Hermes User's Guide

by J. P. Terlouw

Version of
June 18^{th}, 2001

"Hermes was de god, die de wegen en het verkeer daarlangs beschermde. Hij was dan ook de god van de reizigers en de handel, zelfs van de dieven, die immers ook hun geluk langs de wegen zoeken. Hij was de bode der goden, toegerust met vleugelschoenen en herautstaf, die op bevel van Zeus boodschappen overbracht en mensen veilig naar hun doel voerde; ook geleidde hij de doden naar de onderwereld."

Uit: Ernst Hoffmann:
"Goden- en Heldensagen"
bewerkt door Dr. J. H. Croon

This guide describes the user interface of the Groningen Image Processing SYstem (GIPSY). It is meant as both an introduction for new users and a reference document for more experienced users.

Contents

Chapter 1

Introduction

What is Hermes

Hermes is the user interface of GIPSY. It enables users to do multi-tasking in an organised fashion. Users have direct control over the application programs (*tasks*) they run: a running task can be suspended or aborted with a simple command; various settings can be changed while a task is running.

Parameters for a task can be specified at any time; if a task needs information that has not been specified yet, the user is prompted.

Context-sensitive help about tasks is provided at a single keystroke.

Tasks keep the user informed in two ways: they write in a log file in which the user can page and search and they can provide a one-line status message, which can be frequently updated.

Two versions of Hermes exist:

- tHermes, which runs on standard character display terminals. This version is used for interactive work.
- nHermes, a non-interactive version intended for batch work.

Normally Hermes is not started directly by the user, but rather by a shell script which prepares a number of settings before starting Hermes. Refer to the GIPSY users guide.

Conventions in this document

Text that appears literally on the terminal or workstation screen such as names of programs and commands is presented in `typewriter` font. Text which designates variable information is presented in *italic* font.

> Text, presented like this (smaller type and indented margins), contains detailed information and may be skipped at first reading.

Chapter 2

Tasks

2.1 Introduction

Tasks in GIPSY are the application programs which do useful work for the user. They consist of one or more processes (usually one) which communicate with the user through a formalized collection of interface routines. Because of the use of these interface routines, tasks can be run from any version of Hermes.

> Though tasks can be run from all versions of Hermes, not all tasks can be run *successfully* from all versions because a version of Hermes might not implement all interface functions. For instance the `EDITFILE` function is not implemented by nHermes.
>
> `COLA`-scripts can be started as tasks. For information about the use of `COLA` refer to the document `cola.dc1`.

2.2 User input

Introduction

Parameters for a task are passed to it via keywords. A keyword is a character string followed by an equal sign, e.g. `INSET=`. Keyword names are case-insensitive, i.e. uppercase and lowercase letters have the same meaning. Parameters for a task can be specified in the command that starts the task and thereafter at any moment while the task is active. A task requiring input from the user causes Hermes to find out whether it has already been specified and if not, prompt the user. Before a parameter value is passed to a task, Hermes first checks whether it meets the task's request and if not, Hermes rejects that value and prompts again.

As the execution of the task proceeds, Hermes builds up a table of the task's keywords and associated values. At task termination this table is saved in order to allow the user to run the task again with (partly) the same parameters.

Three classes of keywords are recognized:

1. *Forced*, which are keywords for which no task default is allowed. The user must respond to the prompt with a correct value.
2. *Defaulted*, which are keywords which have a default value defined by the task. If the keyword has not already been specified then the user will be prompted. If on this occasion the user types only a `RETURN`, this signifies that the task default is acceptable and should be used. Any other input will override the default.
3. *Hidden*, which are keywords for which the user is not prompted and which have a default value defined by the task. They can be specified by unprompted input by the user. The documentation of a task must be read to find out if it uses hidden keywords.

    The user can set a parameter in the task context to change requests for hidden keywords into requests for defaulted keywords. ("Unhide" hidden keywords.)

    Hidden keywords are sometimes used to control looping in a task. Each cycle round the loop, the task requests the value of a hidden keyword. If it has not been specified, the task takes the default and looping continues. If the user defines the keyword then that value is sent to the task which could cause the task to terminate the loop.

Parameter syntax

Basically Hermes is capable of providing tasks with the following types of parameters: integer numbers, real numbers, double precision reals, logicals and character strings. Tasks may however request input in the form of character strings and perform their own decoding to obtain numerical information. One parameter value can contain one or more elements.

Character input

Depending on what the task specifies, character input can be given in two ways: as *text* or as an *array*. When text is requested, all characters belonging to the keyword, including blanks and comma's, are treated as a single element. In the case of an array, the elements must be separated by blanks or comma's. If the input should contain special characters like (leading) blanks or equals signs, it can be quoted using back-quotes (`).

Normally character input is case-sensitive, i.e. uppercase and lowercase letters have different meanings, but individual tasks can deviate from this or convert to either uppercase or lowercase.

Number input

Floating point or integer numbers can be typed as numbers and/or expressions (see also section "Number input – operators, constants and functions"). They must be separated by blanks or comma's.

Lists of numbers can be specified using the '*start*:*end*:*increment*' notation, where the ':*increment*' part is optional and defaults to one. A list of n identical values can be specified as '*value*::*n*.' A list can also be used as an operand in an expression. In this case the list must be enclosed by square brackets `[ ]` or parentheses `( )`. The expression is then evaluated for each element of the list.

The operator `?` can be used to select one or more items from a list. See example below.

Examples:

```
1 2 3/3 sin(pi)        yields  1.0 2.0 1.0 0.0
log(10)::4             yields  1.0 1.0 1.0 1.0
log(10):log(100):2/4   yields  1.0 1.5 2.0
10**[0 1 5]            yields  1 10 100000
[1:3]+[90:70:-10]      yields  91 82 73
[20:30]?[3 4 5]        yields  22 23 24
```

Suppose an inclination at keyword `INCL=` must be given in degrees, but you want to give the input in axis ratio.

```
INCL= deg(acos(0.3 0.5))
```

The argument for `acos` is a list consisting of the elements 0.3 and 0.5. The expression evaluates these two values for `acos` and converts the resulting values to degrees. In this way the values 0.3 and 0.5 are converted to the angles 72.5424° and 60.0°.

Logical input

Logicals are decoded in the following way: `YES`, `JA` and `TRUE` result in a logical which is true; `NO`, `NEE` and `FALSE` give a logical which is false. It is sufficient to give the first letter of the possible affirmative and negative replies. Any other answer will result in a syntax error.

Examples:

```
PLOTGRIDS= N
OK= YES
```

Input Lists and Recall Files

If a task requests a keyword repeatedly, the user can pre-specify inputs in two ways. The first method is separating the inputs with semicolons, e.g.

```
POS= 10 20; 15 25; 30 30
```

Note that a trailing semicolon designates an empty input for which a default is to be taken. The second method consists of specifying a file ("*recall file*") as input to the keyword. This must be a text file with a name like "*name*`.rcl`". Every line in this file is a separate input. Lines must not be longer than 500 characters. Either the whole file or a part of it can be specified:

<*name*	use the whole file;
<*name* n:m	use line numbers n to m;
<*name* :n	use line numbers 1 to n;
<*name* n:	use line numbers n to the end of the file;
<*name* n	use only line number n.

Here the n:m range specifier should not be confused with a list of numbers as described above.

It is possible to specify a recall file in a semicolon-separated list, but within a recall file semicolons cannot be used. Recall files cannot be used recursively, i.e. a recall file can not contain a reference to an other recall file.

When the task has "consumed" all pre-specified input on a keyword, the next request for that keyword will cause the user to be prompted in the normal way. Inputs given on repeatedly requested keywords are stored in a semicolon-separated list, so if a task is re-run with the previous set of parameters, *all* inputs are available again.

If unprompted input is given on a keyword of which all pre-specified input has not been consumed yet, the new input will supersede the *complete* previous set of inputs.

Default files

Keywords can be prespecified in a text file with a name like "*name*`.def`". This *default file* can contain any number of *keyword=value* pairs. Any user supplied parameters will supersede those obtained from a default file.

The name of the *task* determines which default file will be read, not the name of the executable. This has consequences (and possibilities) for tasks which are run under an alias name.

It is also possible to specify defaults to be used by every task. For this purpose the file `tasks.def` us used. Such global defaults have a limited use. The keyword `GGIOPT=`, used by tasks with a graphical user interface, is an example.

Default files can reside in both the current working directory and the directory `.gipsy` under the user's login directory. Keywords from default files in the working directory supersede identical keywords from `.gipsy`. And keywords from task-specific default files supersede identical global keywords.

Number input – operators, constants and functions

In numerical expressions the following operators are available:

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`**`	exponentiation

The following predefined constants are available:

`PI`	π (3.14159....)
`C`	speed of light (SI)
`H`	Planck (SI)
`K`	Boltzmann (SI)
`G`	gravitation (SI)
`S`	Stefan-Boltzmann (SI)
`M`	mass of Sun (SI)
`P`	parsec (SI)
`BLANK`	universal undefined value

The following mathematical functions can be used:

`abs`(x)	absolute value of x
`sqrt`(x)	square root of x
`sin`(x)	sine of x
`asin`(x)	inverse sine of x
`cos`(x)	cosine of x
`acos`(x)	inverse cosine of x
`tan`(x)	tangent of x
`atan`(x)	inverse tan of x
`atan2`(x, y)	inverse tan (mod 2π) $x = \sin$, $y = \cos$
`exp`(x)	exponential of x (e^x)
`ln`(x)	natural log of x
`log`(x)	log (base 10) of x
`sinh`(x)	hyperbolic sine of x
`cosh`(x)	hyperbolic cosine of x
`tanh`(x)	hyperbolic tangent of x
`rad`(x)	convert x to radians
`deg`(x)	convert x to degrees
`erf`(x)	error function of x
`erfc`(x)	1-error function

`max`(x, y)	maximum of x and y
`min`(x, y)	minimum of x and y
`sinc`(x)	$\sin(x)/x$ (sinc function)
`sign`(x)	sign of x (-1, 0, 1)
`mod`(x, y)	remainder of x/y
`int`(x)	truncates to integer
`nint`(x)	nearest integer
`ranu`(x, y)	generates uniform noise between x and y
`rang`(x, y)	generates gaussian noise with mean x and dispersion y
`ranp`(x)	generates poisson noise with mean x
`ifeq`(x, y, a, b)	a if $x = y$, else b
`ifne`(x, y, a, b)	a if $x \neq y$, else b
`ifgt`(x, y, a, b)	a if $x > y$, else b
`ifge`(x, y, a, b)	a if $x \geq y$, else b
`iflt`(x, y, a, b)	a if $x < y$, else b
`ifle`(x, y, a, b)	a if $x \leq y$, else b

The following functions can be used to obtain data from GIPSY-sets and files:

`descr(`*set*`,` *name*`)`	descriptor item *name* from (sub)set(s) *set*.
`table(`*set*`,` *tab*`,` *col*`,` *rows*`)`	cell(s) from column *col* of table *tab* in (sub)set *set*.
`image(`*set*`,` *box*`)`	pixel(s) from (sub)set *set*.
`file(`*name*`,` *cols*`,` *rows*`)`	number(s) from a column in a text file.

The *set*- and *box* arguments have the same syntax as described in `input.doc`. The argument *rows* has the same syntax as is used for recall files (n:m, :n, n: and n). Lines in the file to be read by the `file`-function must not be longer than 500 characters. Comments can be included in the file by prefixing it by one the characters `!` or `#`. All items in the file should be numbers. If an item cannot be recognized, the value `BLANK` will result. In this way `BLANK`s can also be specified explicitly, e.g. by using the word "BLANK". Empty columns at the end of a line will also yield `BLANK`s. But a completely empty or blank line is treated as a comment.

The following functions can be used on lists:

`count(`x`)`	number of elements in x
`mean(`x`)`	the average of the elements in x
`sum(`x`)`	the sum of the elements in x

Hermes also supports named variables. To these variables numbers and lists of numbers can be assigned. Variable names can be up to 20 characters long; they are not case-sensitive. The maximum number of variables is 40. A variable can be deleted by assigning the value `BLANK` to it. Assignment to a variable is accomplished by issuing a special Hermes command:

variable`=` *value*

or, in tHermes, doing the assignment in the tHermes calculator window (see page 23).

2.3 Task context

Tasks all run in a *context*. This context consists of a number of user-settable parameters:

- Error level. If a task generates an error at or above the current error level, Hermes will abort the task. Default: 4.
- Message level. If a task generates an error at or above the current message level, Hermes will display the associated error message. Default: 1.
- Output mode. This parameter determines whether test messages from tasks will

be displayed and whether messages unnecessary for experienced users will be suppressed. Default: "`NORMAL`".

- Device status. Two parameters indicating whether messages sent to either the screen or the log file will actually be written. Default: screen "`ON`" and log file "`ON`".
- Hide status. This parameter determines whether "hidden" user input requests from the task will cause the user to be prompted anyway. Default: "`ON`", i.e. the user will not be prompted for hidden keywords.
- Working directory. Default: the directory from which Hermes was started. The directory of running tasks cannot be changed.
- Finally there are four parameters which determine what kind of status messages will be logged.

Tasks normally inherit their context from a template context. If a task is started by an other task, it inherits the context of that task. Both the template context and the context of any running task can be changed by the user. The way in which this can be done depends on the specific implementation of Hermes. When Hermes is started, the template context contains the sensible defaults listed above, possibly modified by parameters from the Hermes defaults file (see below).

2.4 Hermes defaults file

The template task context and some other parameters can be defined in a defaults file which will be read when Hermes is started. This file must have the name `hermes.def` and reside in the directory from which Hermes is started or in the directory `.gipsy` under the user's login directory. (Parameters from the file in the start-up directory have precedence.) Its format is a text file which may contain any number of the keyword-value pairs from the table below.

`DIRECTORY=`	working directory for tasks.
`TASKPATH=`	search path for tasks.
`NTASK=`	number of task entries; must be > 0.
`HIDE=`	hide status for keywords; must be `ON` or `OFF`.
`ERRORLEVEL=`	error level; must be in the range 1 to 4.
`MESSAGELEVEL=`	message level; must be in the range 1 to 4.
`OUTPUTMODE=`	output mode; must be `NORMAL`, `EXPERT` or `TEST`.
`SCREEN=`	screen output status; must be `ON` or `OFF`.
`LOGFILE=`	log file output status; must be `ON` or `OFF`.
`PRINTER=`	printer number for screen hardcopies.
`MATCH=`	minimal matching; must be `ON` or `OFF`.
`TEKDEVICE=`	Tektronix terminal type (tHermes only).
`BEEP=`	terminal bell; must be `ON` or `OFF` (tHermes only).
`KEYHELP=`	add keyword help to standard help; must be `ON` or `OFF`.
`WEB_BROWSER=`	full-path location of Hypertext browser and optional command line arguments.
`STARTBROWSER=`	start Hypertext browser automatically.

Example of a typical `hermes.def` file:

```
taskpath=. $HOME/owntasks/$gip_arch
directory=/dj3/users/cia/gipsy
outputmode=expert
match=on keyhelp=off
printer=3
```

Chapter 3

tHermes

3.1 Introduction

tHermes is the version of Hermes which communicates with the user through standard character display terminals, including X11 terminal windows. It organises the terminal screen in a special window-like way. The most important permanent components (windows) of the screen lay-out are:

- User Command Area (UCA). This occupies the lower two lines of the terminal screen and is the focus of user-to-task communication.
- Task Status Area (TSA). This normally consists of four lines above the UCA which each can contain information reflecting the current status of a task.
- Common Output Area (COA), the remaining top part of the screen which shows a part of the log file. Here user commands are logged and tasks can write text.

Other permanent components are a clock display and numbers which indicate which part of the log file is mapped to the COA.

Components of the screen lay-out which are not permanent are called transient windows. These overlay the permanent screen components or other transient windows. For example the context-sensitive help display overlays (part of) the COA. Other examples of transient windows are a menu window allowing the user to change specific task settings, prompter windows in which for instance a search string can be specified, a Unix shell window, etc.

Many functions of tHermes can be invoked by typing a control key (designated as `CTRL-`c, where c is a letter), or by typing an escape sequence (the `ESC` character followed by another character). These are summarized in the tables on pages 26 and 28. Important control keys are: `CTRL-C` to abort a task or to break out of a transient window, `CTRL-O` to switch the "keyboard focus" between terminal windows waiting for keyboard input, `CTRL-L` to repair a damaged screen lay-out and `CTRL-Q` to quit from Hermes. Hermes can also be terminated by typing `CTRL-X CTRL-C` or `ESC Z` or by typing `QUIT`, `STOP`, `END` or `EXIT` in the UCA.

It is not possible to run more than one GIPSY session in the same directory. If tHermes is started in a directory where GIPSY already runs, it exits immediately with an error message. (But nHermes will wait until the other session is finished.)

3.2 User Command Area

The User Command Area (UCA) consists of the bottom two lines of the terminal screen where most interaction with the tasks takes place. The user can type task-related commands in the UCA and Hermes can put there task-related prompts. An important aspect of the prompts is that they are treated in exactly the same way as information typed by the user, i.e. the user can erase or modify a prompt supplied by Hermes as if he typed it himself. Prompts in the UCA are always in the form of a legal complete or partial command.

The information in the UCA is used as the context for a number of functions of Hermes. E.g. `TAB` (or `CTRL-I`) causes the user document of any taskname present in the UCA to be displayed and `CTRL-C` aborts the task if it is active.

The two lines of the UCA are one logical long line which can be edited using the following control keys of which most are EMACS-compatible.

`CTRL-A`	move to start of line
`CTRL-B`	move one position back
`CTRL-D`	forward delete one character
`CTRL-E`	move to end of line
`CTRL-F`	move one position forward
`CTRL-H`	backward delete one character
`CTRL-K`	forward delete rest of line
`CTRL-T`	replace complete line with next active task name (see below)
`CTRL-U`	delete complete line
`DEL`	backward delete one character
`space`	replace complete line with next prompt for user input (see below)

Characters `CTRL-T` and `space` are somewhat special. `CTRL-T` erases the UCA and

then prompts with the name of an active task. It does this in a circular fashion so if more than one task is active, another taskname is displayed when `CTRL-T` is pressed again.
If the UCA is "free" (a subtle concept explained below), pressing the space bar does a similar thing for tasks waiting for input. This makes it easier for the user to do something else before he supplies the requested input, e.g. run another task which supplies information that helps to give the input.
The UCA is "free" when it is empty, or when it only contains an unmodified prompt supplied by Hermes.

Hermes can provide 'minimal matching' for task names and user input keywords typed in the UCA. This facility can be switched on and off by typing `ESC M`. It can also be switched on using the statement `MATCH=ON` in the Hermes defaults file. If it is on, task names will be completed when a blank or `RETURN` is typed on a command line that only contains an abbreviated task name. Keywords will be completed when '`=`' is typed. If the keyword is ambiguous, Hermes will beep and refuse to accept the keyword; if it is not known, Hermes will also beep but will accept the keyword.
Hermes obtains the taskname and keyword information from the `.dc1` documents in `$gip_tsk`. For this reason minimal matching does not work for tasks which are not installed.

Previously entered commands can be brought back to the UCA by using the ANSI up-arrow and down-arrow keys. If the user has typed some text in the UCA, only commands which match this text will be brought back. The commands are saved in the text file `HERMES.cmd`, so that they are preserved between sessions.

3.3 Task Status Area

The Task Status Area (TSA) is the middle section of the terminal screen. It normally consists of four lines which each can contain status information of an active task.

The number of these status lines can be specified using the keyword `NTASK=` in `hermes.def`.

The format of a task status entry is one of the following:

- When a task has been activated, but is not running yet:
  taskname `WAITING TO BE RUN`
- When a task is running:
  taskname `RUNNING`

or
taskname   *status message supplied by task*

- When a task is prompting for user input:
  taskname   *prompt message supplied by task*
- When a task is waiting for a file to be edited by the user:
  taskname   *message supplied by task*
- When a task is suspended:
  taskname   `PAUSING`
  or
  taskname   `PAUSING` *message supplied by task*
- When a task is waiting for an other task to finish:
  taskname   `WAITING FOR` *other taskname*
- When a task has finished processing normally:
  taskname   `+++ FINISHED +++`
- When a task has been aborted by the user:
  taskname   `USER ABORT`
- When a task has encountered an error fatal to execution:
  taskname   *error message* `-FATAL`
- When a task has crashed (i.e. exited without notification to Hermes):
  taskname   `CRASHED`
- When a task cannot be started (incorrect executable, e.g. compiled for a different architecture):
  taskname   `COULD NOT BE STARTED`

3.4 Common Output Area and Log File

The Common Output Area (COA) on the terminal screen is the remaining top area of the terminal screen not used for the UCA or TSA. It is a viewport on the log file. Hermes treats the log file as a "book" with numbered pages. Normally Hermes shows the current page of the book, that is the page on which output is currently being written.

The COA has two modes of operation: page mode and non-page mode. In non-page mode the page on the screen is always the current page. If necessary, Hermes will flip the page. The current page number is displayed at the extreme right of the Task Status Area. In page mode the information on the screen will stay there until the user instructs Hermes to change the page. Though the user does not see it, output can still be written to the log file. In page mode the number of the displayed page is shown to the left of the current page number. Page mode can be switched on and off by pressing `CTRL-P`.

When in non-page mode, Hermes can also scroll the COA instead of flipping pages. This scroll mode can be switched on and off by typing `CTRL-X S`. In scroll mode output is much slower than in 'flip mode'. Therefore this mode is not recommended for use with slow terminals.

The log file can be traversed in backwards direction by pressing the `CTRL-Z` key (or on ANSI keyboards the `PageUp` key). Paging forward is done with the `CTRL-V` key (or ANSI `PageDown`).

Text search in the log file is done by pressing `CTRL-R` (reverse) or `CTRL-S` (forward). Hermes then prompts the user for a search string. Searching in Hermes is case-insensitive.

A specific page number can be brought to the screen by typing `ESC G`, whereafter Hermes prompts for a page number. Signed numbers are treated as relative page numbers.

Sections of the log file can be printed by typing the `ESC H` sequence. Hermes then prompts for a range of pages to be printed. The range can be specified as a single number, in which case only that page will be printed, or using the format '*start*:*end*'. Either *start* or *end* may be omitted, in which case the number of the currently displayed page will be used instead. The default range is the page currently displayed and the preceding two pages. The printer can be selected by typing `ESC P`, which causes a menu to be presented from wich a printer can be chosen. Instead of a printer, output can also be written to a file. To achieve this, choose option `F` from the printer selection menu. This redirection can be cancelled by specifying an empty string.

> The statement that the COA is a viewport on the log file is not quite true. In fact the log file and the COA are different output streams to which tasks can send information selectively. Most tasks however send the same information to both the log file and the COA.

COA command summary

`CTRL-P`	enter or leave page mode
`CTRL-Z`	display previous page and enter page mode
`CTRL-V`	display next page
`CTRL-R`	search forward for text string
`CTRL-S`	search backwards for text string
`CTRL-X S`	toggle scroll mode
`ESC G`	go to specified page number
`ESC H`	make hardcopy of specified page numbers
`ESC P`	select a printer for the hardcopy print command
`ESC <`	go to first page
`ESC >`	go to last page

3.5 Running Tasks

Starting tasks and supplying parameters

Tasks can be started with or without specifying parameters. To start a task without parameters, the name of the task must be typed in the UCA, followed by a `RETURN`:

taskname

To specify parameters, any number of keyword-value pairs separated by blanks can be added to the taskname:

taskname $keyword_1$=$value_1$ $keyword_2$=$value_2$ ...

This format is also used to supply parameters to an already running task.

If there is something wrong with the command, an error message is displayed in the lower right corner of the screen.

> Tasknames may be preceded by an explicit directory. In this case, the task must be present in the specified directory; the task path is not searched. The directory may contain one or more environment variables (indicated by a leading `$`-character).
>
> Tasks can be run again with (partly) the same parameters by prefixing the task start command with an exclamation mark (`!`). Any explicitly specified parameters will supersede the corresponding old parameters. In this case the default file is not used. For historical reasons the previous parameters of a task are called the task's *macro*.

Tasks can be run under a different name ("alias"). The command to do this is:

aliasname(*filename*) ...

Aliasname is the taskname under which the task will run and *filename* is the name of the task's executable file, specified in the correct case (lowercase for standard GIPSY tasks).

Inspecting and saving input parameters

The current set of keyword-value pairs of a task can be inspected by typing `ESC K` while the name of the task is present in the UCA. This causes (part of) the COA to be overlaid with a display containing the current user input parameters of the task. Whenever there is a change in one of the parameters, the display will be adjusted. If the amount of space on the screen is not sufficient to show all parameters, the user can page through the display just like paging through the log file. `CTRL-V` pages forward; `CTRL-Z` pages backwards. To inspect its keywords, the task need not be active. Typing `ESC K` again will remove the overlay screen.

The displayed set of parameters can be saved in a file which can later be used as a default file. This can be achieved by typing `ESC W`, which causes the user to be prompted for a filename (default: *taskname*`.def`).

The contents of a single keyword from the displayed set of parameters can be saved in a file which can later be used as a recall file. Type `ESC R`. The user is first prompted for the keyword, then for the filename.

The parameters of a task can be edited using a standard editor. The editor used is the one defined by the `EDITOR` environment variable and it is invoked by typing `ESC E`. If there is currently a set of parameters displayed on the screen (as the result of typing `ESC K`), these parameters will be edited; otherwise the name of the task is derived from the UCA contents. 'Behind the scenes' Hermes will remain active, but user communication with Hermes is not possible because all typing goes into the editor. Using this editor is the only way in tHermes to change input parameters for a task that is not active.
If Hermes cannot derive a name from either the parameter display or the UCA, it will prompt for a file name as if `CTRL-X E` was typed.

Aborting, suspending and resuming

Tasks can be aborted by typing `CTRL-C`. They can be suspended and resumed by typing `CTRL-W` or `CTRL-G`. If more than one task is active, the target task's name must be present in the UCA.

Changing the task context

The context of a task can be changed using a pop-up menu that can be activated with the sequence `ESC S`. The following context parameters can be changed with this menu:

- error level: 1–4
- message level: 1–4
- output mode: normal, expert or test
- screen output: on or off
- log file output: on or off
- hide "hidden" keywords: on or off

To change the context of a running task, its name must be present in the UCA. If the UCA is empty, the template context will be changed.

Directories and paths

The working directory can be changed by typing `ESC D`. Hermes then prompts the user for a new directory. This directory will then be used for tasks started after the change. Running tasks and Hermes itself are not affected.

The task path can be changed with `ESC T`. Upon the prompt that follows one or more directories separated by blanks can be specified. A directory may contain (or consist of) one or more environment variables (indicated by a leading `$`-character). Whenever a task start command without an explicit path is given, this list is scanned to find the task. If it is not in one of the specified directories, Hermes tries to find the task in `$gip_exe`. The default path is '`.`' (the current working directory).

Getting help

To obtain help information about a task or about Hermes, `TAB` (or `CTRL-I`) can be pressed. This causes (part of) the COA to be overlaid with a display containing the user document of the task of which the name is present in the UCA. If there is also a keyword present (e.g. as the result of a prompt), Hermes positions the document at the first occurrence of the keyword. In this case Hermes will also provide a keyword help window which partially overlays the task documentation. For more information see the section "Keyword help" below. This feature can be switched off by putting the statement `KEYHELP=OFF` in the Hermes defaults file (`hermes.def`).

The help display will 'follow' the task execution; if other prompts appear, the display will be repositioned. If the amount of space on the screen is not sufficient to show the whole document, the user can page through the display just like paging through the log file. `CTRL-V` pages forward; `CTRL-Z` pages backwards. If the UCA is empty, pressing `TAB` will display a summary of Hermes' commands.

It is possible to search for a text string in the document. Pressing `CTRL-S` or `CTRL-R` will cause the user to be prompted for a search string.

Pressing `TAB` again will remove the overlay screen(s).

Hypertext help

Typing `ESC X` will cause Hermes to start a WWW browser. The GIPSY start-up script attempts to determine which browser can be used. This choice can be overridden by defining the environment variable `WEB_BROWSER` before starting GIPSY or by defining the variable `WEB_BROWSER` in the Hermes defaults file (`hermes.def`). Any command line options to the browser can also be specified in the defaults file (but *not* in the environment variable). When the defaults file variable `STARTBROWSER` is set to "`ON`", the browser will be started automatically.

Initially the home page of the new-style GIPSY documentation in hypertext will appear. When the browser is either NCSA Mosaic or Netscape, pressing `TAB` will display the task document in the browser instead of overlaying the COA.

When `ESC X` is typed again, the browser will be terminated and Hermes will revert to normal operation.

Keyword help

Hermes can provide specific help about user input keyword prompts. This can be switched on and off independently of task-related help by typing `ESC TAB`. When it is active, the available options are displayed in the bottom line of the window. The possible options are:

`PageUp`	page back in window
`PageDown`	page forward in window
`ESC` i	go to reference i ($i = 1 \ldots 9$)
`ESC -`	go back one reference
`ESC 0`	go back to "root text"

The window can be temporarily removed by removing the associated prompt by typing

CTRL-U. When the prompt is brought back by typing a space, the window will re-appear.

Task control summary

taskname	start a task
taskname keyword=*value* ...	start a task and/or supply parameters
!*taskname*	start a task with the previous set of input parameters
!*taskname keyword*=*value* ...	like !*taskname*, but given parameters replace previous
taskname(*filename*) ...	run executable *filename* as task *taskname*

`CTRL-C`	abort task
`CTRL-G`	suspend or resume task (synonym of `CTRL-W`)
`CTRL-I`	(=`TAB`) activate or deactivate help display
`CTRL-W`	suspend or resume task (synonym of `CTRL-G`)

`ESC D`	change working directory
`ESC K`	activate and deactivate user input parameter display
`ESC E`	edit parameters or user-specified file
`ESC W`	save displayed parameters in (default) file
`ESC R`	save contents of keyword in (recall) file
`ESC S`	activate menu to modify task context
`ESC T`	change task path
`ESC TAB`	switch user input keyword help display on or off

3.6 Unix shell and text editor

Integrated shell

Typing `ESC U` causes the COA to be overlaid by a Unix shell window or, if the Unix shell is already present, causes the window to be popped down. The shell used is `/bin/csh`. Because the shell is not connected to a real terminal, functionality is somewhat limited: e.g. a screen editor would not work. Also some commands behave differently, e.g. `ls` produces a one-column list. (A multi-column list can be obtained with `ls -C`.)

By default information displayed in the Unix shell window is not logged. Logging to the COA and log file can be enabled (and disabled) by typing `ESC L`.

This shell can be completely terminated by typing `exit` in the shell window.

Overlay shell

Typing `CTRL-X U` causes a Unix shell to be started which overlays the whole terminal screen. 'Behind the scenes' Hermes will remain active, but because all typing goes into the shell, user communication with Hermes is not possible. The shell used is the one defined by the `SHELL` environment variable or, if this is not defined, `/bin/csh`.

This shell can be terminated by typing `CTRL-D` or `exit`.

Text editor

Typing `CTRL-X E` causes the user to be prompted for the name of a file to be edited by a text editor which will overlay the whole terminal screen. 'Behind the scenes' Hermes will remain active, but because all typing goes into the editor, user communication with Hermes is not possible. The editor is the one defined by the environment variable `EDITOR`; if this is not defined, the GIPSY start-up script will attempt to find a one of the editors `mem`, `vi` or `emacs` in the user's path. If no editor can be found, an error message is displayed.

Tasks can also request that a text editor be started. In this case Hermes prompts the user to allow the editor to be started. To start the editor, type `RETURN`; to cancel the edit, `CTRL-C` can be typed and the edit can be postponed by shifting the keyboard focus away from the prompt window by typing `CTRL-O`.

3.7 Tektronix graphics

If the terminal at which Hermes runs is capable of emulating a Tektronix graphics terminal, `CTRL-X T` can be typed to switch on the graphics mode. This mode can be used in connection with tasks for which the plot device `THERTEK` was selected. If the terminal type has not been defined yet, the user will be prompted. This information can also be specified in the Hermes defaults file `hermes.def`. In this case use: `TEKDEVICE=`*terminaltype*. Allowable terminal types are listed in the table below.

In Tektronix mode, Hermes continues to operate normally, but user interaction is somewhat limited. To protect the graphics image, characters typed are not echoed and the overlay Unix shell and editors cannot be started.

Typing `CTRL-X T` again will switch Tektronix mode off. If possible, Hermes then switches back to normal terminal mode, but some terminals do not support this; these terminals must be switched back manually by the user.

Hermes 'remembers' the contents of the graphics image, so if Tektronix mode is switched on again, the image is restored.

Type		switch back
`LADA`	KGB's LAT terminal emulator	automatic
`NCSA`	NCSA Telnet	manual
`XTERM`	X Window terminal emulator	automatic

3.8 Calculator

Typing `ESC C` causes a calculator window to be popped up. In this window, the user can type any numerical expression that would also be acceptable as user input. The resulting value is displayed in the bottom line of the window; for lists only the first and last value are shown. It is also possible to assign a value to a variable.

The calculator window can be removed by typing `ESC C` again.

Chapter 4

nHermes

4.1 Introduction

nHermes (non-interactive Hermes) is the version of Hermes for a non interactive (i.e. batch-) environment. It is derived from the interactive tHermes.

Because the user is not available to be consulted, default values for unspecified keywords will be taken if available. If there is no default available, an error message will be written to the log file and nHermes will stop with the current command. When a command is finished, nHermes will start the next command if given.

If the timeout option is set, a timeout will occur after the user supplied number of minutes and nHermes will terminate the command it is dealing with at that time. The timeout works separately for each user command.

nHermes maintains a log file to which a history of all operations is written. This file can be inspected or printed afterwards.

4.2 Using nHermes

The syntax for starting nHermes is:

`nhermes -l`*logfile* `-t`*minutes command ... command*

The options may appear in any order and may be intermixed with the commands. The `-l` and `-t` flags are optional.

The log file will always have the extension `.LOG` and it contains a history of all operations done and the information generated by the tasks which have been run. The default log file name is `GIPSY.LOG`.

The purpose of the timeout option is to terminate the user command when it uses more time than is thought to be reasonable, e.g. if it is executing an endless loop due to a program or user error.

command can be a `COLA` script or a name of a task with their parameters. The name of the task or the `COLA` script with the parameters must be suitably quoted to keep it together, e.g. between double quotes. If this is not done, nHermes will handle them as if they were separate user commands.

Example:

```
nhermes -lMylog "FLUX INSET=cg1517 v"
```

This command causes nHermes to start task `FLUX` on all subsets of set `cg1517` and put the results in file `Mylog.LOG`.

Status return:

If the last command executed by nHermes was successful, nHermes exits with status code 0; when the last command failed, the value 1 is returned.

Appendix A

tHermes key definitions

Escape sequences

`ESC B`	switch beep associated with prompts on or off
`ESC C`	pop up and down calculator window (23)
`ESC D`	change working directory (19)
`ESC E`	edit a task's input parameters or a user-specified file (18)
`ESC G`	go to specified page number in COA (16)
`ESC H`	make hard copy of COA (16)
`ESC K`	pop up and down keyword display (18)
`ESC L`	switch logging of integrated Unix shell on or off (21)
`ESC M`	switch UCA minimal matching on or off (14)
`ESC P`	select printer for the `ESC H` command (16)
`ESC R`	(if keyword display active) write single task input parameter to file (18)
`ESC S`	change task or template context (19)
`ESC T`	change task path (19)
`ESC U`	pop up and down integrated Unix shell (21)
`ESC W`	(if keyword display active) write all task's input parameters to file (18)
`ESC X`	activate or deactivate hypertext help (20)
`ESC Z`	quit tHermes (13)
`ESC <`	go to first page of COA
`ESC >`	go to last page of COA
`ESC ESC`	circulate through task names in UCA
`ESC TAB`	switch user input keyword help display on or off

ANSI keys

↑	bring back previous commands to UCA (backward direction) (14)
↓	bring back previous commands to UCA (forward direction) (14)
←	move one position back in input line (=`CTRL-B`)
→	move one position forward in input line (=`CTRL-F`)
`PageUp`	page back in COA, help display or keyword display (=`CTRL-Z`)
`PageDown`	page forward in COA, help display or keyword display (=`CTRL-V`)

Control characters

`CTRL-A`	move to start of input line (13)
`CTRL-B`	move one position back in input line (13)
`CTRL-C`	in UCA: abort task; in transient window: break out of window (18)
`CTRL-D`	forward delete character in input line (13)
`CTRL-E`	move to end of input line (13)
`CTRL-F`	move one position forward in input line (13)
`CTRL-G`	resume or suspend task (18)
`CTRL-H`	back delete one character
`CTRL-I`	(=`TAB`) pop help screen up or down (19)
`CTRL-J`	enter input line
`CTRL-K`	forward delete rest of input line (13)
`CTRL-L`	rewrite terminal screen (13)
`CTRL-M`	(=`RETURN`) enter input line
`CTRL-N`	(undefined)
`CTRL-O`	switch between windows accepting keyboard input (13)
`CTRL-P`	enter or leave COA page mode (16)
`CTRL-Q`	quit tHermes (13)
`CTRL-R`	reverse search in COA or help display (16, 20)
`CTRL-S`	forward search in COA or help display (16, 20)
`CTRL-T`	circulate through task names in UCA (13)
`CTRL-U`	clear input line (13)
`CTRL-V`	page forward in COA, help display or keyword display (16, 18, 20)
`CTRL-W`	suspend or resume task (18)
`CTRL-X`	initial character for special special combinations
`CTRL-Y`	(undefined)
`CTRL-Z`	page back in COA, help display or keyword display (16, 18, 20)

`CTRL-X E`	edit user-specified file (22)
`CTRL-X S`	switch COA scroll mode on or off (16)
`CTRL-X T`	switch Tektronix emulation on or off (22)
`CTRL-X U`	overlay Hermes with shell (22)
`CTRL-X CTRL-C`	quit tHermes (13)

(Numbers in parentheses refer to pages in this document.)

Appendix B

Files used by tHermes

File name	Usage	Format
GIPSY.LOG	log file	text file
SCREEN.LOG	terminal screen pages	internal
HERMES.keywd	user input parameters ("macro")	text file
HERMES.cmd	previous commands (for command recall)	text file
*.def	default file	text file
*.rcl	recall file	text file
$TEKNAME	Tektronix graphics file (env. variable)	PGPLOT ("THERTEK")
taskname.key	scratch file	
screen.prt	scratch file	
keys.tmp	scratch file	

The file GIPSY.LOG contains the log of any number of GIPSY sessions and can be edited and printed. If the information contained by it is not necessary anymore, it can be deleted.

SCREEN.LOG contains the screen pages of any number of GIPSY sessions. It cannot be edited or printed. Any modification of this file can cause tHermes to fail. If the information contained by it is not necessary anymore, it can be deleted.

HERMES.keywd contains all current user input parameters of previously run tasks. This file can be edited, provided its format is maintained. It would however be very seldomly necessary to edit this file.

HERMES.cmd contains the last 1000 commands which were typed into the UCA. It is used for tHermes' command recall feature. This file can be edited, but this would be very seldomly necessary.

The files with the extension .def are the user-defined default files.

The files with the extension `.rcl` are the user-defined recall files.

The file to which the environment variable `TEKNAME` points, contains information to be displayed on a Tektronix 4010 compatible graphics terminal. It can only be written by the `PGPLOT` driver for the "`THERTEK`" device.

The scratch files have normally all been cleaned up when tHermes exits. If such a file is present anyway, it can be deleted without any problem.