**Gentran**

Original Authors Barbara Gates and Paul Wang

Gentran is a powerful generator of foreign language code. Currently it can generate FORTRAN, 'C', and RATFOR code from Maxima language code. Gentran can translate mathematical expressions, iteration loops, conditional branching statements, data type information, function definitions, matrtices and arrays, and more.

**Main gentran functions (special forms):**

**gentran**(*stmtl, stmt2, ... , stmtn , [fl, f2, ... , fm]*)

Translates each stmt into formatted code in the target language. A substantial subset of expressions and statements in the Maxima programming language can be translated directly into numerical code. The **gentran** command translates Maxima statements or procedure definitions into code in the target language (**gentranlang:** fortran, c, or ratfor). Expressions may optionally be given to Maxima for evaluation prior to translation.

*stmtl, stmt2, ... , stmtn* is a sequence of one or more statements, each of which is any Maxima user level expression, (simple, group, or block) statement, or procedure definition that can be translated into the target language.

*[fl, f2, ... , fm]* is an optional list of output files to which translated output will be written. They can be any of the following:

***string*** = the name of an output file in quotes

**true** (no quotes) = the terminal

**false** = the current output file(s)

**all** = all files currently open for output by gentran

If the files are not open they will be opened; if they are open, output will be appended to them. Filenames are given as quoted strings. If the optional variable **genoutpath** (string, including the final /) default **false** is set, it will be prepended to the output file names. If the output file list is omitted, output will be written to the current output, generally the terminal. **gentran** returns (a list of) the name(s) of file(s) to which code was written.

**gentranout**(*fl, f2, ... , fn*)

Gentran maintains a list of files currently open for output by gentran commands only. gentranout inserts each file name represented by *fl, f2,... , fn* into that list and opens each one for output. It also resets the current output file(s) to include all files in *fl, fl, f2, ... , fn*. gentranout returns the list of files represented by *fl, f2, ... , fn*; i.e., the current output file(s) after the command has been executed.

**gentranshut**(*fl, f2, ... , fn*)

gentranshut creates a list of file names from *fl, f2, ... , fn*, deletes each from the output file list, and closes the corresponding files. If (all of) the current output file(s) are closed, then the current output file is reset to the terminal. gentranshut returns (a list of) the current output file(s) after the command has been executed. **gentranshut**(**all**) will close all gentran output files.

**gentranpush**(*f1, f2, ... , fn*)

gentranpush pushes the file list onto the output stack. Each file in the list that is not already open for output is opened at this time. The current output file is reset to this new element on the top of the stack.

**gentranpop**(*fl, f2, ... , fn*)

gentranpop deletes the top-most occurrence of the single element containing the file name(s) represented by *fl, f2, ... , fn* from the output stack. Files whose names have been completely removed from the output stack are closed. The current output file is reset to the (new) element on the top of the output stack. gentranpop returns the current output file(s) after this command has been executed.

**gentranin**(*fl, f2, ... , fn*, *[fl,f2, ... , fm]*)

gentranin processes mixed-language template files consisting of active parts (delimited by <<…>>) containing Maxima statements, including calls to gentran, and passive parts, assumed to contain statements in the target language (including comments), which are transcribed verbatim. Input files are processed sequentially and the results appended to the output. The presence of >> in passive parts of the file (except for in comments) is interpreted as an end-of-file and terminates processing of that file. The optional list of output files *[fl,f2, ... , fm]* each receive a copy of the entire output. All filespecs are quoted strings. Input files may be given as (quoted string) filenames, which will be located by Maxima **file\_search**. The optional variable **geninpath** (default **false** ) must be a *list* of quoted strings describing the paths to be searched for the input files. If it is set, that list replaces the standard Maxima search paths.

Active parts may contain any number of Maxima expressions and statements. They are not copied directly to the output. Instead, they are given to Maxima for evaluation. All output generated by each evaluation is sent to the output file(s). Returned values are only printed on the terminal. Active parts will most likely contain calls to gentran to generate code. This means that the result of processing a template file will be the original template file with all active parts replaced by generated code. If *[f1, f2, ... , fm]* is not supplied, then generated code is simply written to the current output file(s). However, if it is given, then the current output file is temporarily overridden. Generated code is written to each file represented by *fl,f2, ... , fn* for this command only. Files which were open prior to the call to gentranin will remain open after the call, and files which did not exist prior to the call will be created, opened, written to, and closed. The output file stack will be exactly the same both before and after the call. gentranin returns (to the terminal) the name(s) of (all) file(s) written to by this command.

**gentraninshut**()

A cleanup function to close input files in case where gentranin hung due to error in template.

**tempvar**(*type*)

Generates temporary variable names by concatenating **tempvarname** (default **‘t**) with sequence numbers. If *type* is non-false, *e.g.* “real\*8” the corresponding type is assigned to the variable in the gentran symbol table, which may be used to generate declarations depending on the setting of the **gendecs** flag. It is the users responsibility to make sure temporary variable names do not conflict with the main program.

**markvar**(*vname*)

markvar "marks" variable name *vname* to indicate that it currently holds a significant value.

**unmarkvar**(*vname*)

unmarkvar "unmarks" variable name *vname* to indicate that it no longer holds a significant value.

**markedvarp**(*vname*)

markedvarp tests whether the variable name *vname* is currently marked.

**gendecs**(*name*)

The gendecs command can be called any time the gendecs flag is switched off to retrieve all type declarations from Gentran's symbol table for the given subprogram name (or the "current" subprogram if false is given as its argument).

**gentran\_on**(*sw*)

Turns on the mode switch *sw*.

**gentran\_off**(*sw*)

Turns the given switch, *sw*, off.

**Mode switches:**

fortran default: off

ratfor default: off

c default: off

These mode switches change the default mode of Maxima from evaluation to translation. They can be turned on and off with the gentran commands gentran\_on and gentran\_off. Each time a new Maxima session is started up, the system is in evaluation mode. It prints a prompt on the user's terminal screen and waits for an expression or statement to be entered. It then proceeds to evaluate the expression, prints a new prompt, and waits for the user to enter another expression or statement. This mode can be changed to translation mode by turning on either the fortran, ratfor or c switches. After one of these switches is turned on and until it is turned off, every expression or statement entered by the user is translated into the corresponding language just as if it had been given as an argument to the gentran command. Each of the special functions that can be used from within a call to gentran can be used at the top level until the switch is turned off.

**gendecs** default: on

When the gendecs switch is turned on, gentran generates type declarations whenever possible. When gendecs is switched off, type declarations are not generated. Instead, type information is stored in gentran's symbol table but is not retrieved in the form of declarations unless and until either the gendecs command is called or the gendecs flag is switched back. **Note**: Generated declarations may often be placed in an inappropriate place (*e.g.* in the middle of executable fortran code). Therefore the gendecs flag is turned off during processing of templates by **gentranin**.

**Option Variables:**

**gentranlang** *default*: fortran

Selects the target numerical language. Currently, gentranlang must be fortran, ratfor, or c. Note that symbols entered in Maxima are case-sensitive. gentranlang should not be set to FORTRAN, RATFOR or C.

**fortlinelen** *default*: 72

Maximum number of characters printed on each line of generated FORTRAN code.

**minfortlinelen** *default*: 40

Minimum number of characters printed on each line of generated FORTRAN code.

**fortcurrind** *default*:0

Number of blank spaces printed at the beginning of each line of generated FORTRAN code (after column 6).

**ratlinelen** *default*: 80

Maximum number of characters printed on each line of generated Ratfor code.

**clinelen** *default*: 80

Maximum number of characters printed on each line of generated 'C' code.

**minclinelen** *default*: 40

Minimum number of characters printed on each line of generated 'C' code.

**ccurind** *default*: 0

Number of blank spaces printed at the beginning of each line of generated'C' code.

**tablen** *default*: 4

Number of blank spaces printed for each new level of indentation. (Automatic indentation can be turned off by setting this variable to 0.)

**genfloat** *default*: **false**

When set to true (or any non-false value), causes integers in generated numerical code to be converted to floating point numbers, except in the following places: array subscripts, exponents, and initial, final, and step values in do-loops. An exception (for compatibility with Macsyma 2.4) is that numbers in exponentials (with base %e only) are double-floated even when genfloat is false.

**dblfloat** *default*: **false** If dblfloat is set to true, floating point numbers in gentran output in implementations (such asWindows Maxima under CLISP) in which float and double-float are the same will be printed as \*.d0. In implementations in which float and double-float are different, floats will be coerced to double-float before being printed.

**gentranseg** *default*: **true**

**maxexpprintlen** *default*: 800

When **gentranseg** is true (or any non-false value), causes Gentran to "segment" large expressions into subexpressions of manageable size. The segmentation facility generates a sequence of assignment statements, each of which assigns a subexpression to an automatically generated temporary variable name. This sequence is generated in such a way that temporary variables are re-used as soon as possible, thereby keeping the number of automatically generated variables to a minimum. The maximum allowable expression size can be controlled by setting the **maxexpprintlen** variable to the maximum number of characters allowed in an expression printed in the target numerical language (excluding spaces and other whitespace characters automatically printed by the formatter). When the segmentation routine generates temporary variables, it places type declarations in the symbol table for those variables if possible. It uses the following rules to determine their type:

1. If the type of the variable to which the large expression is being assigned is already known (i.e., has been declared by the user via a TYPE form), then the temporary variables will be declared to be of that same type. 2. If the global variable **tempvartype** has a non-false value, then the temporary variables are declared to be of that type. 3. Otherwise, the variables are not declared unless **implicit** has been set to **true**.

**gentranopt** *default*: **false**

When set to true (or any non-false value), causes Gentran to replace each block of straightline code by an optimized sequence of assignments obtained from the Maxima optimize command. (The optimize command takes an expression and replaces common subexpressions by temporary variable names. It returns the resulting assignment statement, preceded by common-subexpression-to-temporary-variable assignments.

**tempvarname** *default*: **‘t**

Name used as the prefix when generating temporary variable names.

**optimvarname** (*default*: **‘u**) is the preface used to generate temporary file names produced by the optimizer when **gentranopt** is **true**. When both gentranseg and gentranopt are true, the optimizer generates temporary file names using **optimvarname** while the segmentation routine uses **tempvarname** preventing conflict.

**tempvarnum** *default*: 0

Number appended onto tempvarname to create a temporary variable name. If the temporary variable name resulting from appending tempvarnum onto the end of tempvarname has already been generated and still holds a useful value or has a different type than requested, then the number is incremented until one is found that was not previously generated or does not still hold a significant value or a different type.

**tempvartype** *default*: **false**

Target language variable type (e.g., INTEGER, REAL•8, FLOAT, etc.) used as a default for automatically generated variables whose type cannot be determined otherwise. If tempvartype is false, then generated temporary variables whose type cannot be determined are not automatically declared.

**implicit** *default*: **false**

If implicit is set to **true** temporary variables are assigned their implicit type according to Fortran rules based on the initial letter of the name. If gendecs is on, this results in printed type declarations.

**gentranparser** *default*: **false**

If gentranparser is set to **true** Maxima forms input to gentran will be parsed and an error will be produced if an expression cannot be translated. Otherwise, untranslatable expressions may generate anomalous output, sometimes containing explicit calls to Maxima functions.

**genstmtno** *default*: 25000

Number used when a statement number must be generated. Note: it is the user's responsibility to make sure this number will not clash with statement numbers in template files.

**genstmtincr** *default*: 1

number by which genstmtno is incremented each time a new statement number is generated.

**Evaluation Forms:**

The following special functions can be included in Maxima statements which are to be translated by the gentran command to indicate that they are to be partially or fully evaluated by Maxima before being translated into numerical code. Note that these functions have the described effect only when supplied in arguments to the gentran command.

**eval**(*exp*)

Where *exp* is any Maxima expression or statement which, after evaluation by Maxima, results in an expression that can be translated by gentran into the target language. When eval is called from an argument that is to be translated, it tells gentran to give the expression to Maxima for evaluation first, and then to translate the result of that evaluation.

**rsetq**(*var*, *exp*)

Where *var* is any Maxima variable, matrix or array element, and *exp* is any Maxima expression which, after evaluation by Maxima results in an expression that can be translated by Gentran into the target language. This is equivalent to VAR : EVAL(EXP) ;

**lsetq**(*var*, *exp*)

Where *var* is any Maxima user level matrix or array element with indices which, after evaluation by Maxima, will result in expressions that can be translated by Gentran, and *exp* is any Maxima user level expression that can be translated into the target language. This is equivalent to VAR[EVAL(S1) ,EVAL(S2) , ... ] : EXP where sl, s2, ... are indices.

**lrsetq**(*var*, *exp*)

Where *var* is any Maxima matrix or array element with indices which, after evaluation by Maxima, will result in expressions that can be translated by Gentran; and *exp* is any user level expression which, after evaluation, will result in an expression that can be translated by Gentran into the target language. This is equivalent to VAR[eval(S1),EVAL(s2)...] : EVAL(EXP);

**type** (*type,v1…vn*)

Places information in the gentran symbol table to assign *type* to variables *v1…vn*. This may result in type declarations printed by gentran depending on the setting of gendecs. **type** must be called from within gentran and does not evaluate its arguments unless **eval**() is used.

**usefortcomplex** *default*:**false**

If usefortcomplex is true, real numbers in expressions declared to be complex by *type(complex,…)* will be printed in Fortran complex number format *(realpart,0.0)*. This is a purely syntactic device and does not carry out any complex calculations.

**literal**(*argl, arg2, ... , argn*)

where argl, arg2, ... , argn is an argument list containing one or more arg's, each of which either is, or evaluates to, an atom. The atoms *tab* and *cr* have special meanings. arg's are not evaluated unless given as arguments to eval. This function call is replaced by the character sequence resulting from concatenation of the given atoms. Double quotes are stripped from all string type arg's, and each occurrence of the reserved atom *tab* or *cr* is replaced by a tab to the current level of indentation, or an end-of-line character.