExc);
////////////////////////////////////////////////////////////////////////////
```

Constantes

Text XsmHea

```
////////////////////////////////////////////////////////////////////////////
Text XsmHea =
"<?xml version='1.0' encoding='UTF-8'?>
<urlset xmlns='http://www.google.com/schemas/sitemap/0.84'>
<url>
  <loc>DOM</loc>
  <priority>"+XsmPri+"</priority>
  <lastmod>YEA-MTH-DAYTHOU:MIN:SEC+00:00</lastmod>
  <changefreq>"+XsmFrq+"</changefreq>
</url>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Semilla para la cebecera del sitemap en Xml.", XsmHea);
////////////////////////////////////////////////////////////////////////////
```

Text XsmEnd

```
////////////////////////////////////////////////////////////////////////////
Text XsmEnd = "
</urlset>"; // Parece que no se quiere el último salto de línea
////////////////////////////////////////////////////////////////////////////
PutDescription("Texto final de un sitemap, etiqueta de cierre Xml.", XsmEnd);
////////////////////////////////////////////////////////////////////////////
```

Text XsmUrl

```
////////////////////////////////////////////////////////////////////////////
Text XsmUrl = "
<url>
  <loc>URL</loc>
  <priority>"+XsmPri+"</priority>
  <lastmod>YEA-MTH-DAYTHOU:MIN:SEC+00:00</lastmod>
  <changefreq>"+XsmFrq+"</changefreq>
</url>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Semilla para la url de un fichero Xml de sitemap.", XsmUrl);
////////////////////////////////////////////////////////////////////////////
```

Set XsmDateRepTab()

```
////////////////////////////////////////////////////////////////////////////
Set XsmDateRepTab(Date dteFil) // Fecha con segundo de un fichero
////////////////////////////////////////////////////////////////////////////
{
  //                            123456789.123456789.
  //                            y2005m07d20h17i40s35.00
  Text dteTxt = FormatDate(dteFil, "%cy%Ym%md%d%uh%hi%is%s");
  [[ SetOfText("YEA", Sub(dteTxt,  2,  5)),
     SetOfText("MTH", Sub(dteTxt,  7,  8)),
     SetOfText("DAY", Sub(dteTxt, 10, 11)),
     SetOfText("HOU", Sub(dteTxt, 13, 14)),
     SetOfText("MIN", Sub(dteTxt, 16, 17)),
     SetOfText("SEC", Sub(dteTxt, 19, 20)),
     SetOfText("'",   Char(34)           )
  ]]
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una tabla de reemplazamientos para las fechas.",
XsmDateRepTab);
////////////////////////////////////////////////////////////////////////////
```

Real XsmDir()

```
////////////////////////////////////////////////////////////////////////////
Real XsmDir(Text xsmFil, // Fichero de salida
            Text dirPth, // Directorio a explorar
            Text urlDom) // Dominio con /, ie. http://www.omrforms.es/
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("Output file: "+xsmFil+"\n"+
               "Input path:  "+dirPth+"\n"+
               "Domain:      "+urlDom);

  Set  setDir = EvalSet(XsmTyp, Set(Text filExt) // htm, html, pdf,...
  {
    Text WriteLn("Get files for: "+filExt);
    DirExtAll(dirPth, filExt, TRUE, TRUE)
  });
  Set  getDir = BinGroup("<<", setDir);

  Text WriteLn("Init Xml site map");
  Text WriteFile(xsmFil, ReplaceTable(XsmHea, // Domain and date
                 [[ [["DOM",urlDom]] ]] << XsmDateRepTab(Now)));

  Text WriteLn("Exclusions, not "+XsmExc);
  Set  getSel = Select(getDir, Real(Text pth) { !TextFind(pth, XsmExc) });

  Set  filCic = EvalSet(getSel, Real(Text filPth)
  {
    Text filUrl = Replace(filPth, "web/", urlDom);
    Date filDte = FileTime(filPth);
    Set  repTab = XsmDateRepTab(filDte) <<
```

```
    [[
       [[ "URL", filUrl ]]
    ]];
    Text AppendFile(xsmFil, ReplaceTable(XsmUrl, repTab));
    TRUE
  });

  Text AppendFile(xsmFil, XsmEnd);
  Text WriteLn("Xml site map: "+F(Card(filCic))+" pages");

  Card(filCic)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Crea un sitemap con el contenido de un directorio, retorna el numero de
ficheros includos en el sitemap.",
XsmDir);
////////////////////////////////////////////////////////////////////////////
```

glo.tol de iForense

glossary, vocabulary, functions.

Declaraciones

Constantes

- `Text GloAll`
  Table with all the glossary, terms and their definitions.
- `Text GloLow`
  All the glossary in lower case for test keywords.
- `Set GloMem`
  Set with the memory of all terms writed.
- `Text GloTop`
  Top terms of the glossary. Is a text for use in posts when no other terms are defined.
- `Set GloDef`
  Default terms of the glossary to avoid doubles definitions. For internal use in this library.

Funciones

- `Text GloFind(Text trmTxt)`
  Retorna para un termino el mismo termino y su definicion. La busqueda la realiza de forma exacta con sus mayusculas y minusculas. Retorna la tira vacia si no la encuentra emitiendo un mensaje de error.
- `Real GloTest(Set trmKey)`
  Prueba un conjunto de palabras clave para ver si aparecen como terminos del glosario y visualiza las que pudieran serlo. Retorna el numero de palabras cadidatas.
- `Text GloQuote(Text trmTxt)`
  Retorna para un termino el mismo termino y su definicion como un parrafo Html de tipo blockquote. La busqueda la realiza de forma exacta con sus mayusculas y minusculas. Retorna la tira vacia si no encuentra la definicion.
- `Text GloRand(Text trmLst)`
  Retorna para un termino, elegido al azar de entre una lista de terminos, el mismo termino y su definicion como un parrafo Html de tipo blockquote. La busqueda la realiza de forma exacta con sus mayusculas y minusculas. Retorna la tira vacia si no encuentra la definicion.
- `Real GloStats(Real numMin)`
  Retorna el numero de terminos con igual o mas numMin ocurrencias y como efecto lateral los visualiza.

Constantes

Text GloAll

```
////////////////////////////////////////////////////////////////////////
Text GloAll = ReadFile("../../agenda/20.glosario.all");
////////////////////////////////////////////////////////////////////////
PutDescription(
"Table with all the glossary, terms and their definitions.",
GloAll);
////////////////////////////////////////////////////////////////////////
```

Text GloLow

```
////////////////////////////////////////////////////////////////////////
Text GloLow = ToLower(GloAll);
////////////////////////////////////////////////////////////////////////
PutDescription(
"All the glossary in lower case for test keywords.",
GloLow);
////////////////////////////////////////////////////////////////////////
```

Set GloMem

```
////////////////////////////////////////////////////////////////////////
Set GloMem = Copy(Empty);
////////////////////////////////////////////////////////////////////////
PutDescription(
"Set with the memory of all terms writed.",
GloMem);
////////////////////////////////////////////////////////////////////////
```

Text GloTop

```
////////////////////////////////////////////////////////////////////////
Text GloTop = "En vivo|Informática forense|Escenario forense digital|"+
              "Perito informático|Perito judicial|Perito|Post-mortem|"+
              "Cadena de custodia|Ciencia forense|Prueba digital|Prueba";
////////////////////////////////////////////////////////////////////////
PutDescription(
"Top terms of the glossary. Is a text for use in posts when no other terms
are defined.",
GloTop);
////////////////////////////////////////////////////////////////////////
```

Set GloDef

```
////////////////////////////////////////////////////////////////////////
Set GloDef = Unique(Txt2Set(

// 2 Bibliografía sobre informática forense y pericial
  "Black Hat|CSO|NCJRS|"+

// 4.2 Entidades citadas
  "ACM|ANSI|cmdLabs|ECCouncil|Honeynet Project|IEEE|ISO|RedIRIS|"+
  "SANS Institute|SWGDE|SWGIT|"+

// 1.1 Herramientas citadas
  "Autopsy Forensic Browser|The Sleuth Kit|"+

// 5 Glosario de informática forense y pericial
  "Abrir una cuenta|Acceso|Actualización|Administrador de web|"+
  "Administrador|Alfanumérico|Anónimo|API|Arc|Bloqueadores de escritura|"+
  "Arranque|ASCII|Autenticación|Binario|BIOS|Arquitectura del computador|"+
  "Bluetooth|Bombardeo de correo electrónico|Botnet|Caballo de Troya|"+
```

```
  "Caída del sistema|Ciberespacio|Ciberpunk|Cluster|CMOS|"+
  "Codificación redundante|Codificación|Código corrector|Código detector|"+
  "Código fuente|Compactación de memoria|Confesión judicial|Contraseña|"+
  "Cookie|Copia de seguridad|Correo electrónico spam|Cracker|"+
  "Cuenta de correo electrónico|Cuenta de usuario|Demandado|"+
  "Denegación de servicio|Densidad de grabación|Desbordamiento|Descargar|"+
  "Dirección de correo electrónico|Dominio|Esculpido de fichero|Ética|"+
  "Excepción|Exploit|Fichero comprimido|Firma digital|Firma electrónica|"+
  "Firmware|Formatear|Gusano|Hacker|Hashing|Hiperenlace|"+
  "Hipertexto|Hipótesis|Honeypot|Host|HTML|HTTP|Identificador de usuario|"+
  "Inodo|Insaculación|Interbloqueo|Interdicción|Juramento|Keylogger|"+
  "Login|Logout|Lugar del hallazgo|Mainframe|Máscara|Mensaje|Micronúcleo|"+
  "MIME|Modo privilegiado|Moral|Motor de búsqueda|Navegador|"+
  "Nombre de usuario|Núcleo de un sistema operativo|Página de web|Parche|"+
  "Partición|Periférico|Personalizar|Phishing|Pila|Pirata informático|"+
  "Piratería informática|Pista|Pornografía ilegal|Pornografía infantil|"+
  "Pornografía|Procesador|Proceso|Programa ejecutable|Programa fuente|"+
  "Programa objeto|Programa|Programación|Programador|Promesa|"+
  "Prueba|Prueba testifical|Puerta trasera|Puerto|RAM|Recoger|Red social|"+
  "Residente|Respuesta a incidentes|ROM|Rootkit|Script|SCSI|"+
  "Sector de arranque|Sector|Seguridad de la información|"+
  "Seguridad del sistema|Seguridad del software|Seguridad informática|"+
  "Sistema informático|Software malicioso|SQL|"+
  "Tecnología de la información|Terminal|Transductor|URL|Virus|Volcado|"+
  "Web crawler", "|"));
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Default terms of the glossary to avoid doubles definitions.
For internal use in this library.",
GloDef);
////////////////////////////////////////////////////////////////////////////
```

Text GloFind()

```
////////////////////////////////////////////////////////////////////////////
Text GloFind(Text trmTxt) // Term
////////////////////////////////////////////////////////////////////////////
{
  Text trmTag = "["+trmTxt+"]: ";
  Text trmFnd = TxtBetween2Tag(GloAll, "\n> "+trmTag, "\n", FALSE);
  If(trmFnd != "",
  {
    Text trmRep = ReplaceTable(trmTag+trmFnd, [[ [["[", "<b>" ]],
                                                 [["]", "</b>"]] ]]);

    Text urlTxt = TxtBetween2Tag(trmRep+"|", "| Url: ", "|", FALSE);

    Case(
      urlTxt=="", trmRep, // No hay Urls / no publicar urls de la wikipedia
      TextFind(urlTxt,"wikipedia"), Replace(trmRep, "| Url: "+urlTxt, ""),
      TRUE,
      {
        Set  urlSet = Tokenizer(urlTxt, "*"); // Urls puede haber varias con *
        Set  urlSel = Select(urlSet, Real(Text url)  { Compact(url)!="" });
        Set  urlTab = EvalSet(urlSel, Set(Text url)
        {
          Text urlCmp = Compact(url);
               // Una mayuscula para evitar cambios en urls cuando se parecen
          Text urlLnk = "<a href="+Q(FirstToUpper(urlCmp))+">"+urlCmp+"</a>";
          SetOfText(urlCmp, urlLnk)
        });
/*      Set  srtTab = Sort(urlTab, Real(Set a, Set b)
                      { Compare(TextLength(b[1]),TextLength(a[1])) });
                      // Cambiar las Url más largas las primeras
                      Text WriteLn(F(srtTab));
        TxtReplaceSecuence(trmRep, srtTab) // Convierte Urls en links */
        TxtReplaceSecuence(trmRep, urlTab) // Convierte Urls en links
      })
  },
  {
    Text WriteLn("\nERROR "+trmTag+" not found in the glossary.\n");
    ""
```

```
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna para un termino el mismo termino y su definicion.
La busqueda la realiza de forma exacta con sus mayusculas y minusculas.
Retorna la tira vacia si no la encuentra emitiendo un mensaje de error.",
GloFind);
////////////////////////////////////////////////////////////////////////
```

Real GloTest()

```
////////////////////////////////////////////////////////////////////////
Real GloTest(Set trmKey) // Set of terms from keywords, lowercase all
////////////////////////////////////////////////////////////////////////
{
  Text Write("Glossary: ");
  Set cicTst = EvalSet(trmKey, Real(Text trmTxt)
  {
    Text trmTag = "> ["+trmTxt; // Solo el inicio
    If(!TextFind(GloLow, trmTag), 0,
       { Text Write(FirstToUpper(trmTxt)+"|"); 1 })
  });
  Text WriteLn("\n");
  SetSum(cicTst)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Prueba un conjunto de palabras clave para ver si aparecen como terminos
del glosario y visualiza las que pudieran serlo.
Retorna el numero de palabras cadidatas.",
GloTest);
////////////////////////////////////////////////////////////////////////
```

Text GloQuote()

```
////////////////////////////////////////////////////////////////////////
Text GloQuote(Text trmTxt) // Term
////////////////////////////////////////////////////////////////////////
{
  Text trmDef = GloFind(trmTxt);
  If(trmDef == "", "", // Not found
     "<blockquote><p>"+trmDef+"</p></blockquote>")
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna para un termino el mismo termino y su definicion como un parrafo
Html de tipo blockquote.
La busqueda la realiza de forma exacta con sus mayusculas y minusculas.
Retorna la tira vacia si no encuentra la definicion.",
GloQuote);
////////////////////////////////////////////////////////////////////////
```

Text GloRand()

```
////////////////////////////////////////////////////////////////////////
Text GloRand(Text trmLst) // List of texts separated with |
////////////////////////////////////////////////////////////////////////
{
  Set trmSet = Unique(Txt2Set(trmLst, "|")); // Todos los terminos
  Set trmNew = trmSet - GloMem; // Los no utilizados

  // Lo nuevos si quedan y si no los por defecto
```

```
  Set trmUse = If(Card(trmNew), trmNew,
  {
    Set trmLef = GloDef - GloMem; // Los por defecto no utilizados
    If(Card(trmLef), trmLef, GloDef) // Si se han usado todos -> repite
  });

  Text trmTxt = SetGetRand(trmUse); // Uno al azar
  Set(GloMem := Copy(GloMem << SetOfText(trmTxt)));

  GloQuote(trmTxt)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna para un termino, elegido al azar de entre una lista de terminos,
el mismo termino y su definicion como un parrafo Html de tipo blockquote.
La busqueda la realiza de forma exacta con sus mayusculas y minusculas.
Retorna la tira vacia si no encuentra la definicion.",
GloRand);
////////////////////////////////////////////////////////////////////////
```

Real GloStats()

```
////////////////////////////////////////////////////////////////////////
Real GloStats(Real numMin) // Escribe los terminos con >= numMin ocurrencias
////////////////////////////////////////////////////////////////////////
{
  // Cuenta ocurrencias
  Set  gloCla = Classify(GloMem, Real(Text a, Text b)
       { Compare(a,b) });

  // Selecciona las que tengan igual o mas de numMin ocurrencias
  Set  gloSel = Select(gloCla, Real(Set a)
       { GE(Card(a), numMin) });

  // Ordena de mas a menos ocurrencias
  Set  gloSrt = Sort(gloSel, Real(Set a, Set b)
       { Compare(Card(b), Card(a)) });

  // Visualiza
  Text WriteLn("");
  Set  gloWri = EvalSet(gloSrt, Real(Set a)
  {
    Text WriteLn(FormatReal(Card(a), "%3.0lf")+" "+a[1]);
    TRUE
  });

  Card(gloWri)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el numero de terminos con igual o mas numMin ocurrencias y como
efecto lateral los visualiza.",
GloStats);
////////////////////////////////////////////////////////////////////////
```

pdb.tol de iForense

Posts database functions.

Declaraciones

Constantes

- `Text PdbSep`

  Post separator inside the agendas.

Funciones

- `Set PdbRead(Text inpDir)`

  Reads and returns a post database.

- `Set PdbFirstN(Set inpSet, Real maxNum, Code funSel)`

  Returns the maxNum recents posts for funSel.

Constantes

Text PdbSep

```
////////////////////////////////////////////////////////////////////////////
Text PdbSep = Repeat("_",78);
////////////////////////////////////////////////////////////////////////////
PutDescription("Post separator inside the agendas.", PdbSep);
////////////////////////////////////////////////////////////////////////////
```

Set PdbRead()

```
////////////////////////////////////////////////////////////////////////////
Set PdbRead(Text inpDir)
////////////////////////////////////////////////////////////////////////////
{
  Real err(Text msg) { Text WriteLn("\nERROR: "+msg+"\n"); FALSE }; // Functio

  Text filSep = Char(7);

  Text WriteLn("Reading "+inpDir+"...");
  Text inpAll = DirReadFiles(inpDir, "age", filSep);

  Text inpTxt = Replace(inpAll, PdbSep+"\n",filSep);
  Set  inpSet = Tokenizer(inpTxt,filSep);

  Set  inpTab = EvalSet(inpSet, Set(Text inf)
  {
    // Read
    Set  pstCla = Txt2Set(     TxtBetween2Tag(inf,"<Pst.Cla>", "\n<Pst.", TRUE
    Text pstSta = Sub(         TxtBetween2Tag(inf,"<Pst.Sta>", "\n<Pst.", TRUE
    Text pstCod =              TxtBetween2Tag(inf,"<Pst.Cod>", "\n<Pst.", TRUE
    Text pstTh1 = PhtExpText(  TxtBetween2Tag(inf,"<Pst.Tit>", "\n<Pst.", TRUE
    Text pstTit = TxtOutHtmScr(TxtBetween2Tag(inf,"<Pst.Tit>", "\n<Pst.", TRUE
    Date pstDte = Eval(        TxtBetween2Tag(inf,"<Pst.Dte>", "\n<Pst.", TRUE
    Text pstAut =              TxtBetween2Tag(inf,"<Pst.Aut>", "\n<Pst.", TRUE
    Text pstTxt =              TxtBetween2Tag(inf,"<Pst.Txt>", "\n<Pst.", FALS

    // Check
```

```
    Text chkAut = If(pstAut=="", "Antonio Salmerón", pstAut);

    Set  chkCla = EvalSet(pstCla, Real(Text claNam) // Category class
    { If(claNam <: CatAll, TRUE, err("Class: "+pstCod+" "+claNam)) });

    // Store
    Set  pstObj = PdbSt(pstCla, pstSta, pstCod, pstTh1, pstTit, pstDte, chkAut
//  Text WriteLn(PhtFileName(TxtOutside2Tag(pstObj->pstTit, "<", ">")));
    pstObj
  });

  Set inpSel = Select(inpTab, Real(Set a) { a->pstSta != "A" }); // Not delete

  Set inpSor = Sort(inpSel, Real(Set a, Set b) // By code
              { Compare(a->pstCod, b->pstCod) });

  // Check
  Real chkDup =
  {
    Set codCla = Classify(inpTab, Real(Set a, Set b) // By name
                { Compare(a->pstCod,b->pstCod) });
    Set codDup = EvalSet(codCla, Real(Set cla)
    {
      Real crd = Card(cla);
      Text nam = cla[1]->pstCod;
      If(EQ(crd,1), TRUE, err("Name: "+nam+" "+FormatReal(crd,"%.0lf")+" times
    });

    Set pthCla = Classify(inpTab, Real(Set a, Set b) // By external path / Tit
                { Compare(PhtFileName(TxtOutside2Tag(a->pstTit, "<", ">")),
                          PhtFileName(TxtOutside2Tag(b->pstTit, "<", ">"))) }
    Set pthDup = EvalSet(pthCla, Real(Set cla)
    {
      Real crd = Card(cla);
      Text pth = cla[1]->pstCod;
      If(EQ(crd,1), TRUE, err("Similar titles:\n  ["+cla[1]->pstTit+"]\n  ["+c
    });
    Card(codDup)+Card(pthDup)
  };

  Text WriteLn("Status ABC "+FormatReal(Card(inpTab),"%3.0lf")+" registers");
  Text WriteLn("Status _BC "+FormatReal(Card(inpSel),"%3.0lf")+" registers");

  inpSor
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Reads and returns a post database.",
PdbRead);
////////////////////////////////////////////////////////////////////////////
```

Set PdbFirstN()

```
////////////////////////////////////////////////////////////////////////////
Set PdbFirstN(Set  inpSet, // Post database
              Real maxNum, // Maximum numbers of posts to return
              Code funSel) // Post selection conditions
////////////////////////////////////////////////////////////////////////////
{
  Set  selFst = Select(inpSet, funSel); // Select all that funSel()
  SetFirstN(selFst, maxNum)             // Fst maxNum or all if there are few
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the maxNum recents posts for funSel.",
PdbFirstN);
////////////////////////////////////////////////////////////////////////////
```

pht.tol de iForense

Posts Html functions.

Declaraciones

Funciones

- `Text PhtExpText(Text txtPst)`
  Expande pequeños textos muy habituales dentro de los posts.
- `Text PhtXlsInclude(Text xlsPth)`
  Retorna, para su inclusión, el resultado de leer el Css y el Html de una tabla Excel volcada como Html en el fichero de ruta xlsPth.
- `Text PhtName(Set pstObj)`
  Retorna el nombre del post como su codigo si lo tiene y si no lo tiene utiliza la fecha para crear un nombre.
- `Text PhtAName(Set pstObj)`
  Retorna un enlace con el nombre de un post.
- `Text PhtLinkTag(Text tagLbl, Real tinLbl)`
  Retorna un enlace para una etiqueta que puede ser corta o larga.
- `Text PhtLink(Set pstObj, Real tinLbl)`
  Retorna un enlace para un post recibiendolo como objeto y utilizando una etiqueta que puede ser corta o larga.
- `Text PhtLinkCatSet(Set setCat, Real tinLbl)`
  Retorna una lista de tipo li con los enlaces para todas las categorias de un conjunto.
- `Text PhtLinkPstSet(Set setPst, Real tinLbl)`
  Retorna una lista de tipo li con los enlaces para todos los post de un conjunto.
- `Text PhtFileName(Text tagLbl)`
  Retorna limpia la ruta de un fichero eliminado los acentos y todos los caracteres que no son basicos.
- `Text PhtTitle(Text tagLbl, Real tinLbl)`
  Retorna para ciertas etiquetas otras mas largas o cortas para utilizarse en los diferentes tipos de menus.

Text PhtExpText()

```
////////////////////////////////////////////////////////////////////////
Text PhtExpText(Text txtPst) // Expand basic literals
////////////////////////////////////////////////////////////////////////
{ ReplaceTable(txtPst, PhtRepTab, 1) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Expande pequeños textos muy habituales dentro de los posts.",
PhtExpText);
////////////////////////////////////////////////////////////////////////
```

Text PhtXlsInclude()

```
////////////////////////////////////////////////////////////////////////
Text PhtXlsInclude(Text xlsPth) // Excel.html path
////////////////////////////////////////////////////////////////////////
{
  Text xlsHtm = ReadFile(xlsPth);
  Text iniCss = "<style ";
  Text endCss = "</style>";
  Text xlsCss = iniCss +
                TxtBetween2Tag(xlsHtm, iniCss, endCss, FALSE) +
                endCss;
  Text iniBdy = "<body>";
  Text endBdy = "</body>";
  Text xlsBdy = TxtBetween2Tag(xlsHtm, iniBdy, endBdy, FALSE);

  "\n"+xlsCss+"\n"+xlsBdy+"\n"
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna, para su inclusión, el resultado de leer el Css y el Html de una
tabla Excel volcada como Html en el fichero de ruta xlsPth.",
PhtXlsInclude);
////////////////////////////////////////////////////////////////////////
```

Text PhtName()

```
////////////////////////////////////////////////////////////////////////
Text PhtName(Set pstObj) // If exists returns else build one
////////////////////////////////////////////////////////////////////////
{
  If(pstObj->pstCod!="", pstObj->pstCod,
                         FormatDate(pstObj->pstDte,"%c%Y%m%d%h%m"))
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el nombre del post como su codigo si lo tiene y si no lo tiene
utiliza la fecha para crear un nombre.",
PhtName);
////////////////////////////////////////////////////////////////////////
```

Text PhtAName()

```
////////////////////////////////////////////////////////////////////////
Text PhtAName(Set pstObj)
////////////////////////////////////////////////////////////////////////
{ "<a name='"+PhtName(pstObj)+"'></a>" };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un enlace con el nombre de un post.",
PhtAName);
////////////////////////////////////////////////////////////////////////
```

Text PhtLinkTag()

```
////////////////////////////////////////////////////////////////////////
Text PhtLinkTag(Text tagLbl, // Tag label
                Real tinLbl) // Tiny or long label
////////////////////////////////////////////////////////////////////////
{
  "<a href=" + Q("../"+PhtFileName(tagLbl)) + ">" +
                       PhtTitle   (tagLbl, tinLbl) + "</a>"
```

```
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un enlace para una etiqueta que puede ser corta o larga.",
PhtLinkTag);
////////////////////////////////////////////////////////////////////////
```

Text PhtLink()

```
////////////////////////////////////////////////////////////////////////
Text PhtLink(Set  pstObj, // Post object
             Real tinLbl) // Tiny or long label
////////////////////////////////////////////////////////////////////////
{
  Text pstTit = TxtOutside2Tag(pstObj->pstTit, "<", ">");
  PhtLinkTag(pstTit, tinLbl)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un enlace para un post recibiendolo como objeto y utilizando una
etiqueta que puede ser corta o larga.",
PhtLink);
////////////////////////////////////////////////////////////////////////
```

Text PhtLinkCatSet()

```
////////////////////////////////////////////////////////////////////////
Text PhtLinkCatSet(Set  setCat, // Set of category names
                   Real tinLbl) // Tiny or long label
////////////////////////////////////////////////////////////////////////
{
  SetSum(EvalSet(setCat, Text(Text cat)
                 { "<li>" + PhtLinkTag(cat, tinLbl) + "</li>" }))
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una lista de tipo li con los enlaces para todas las categorias de un
conjunto.",
PhtLinkCatSet);
////////////////////////////////////////////////////////////////////////
```

Text PhtLinkPstSet()

```
////////////////////////////////////////////////////////////////////////
Text PhtLinkPstSet(Set  setPst, // Set of posts
                   Real tinLbl) // Tiny or long label
////////////////////////////////////////////////////////////////////////
{
  SetSum(EvalSet(setPst, Text(Set pst)
                 { "<li>" + PhtLink(pst, tinLbl) + "</li>" }))
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una lista de tipo li con los enlaces para todos los post de un
conjunto.",
PhtLinkPstSet);
////////////////////////////////////////////////////////////////////////
```

Text PhtFileName()

```
////////////////////////////////////////////////////////////////////////
Text PhtFileName(Text tagLbl) // File tag
```

```
////////////////////////////////////////////////////////////////////////////
{
  Text tagLow = ToLower(tagLbl);
  Text tagRep = ReplaceTable(tagLow,
    [[
       [["á", "a"]], [["é", "e"]], [["í", "i"]], [["ó", "o"]], [["ú", "u"]],
       [["ñ", "n"]]
    ]]);

  Set  letCic = For(1, TextLength(tagRep), Text(Real pos)
  {
    Text let = Sub(tagRep, pos, pos);
    If(Or(And(let>="0", let<="9"), And(let>="a", let<="z")), let, "")
  });
  Text tagCls = Set2Txt(letCic, "", "", "", "", "", "", "", "");

  Text dirNam = Case(
    tagLbl == "Completo", "categorias", // Categoria especial
    tagLbl <: CatAll,     "categorias", // Categoria normal
    TRUE,                 "articulos"); // El resto son articulos

  dirNam + "/" + tagCls + ".html"
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna limpia la ruta de un fichero eliminado los acentos y  todos los
caracteres que no son basicos.",
PhtFileName);
////////////////////////////////////////////////////////////////////////////
```

Text PhtTitle()

```
////////////////////////////////////////////////////////////////////////////
Text PhtTitle(Text tagLbl, // For menus
              Real tinLbl) // Tiny or long label
////////////////////////////////////////////////////////////////////////////
{
  Case
  (
    tagLbl == "Presentación",
      If(tinLbl, "Inicio",
         "Presentación del contenido"),

    tagLbl == "Derecho e informática",
      If(tinLbl, "Derecho e<br />informática",
         "Área de conceptos del Derecho para informáticos forenses y peritos")

    tagLbl == "Informática forense",
      If(tinLbl, "Informática<br />forense",
         "Área de conceptos fundamentales de informática forense y pericial"),

    tagLbl == "Investigación básica",
      If(tinLbl, "Investigación<br />básica",
         "Área de investigación en sistemas informáticos"),

    tagLbl == "Investigación avanzada",
      If(tinLbl, "Investigación<br />avanzada",
         "Área de investigación avanzada en informática forense"),

    tagLbl == "Base documental",
      If(tinLbl, "Base<br />documental",
         "Área del caso práctico personal y de la base documental"),

    tagLbl == "Área",
      If(tinLbl, "Áreas",
         "Áreas de conocimiento"),

    tagLbl == "Módulo",
      If(tinLbl, "Módulos",
         "Módulos de informática forense y pericial"),

    tagLbl == "Bibliografía",
```

```
      If(tinLbl, "Bibliografía",
         "Bibliografía seleccionada"),

    TRUE, tagLbl
  )
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna para ciertas etiquetas otras mas largas o cortas para utilizarse
en los diferentes tipos de menus.",
PhtTitle);
////////////////////////////////////////////////////////////////////////////
```

inc.tol de iForense

Inclusion of common functions and application functions

Declaraciones

Inclusiones comunes

- `Set txtInc`
  Text functions.
- `Set dteInc`
  Date functions.
- `Set setInc`
  Set functions.
- `Set filInc`
  File functions.
- `Set dirInc`
  Directory functions.
- `Set tmeInc`
  Macro expansor for Tol inside Html.
- `Set ftpInc`
  Ftp functions.
- `Set xsmInc`
  Xml site maps functions.
- `Set imgInc`
  Image functions for Html code.

Inclusiones de aplicación

- `Set pdbInc`
  Posts database funtions.
- `Set phtInc`
  Posts html functions.
- `Set gloInc`
  Glossary, vocabulary, functions.

Set txtInc

```
////////////////////////////////////////////////////////////////////////////
Set  txtInc = Include("cmm/txt.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Text functions.", txtInc);
////////////////////////////////////////////////////////////////////////////
```

Set dteInc

```
////////////////////////////////////////////////////////////////////////////
Set  dteInc = Include("cmm/dte.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Date functions.", dteInc);
////////////////////////////////////////////////////////////////////////////
```

Set setInc

```
////////////////////////////////////////////////////////////////////////////
Set  setInc = Include("cmm/set.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Set functions.", setInc);
////////////////////////////////////////////////////////////////////////////
```

Set filInc

```
////////////////////////////////////////////////////////////////////////////
Set  filInc = Include("cmm/fil.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("File functions.", filInc);
////////////////////////////////////////////////////////////////////////////
```

Set dirInc

```
////////////////////////////////////////////////////////////////////////////
Set  dirInc = Include("cmm/dir.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Directory functions.", dirInc);
////////////////////////////////////////////////////////////////////////////
```

Set tmeInc

```
////////////////////////////////////////////////////////////////////////////
Set  tmeInc = Include("cmm/tme.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Macro expansor for Tol inside Html.", tmeInc);
////////////////////////////////////////////////////////////////////////////
```

Set ftpInc

```
////////////////////////////////////////////////////////////////////////////
Set  ftpInc = Include("cmm/ftp.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Ftp functions.", ftpInc);
////////////////////////////////////////////////////////////////////////////
```

Set xsmInc

```
////////////////////////////////////////////////////////////////////////////
Set  xsmInc = Include("cmm/xsm.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Xml site maps functions.", xsmInc);
////////////////////////////////////////////////////////////////////////////
```

Set imgInc

```
////////////////////////////////////////////////////////////////////////////
Set  imgInc = Include("cmm/img.tol"); //
////////////////////////////////////////////////////////////////////////////
PutDescription("Image functions for Html code.", imgInc);
////////////////////////////////////////////////////////////////////////////
```

Set pdbInc

```
////////////////////////////////////////////////////////////////////////////
Set  pdbInc = Include("app/pdb.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Posts database funtions.", pdbInc);
////////////////////////////////////////////////////////////////////////////
```

Set phtInc

```
////////////////////////////////////////////////////////////////////////////
Set  phtInc = Include("app/pht.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Posts html functions.", phtInc);
////////////////////////////////////////////////////////////////////////////
```

Set gloInc

```
////////////////////////////////////////////////////////////////////////////
Set  gloInc = Include("app/glo.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Glossary, vocabulary, functions.", gloInc);
////////////////////////////////////////////////////////////////////////////
```