COMPARESOURCES

By: Bill van Melle (vanMelle.pa@Xerox.com)

Browser added by Ron Kaplan (12/2021)

INTRODUCTION

COMPARESOURCES is a program for comparing two versions of a Lisp source file for differences. The comparison is completely brute-force: COMPARESOURCES reads the complete contents of both files, and compares all the expressions for differences. The files need not be ones produced by MAKEFILE, as COMPARESOURCES reads the contents with READFILE; however, the program is tuned for files of the type produced by MAKEFILE.

HOW TO USE IT

The interface consists of a two functions, COMPARESOURCES and CSBROWSER.

(COMPARESOURCES *FILEX FILEY EXAMINE DW? LISTSTREAM*) [Function]

Compares the files named *FILEX* and *FILEY* for differences. For each type of file object (function, variable, record, etc), COMPARESOURCES identifies which objects of that type differ, and for each such object prints on *LISTSTREAM* a comparison using the function COMPARELISTS. If an object exists on only one of the two files, this fact is noted instead by the message "*name* is not on *file*".

If *DW?* is true, COMPARESOURCES calls DWIMIFY on each function body before performing the comparison. This is useful for comparing a file made with CLISP prettyprinted with one made without.

If *EXAMINE* is true, COMPARESOURCES calls the editor to allow you to more closely examine expressions that differ. Its value is either T, meaning call the editor in all cases, or an atom or a list of atoms chosen from among the following:

OLD	Call the editor for changed objects that are on both files.
NEW	Call the editor for objects that are on only one file.
MISC	Call the editor for changed but otherwise unclassified expressions.
2WINDOWS	Call the editor separately for each pair of changed objects.

In the OLD and MISC cases, the editor is called on a list of two elements, the two expressions. In the NEW case, the editor is called on just the single new expression.

The value returned by COMPARESOURCES is a list whose elements are of the form (type . names), listing the names by type of all objects found to be different. Expressions of no particular type are identified collectively as "(Other --)".

(CSBROWSER *FILEX FILEY DW? LABEL1 LABEL2*) [Function]

This directs the output of a call to COMPARESOURCESto a scrollable window. Clicking on the output for each identified difference brings up for further examination side-by-side SEDIT windows on the

different objects. LABEL1 and LABEL2 are optional strings that override the default way of constructing the title for the broswer window.

FORM OF THE OUTPUT

The output of COMPARESOURCES is in several sections. First, all functions are compared. Then expressions of other types (variables, macros, etc) are compared. When a difference is found, COMPARESOURCES prints the name of the object and calls COMPARELISTS, the same Interlisp function called by COMPARE and COMPAREDEFS. Finally, expressions inside of DECLARE: forms are recursively analyzed in a separate section in the same fashion. All DECLARE: forms of the same applicability (e.g., EVAL@COMPILE DONTCOPY) are handled in the same subsection.

The output of COMPARELISTS takes one of three forms. The usual form is an abbreviated printing of the two expressions with equal elements in the two structures denoted by "&" or "-*n*-" for a subsequence of *n* identical expressions. Identical elements are printed only for purposes of establishing the context of differences. For example,

```
COMPARESOURCES:
(LAMBDA -3- (PROG -16- (for -10- (COND (& (printout & T -4-) -2-)))
(TERPRI --) &))
(LAMBDA -3- (PROG -16- (for -10- (COND (& (printout &   -4-) -2-)))
            &))
```

indicates that in the function `COMPARESOURCES`, an extra argument was added to a `printout` form, and a `(TERPRI --)` expression was added before the final element of the `PROG`. The first 17 elements of the `PROG` form were unchanged, as were the first 11 and last 2 of the `for`.

A more abbreviated form of output occurs when the expressions differ only in a global substitution. In this case, COMPARELISTS prints "`(`*x* `->` *y*`)`" to denote that all occurrences of *x* in the first expression were replaced by *y* in the second expression, and there were no other changes.

Finally, COMPARELISTS prints "`SAME`" if the expressions are "the same". Since COMPARESOURCES only calls COMPARELISTS when the two expressions are not EQUAL, the output `SAME` specifically means that the expressions differ only in the bodies of comments (which COMPARELISTS ignores).

USER EXTENSIONS

COMPARESOURCES already "knows" about several kinds of file package objects, including FNS, VARS, MACROS, RECORDS, and PROPS. Any expression not identifiable as some particular type is compared as a vanilla expression. You can extend the set of types it knows about by adding to the following list:

COMPARESOURCETYPES [Variable]

The elements of this list are lists of the form

```
(TYPE  PREDICATEFN  COMPAREFN  IDFN  DESCRIPTION)
```

as follows:

TYPE — The file package type of the object (or whatever name you wish to give it in the case of fictitious object types).

PREDICATEFN A function of one argument, a single top-level expression as read from the file, that returns true if the expression is of the desired type.

COMPAREFN A function of three arguments, one expression from each file (both guaranteed to have satisfied the PREDICATEFN), and the listing stream. COMPAREFN should compare the two expressions in some appropriate way, printing its results to the listing stream. A typical COMPAREFN calls the function COMPARELISTS on some subform of the expressions. If COMPAREFN is NIL, COMPARELISTS is used.

IDFN A function of one argument, an expression, that returns the "name" of the object described by the expression. Two expressions are assumed to define the same object if their names are EQUAL. The name corresponds roughly to a file package name. For example, for type VARS it is the variable name; for type PROPS it is a pair (atom propname). If IDFN is NIL, CADR is used.

DESCRIPTION A string identifying the kind of object, for use in the comparison printout. If DESCRIPTION is NIL, (L-CASE TYPE T) is used.