NAME

sim – find similarities in C, Java, Pascal, Modula-2, Lisp, Miranda or text files

SYNOPSIS

sim_c [**–[defFinpsST] –r** *N* **–t** *N* **–w** *N* **–o** *F*] file ... [/ [file ...]]
sim_c ...
sim_java ...
sim_pasc ...
sim_m2 ...
sim_lisp ...
sim_mira ...
sim_text ...

DESCRIPTION

Sim_c reads the C files *file* ... and looks for pieces of text that are similar; two pieces of program text are similar if they only differ in layout, comment, identifiers and the contents of numbers, strings and characters. If any runs of sufficient length are found, they are reported on standard output; the number of significant tokens in the run is given between square brackets.

Sim_java does the same for Java, *sim_pasc* for Pascal, *sim_m2* for Modula-2, *sim_lisp* for Lisp, and *sim_mira* for Miranda. *Sim_text* works on arbitrary text; it is occasionally useful on shell scripts.

The program can be used for finding copied pieces of code in purportedly unrelated programs (with **–s** or **–S**), or for finding accidentally duplicated code in larger projects (with **–f**).

If a / is present between the input files, the latter are divided into a group of "new" files (before the /) and a group of "old" files; if there is no /, all files are "new". Old files are never compared to each other. Since the similarity tester reads the files several times, it cannot read from standard input. (See, however, the **–i** option.)

There are the following options:

–d The output is in a diff(1)-like format instead of the default 2-column format.

–e Each file is compared to each file in isolation; this will find all similarities between all texts involved, regardless of duplicates.

–f Runs are restricted to pieces with balancing parentheses, to isolate potential functions (C, Java, Pascal, Modula-2 and Lisp only).

–F The names of functions in calls are required to match exactly (C, Java, Pascal, Modula-2 and Lisp only).

–i The names of the files to be compared are read from standard input, including a possible / ; the file names need to be separated by layout. This allows a very large number of file names to be specified; it differs from the @ facility provided by some compilers in that it handles file names only, and does not recognize option arguments.

–n Similarities found are only summarized, not displayed.

–o F The output is written to the file named *F*.

–p The output is given in similarity percentages; see below.

–r N The minimum run length is set to *N* (default is *N* = 24).

–s The contents of a file are not compared to itself (–s for "not self").

–S The contents of the new files are compared to the old files only – not between themselves.

–t N In combination with the **–p** option, sets the threshold (in percents) below which similarities will not be reported.

–T A more terse and uniform form of output is produced, which may be more suitable for post-processing.

−w N The page width used is set to *N* columns (default is *N* = 80).

The **−p** option results in lines of the form `F consists for x % of G material` meaning that `x` % of `F`'s text can also be found in `G`. Note that this relation is not symmetric; it is in fact quite possible for one file to consist for 100 % of text from another file, while the other file consists for only 1 % of text of the first file, if their lengths differ enough. A threshold can be set using the **−T** option. Note also that the granularity of the recognized text is still governed by the **−r** option or its default.

Care has been taken to keep all internal processes linear in the length of the input, with the exception of the matching process which is almost linear, using a hash table; various other tables are used for speed-up. If, however, there is not enough memory for the tables, they are discarded in order of unimportance, under which conditions the algorithms revert to their quadratic nature.

AUTHOR

Dick Grune, Vrije Universiteit, Amsterdam.

BUGS

Strong periodicity in the input text (like a table of *N* almost identical lines) causes problems. *Sim* tries to cope with this but cannot avoid giving appr. *log N* messages about it. The best advice is still to take the offending files out of the game.

Since it uses *lex(1)* on some systems, it may dump core on any weird construction that overflows *lex*'s internal buffers.