Umentiler

Lee Kindness

COLLABORATORS			
	TITLE : Umentiler		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Lee Kindness	July 28, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Umentiler

1.1 Umentiler />Version Umentiler NONAME NODATE<\

```
   Umentiler :                    Umentiler version />Version Umentiler NONAME  ↩
      NODATE<\
                                  Copyright ©/>DATE LF="%Y"<\ Lee Kindness
 document
     compiler                     Easier document management... or something!


                       Introduction   : What does it do?
                          Usage       : More options...
                          Legal       : Copyright, distribution...
                         History      : Version history
                          Other       : Other programs
                         Contact      : Author contact

---
Compiled               />Date from Umentiler LF="%A %d %B %Y %q:%M:%S"<\
Distribution built     />Date LF="%A %d %B %Y %q:%M:%S"<\
Exe size               />Program NL List Umentiler LFORMAT=%l<\ bytes
Main source size       />Program NL List source/umentiler.c LFORMAT=%l<\ bytes
Version                />Version Umentiler FULL<\
```

1.2 Introduction

```
Right... what does it do? Umentiler simply compiles documents. It replaces
and expands certain tags in the file. For example you could insert the
current time, version of a command, output from a command, another file
into a file.

...But why? Well I am the author of over 28 programs, and a constantly
releasing new versions. When a new version is released I would have to
manually update the version numbers and my (constantly changing) email
addresses... Rather frustrating!

This guide, of course, has been compiled with the aid of Umentiler. The
uncompiled version is called 'Umentiler_uc.guide'. You may wish to compare
```

these documents...

1.3 Usage

```
Invocation  : Launching Umentiler
  Format    : Format of tags in files
```

1.4 Launching Umentiler

Umentiler must be launched from a Shell. It accepts the following argument template:

```
SOURCE/A,DESTINATION,QUIET/S
```

In short the first argument is the file to process and the second is the destination. If you do not pass DESTINATION then SOURCE will be overwritten! For example:

```
>Umentiler textfile textfile.um
```

Will expand all the tags in 'textfile' and output to 'textfile.um'

If you specify QUIET then Umentiler will not print out the number of processed, escaped and invalid tags.

1.5 Tag Format

(If you are using Amigaguide rather than Multiview to view this then the output might not be correct...)

The source file that you supply Umentiler with should be plain ASCII. It can contain 'tags' which will be processed by Umentiler. The format of the tags is:

```
!/>Command options...<\
```

Where 'Command' is one of:

```
Insert
Version
Program
 Date
Setvar
  Var
```

The command and arguments are not case sensitive. All the commands accept arguments in the normal Amiga ReadArgs template form.

The output from the Insert and Program commands is itself parsed for tags. They are then replaced.

To escape a tag, ie if you really want to include it in your text, then
you simply precede it by '!':

```
!!/>This will not be processed<\
```

You can add as many '!'s as you wish, one less is always printed.

NOTE:

If you are putting an Amigaguide tag directly after an escaped Umentiler
tag then you should leave a space between the end of the Umentiler command
and the start of the Amigaguide tag:

```
@{i}!!/>Command options...<\ @{ui}
```

rather than:

```
@{i}!!/>Command options...<\
```

Failing to leave a space will cause the Amigaguide tag to be escaped by the
trailing '\' in the escaped Umentiler tag.

1.6 The INSERT Command

The Insert command is used to insert the contents of a file. It accepts
the following argument template:

```
FILE/A
```

FILE: The path and name of the file to insert, must be given.

For example:

```
!/>Insert S:Startup-Sequence<\
```

would insert your startup-sequence into the file.

1.7 The VERSION Command

The Version command is used to insert version information from a program
or file. It accepts the following argument template:

```
FROM/A,F=FULL/S,NN=NONAME/S,NV=NOVER/S,ND=NODATE/S
```

FROM: The file or program to read the version information from. Must be
    given. NOTE since this command uses the 'Version' AmigaDOS command
    you can also supply additional arguments to 'Version' in it, you
    could for example use "exec.library INTERNAL" for this argument.

FULL: Print out the whole persion string, do not parse further with the
    NONAME, NOVER and NODATE options. Usefull if the version string

```
     is nonstandard.

 NONAME: Specifies not to print the name part of the version information.
     You can use NN for short.

 NOVER: Specifies not to print the version part of the version information.
     You can use NV for short.

 NODATE: Specifies not to insert the date section of the version information.
     You can use ND for short.

 For example:

   !/>Version C:Dir<\

 would produce:

   />Version C:Dir<@{ui}

   !/>Version C:List NONAME ND<\

 would produce:

   />Version C:List NONAME ND<@{ui}

   !/>Version C:Type ND<\

 would produce

   />Version C:Type ND<@{ui}

NOTE:

 If the version string of the program is not 100% in the standard format then
 the results of the NONAME, NOVER and NODATE command options will not be as
 expected! This is not my fault it is the fault of that programs author.
```

1.8 The PROGRAM Command

```
 The Program command is used to insert the output of a command or script.
 By default the commands input stream is NIL:, you can change this by using
 redirection in the CMDLINE. It accepts the following argument template:

   NL=NOLINES/S,CMDLINE/A/F

 NOLINES: Specifies that you want any newline characters generated by the
     command to be stripped. You can use NL as a shortcut.

 CMDLINE: The command line to launch. This argument automatically expands
     to the end of the tag, so if the command line has spaces in it there
     is no need to suround it in quotes. Must be given.

 For example:

   !/>Program List C:d#?<\
```

```
 would use an 8000 byte stack and produce:

   />Program List C:d#?<@{ui}

   C:Dir is !/>Program NOLINES List C:Dir LFORMAT=%l<\ bytes long

 would produce:

   C:Dir is />Program NOLINES List C:Dir LFORMAT=%l<\ bytes long


NOTE:

 If you re-direct the output of the command to NIL: then it has the effect
 of executing a command but not inserting any text.
```

1.9 The DATE Command

```
 The Date command can be used to insert the date or time. It accepts the
 following argument template:

   FROM/K,LF=LOCALEFORMAT/K,F=FORMAT/N/K,NY=NODAY/S,ND=NODATE/S,NT=NOTIME/S

 FROM: Normally the date/time is read in from the system clock. If this
     keyword is specified then they are read in from the specified file.
     Keyword.

 LOCALEFORMAT: This keyword can be used to specify a locale date format
     string. Keyword. Can use FL for short. The format string may contain
     the following formatting codes:

         %a - abbreviated weekday name
         %A - weekday name
         %b - abbreviated month name
         %B - month name
         %c - same as "%a %b %d %H:%M:%S %Y"
         %C - same as "%a %b %e %T %Z %Y"
         %d - day number with leading 0s
         %D - same as "%m/%d/%y"
         %e - day number with leading spaces
         %h - abbreviated month name
         %H - hour using 24-hour style with leading 0s
         %I - hour using 12-hour style with leading 0s
         %j - julian date
         %m - month number with leading 0s
         %M - the number of minutes with leading 0s
         %n - insert a linefeed
         %p - AM or PM strings
         %q - hour using 24-hour style
         %Q - hour using 12-hour style
         %r - same as "%I:%M:%S %p"
         %R - same as "%H:%M"
         %S - number of seconds with leadings 0s
         %t - insert a tab character
```

```
        %T - same as "%H:%M:%S"
        %U - week number, taking Sunday as first day of week
        %w - weekday number
        %W - week number, taking Monday as first day of week
        %x - same as "%m/%d/%y"
        %X - same as "%H:%M:%S"
        %y - year using two digits with leading 0s
        %Y - year using four digits with leading 0s
        %% - percentage sign

    You will require Workbench 2.1 or higher to use this option. If you
    use this option then the FORMAT, NODAY, NODATE and NOTIME options
    are ignored (unless locale.library is unavailable).
    (NOTE: it appears that locale.library has a bug in it... this results
    in the %U and %W formatting codes giving incorrect results)

 FORMAT: Format of the produced date. Keyword. Can use F for short. You can
    use any of the following:

        0 - DOS format, dd-mmm-yy (Default)
        1 - International format, yy-mm-dd
        2 - USA format, mm-dd-yy
        3 - Canadian/British format, dd-mm-yy
        4 - Default locale format

 NODAY: Specifies that the day part of the date string should not be inserted.
    Can use NY for short.

 NODATE: Specifies that the date part of the date string should not be
    inserted. Can use ND for short.

 NOTIME: Specifies that the time part of the date string shouldn't be inserted.
    Can use NT for short.

For example:

   !/>Date LF="%n%A %d %B %Y%n%a %d %b %y%nDay%t%j"<\

would produce:

   />Date LF="%n%A %d %B %Y%n%a %d %b %y%nDay%t%j"<@{ui}

   !/>Date F=0<\
   !/>Date F=1<\
   !/>Date F=2<\
   !/>Date F=3<\
   !/>Date F=4<\

would produce:

   />Date F=0<@{ui}
   />Date F=1<@{ui}
   />Date F=2<@{ui}
   />Date F=3<@{ui}
   />Date F=4<@{ui}

   !/>Date NODATE NOTIME<\
```

```
   !/>Date NODAY NOTIME<\
   !/>Date NODAY NODATE<\

would produce:

   />Date NODATE NOTIME<@{ui}
   />Date NODAY NOTIME<@{ui}
   />Date NODAY NODATE<@{ui}
```

1.10 The SETVAR Command

```
This command never causes any text to be inserted. It allows you to set up
a local environmental variable. This could be useful say if you were wanting
to include a legal document (copyright, disclaimer...) in each of your
program documents. You would use this command to set a variable to the
program name and then in the lagal document you could insert the program
name using the VAR command. This command accepts the following
argument template:

  NAME/A,TEXT/A/F

NAME: The name of the variable to create. This name follows file system
    naming rules. Must be given.

TEXT: What to set the variable to. Expands to the end of the tag, so no
    need to enclose in quotes. Must be given.

For example:

  !/>Setvar PROGRAM Umentiler<\

would setup a variable called 'PROGRAM' and set it to 'Umentiler'

  !/>Setvar NAME Lee Kindness<@{ui}

would set up a local variable called 'NAME' and set it to 'Lee Kindness'
```

1.11 The VAR Command

```
The Var command can be used to insert the contents of local and global
environmental variables. It accepts the following argument template:

  FROM/A,NL=NEWLINE/S,G=GLOBAL/S,L=LOCAL/S

NAME: Name of the variable to insert. Must be given.

NEWLINE: If specified then a newline will be inserted after the variable
    is printed. Can use NL for short.

LOCAL: Specifies only to search for a local variable that matches NAME.
    By default a local variable is searched for first, then a global. Can
    use L for short.
```

```
 GLOBAL: Specifies only to search for a global variable. Can use G for
     short.

 For example:

   Process !/>Var process L<\, RC = !/>Var RC L<\, Secondary RC = !/>Var Result2 L ↩
      <\

 would insert:

   Process />Var process L<\, RC = />Var RC L<\, Secondary RC = />Var Result2 L<@{ ↩
      ui}

   !/>Setvar PROGRAM Umentiler<\!/>Var PROGRAM L<\

 would insert:

   />Setvar PROGRAM Umentiler<\/>Var PROGRAM L<@{ui}
```

1.12 Disclaimer, Distrubution and Copyright

```
/>Insert S:Legal<\
```

1.13 Contact

```
/>Insert S:Contact<\

I hope you find the program useful.

/>Insert S:Signature<\
```

1.14 History

```
/>Insert History<\
```

1.15 Other Programs

```
/>Insert S:Programs<\
```