1. Introduction

1.1 Purpose of this part of the documentation

This description of $\kappa\lambda\epsilon\iota\omega$ is intended to give users who are already familiar with the system a brief but comprehensive overview of currently available functions. This description is in fact the most up-to-date description of the system. Where this manual conflicts with other parts of the documentation – in particular descriptions of individual functions published in the "Halbgraue Reihe zur Historischen Fachinformatik" – the manual takes precedence.

1.2 The principles underlying command-based work

Users generally issue their instructions to software systems in one of two different ways: either by giving commands, formulated in a formal *command language*, or by choosing from a series of options displayed in the form of a *menu*.

$\kappa\lambda\epsilon\iota\omega$ gives the user the option of choosing a menu-driven interface. This is described in a separate manual (volume B10 in the "Halbgraue Reihe", and in part in the English tutorial volume), which is included in the system documentation for version 4.3.1 and subsequent versions. The menu interface allows the user to access $\kappa\lambda\epsilon\iota\omega$'s most important functions: in fact, certain parts of the system, such as the interactive full-text system, are only accessible via the menu interface because of the way they work. Generally speaking, however, $\kappa\lambda\epsilon\iota\omega$'s menu interface has two limitations. First, there is still a considerable amount of work to be done before users can access *all* the system's functions from menus; and second, many of the commands in the menu-driven version are deliberately somewhat limited in terms of flexibility, to keep the interface as clear and user-friendly as possible.

Thus the full power of the system is only available to users who are prepared to familiarise themselves with the command language. The main disadvantage of a command-based system is that users must learn the command language, which takes time and effort. The advantage of this kind of system, on the other hand, is that once users have mastered the command language they can formulate highly sophisticated queries for themselves, including queries which even the software's original designers may not have imagined in detail.

1.3 Some practical tips

The commands described below should be written in a command file. You can use any text editor or word processing program[1] to write a command file, after which you import it into $\kappa\lambda\epsilon\iota\omega$ for processing.

Regardless of the operating system you are using, you can import a command file into $\kappa\lambda\epsilon\iota\omega$ by entering the following command:

`kleio [ <input file> | - [ <output file> ] ]`

If you call up $\kappa\lambda\epsilon\iota\omega$ *without* first defining an `<input file>` or `<output file>`, the system will operate in interactive mode, i.e. from the menu-driven interface described in volume B10 of the "Halbgraue Reihe". If you activate the program in this mode unintentionally, you can always quit the program by pressing the ESCAPE key on your computer keyboard.

The `<input file> and <output file>` share the following characteristics: if the files are stored in the same directory (catalogue, file directory, etc.) as the one you are currrently in, every filename component should be joined to other filename components by a full stop (period). So for example, on a computer running MS-DOS you would write `file.ext`; on a VM/CMS system you would write `fn.ft.fm` – **not** `fn fm ft`. Similarly, whenever you refer to another file system (directory, catalogue, file directory etc.), you should use standard operating system conventions in order to specify the location of the files you want to access. Thus on an MS-DOS system, for example, you might enter `\mydirectory\mysubdirectory\myfile.ext`; whereas on a UNIX system you would enter `/mydirectory/mysubdirectory/myfile`.

If no `<input file>` exists yet, or if the local operating system will not allow you to read the input file's contents, $\kappa\lambda\epsilon\iota\omega$ will abort instead of executing any further commands.

If the `<output file>` already exists, $\kappa\lambda\epsilon\iota\omega$ will attempt to overwrite it. If the operating system prevents the program from doing this, or does not allow you to create a brand new output file, $\kappa\lambda\epsilon\iota\omega$ will abort instead of executing any further commands.

[1] However, you must take care to ensure that any command files you write are *ASCII files*, i.e. that your text editor or word processor does not insert any internal printing control codes into the text. $\kappa\lambda\epsilon\iota\omega$ interprets tabs as single spaces. $\kappa\lambda\epsilon\iota\omega$ supports most of the best-known word processing programs, including: (a) WordPerfect, Nota Bene, XY Write and Euro Script, any of which you may use as long as you do not attempt to set specific printing attributes. (b) Word: this program automatically places binary data at the beginning and end of every file. Methods of suppressing this feature vary depending on the version of Word and particular preset options you are using. You can distinguish an ASCII file by the fact that it does not contain any incomprehensible characters when you output the file to screen using the `type` command at the DOS prompt. (c) WordStar: careful! – this program uses its own internal code for individual characters. Save your file as a Non-Document File; you are unlikely to experience any problems with it unless it contains special foreign characters. If it does, you should convert it into a true ASCII file using WS Convert before attempting to import the file into $\kappa\lambda\epsilon\iota\omega$.

If you call up $\kappa\lambda\epsilon\iota\omega$ by entering:

`kleio <input file>`

$\kappa\lambda\epsilon\iota\omega$ will execute all commands contained in the `<input file>` and send the results to the default output device (almost always the screen). *If you use this syntax to call up* $\kappa\lambda\epsilon\iota\omega$*, the program will* not *make any attempt to wait until you are ready before sending each page of output to the screen or other output device.*

If you call up $\kappa\lambda\epsilon\iota\omega$ using the following syntax:

`kleio <input file> <output file>`

the program will execute the commands in the `<input file>` and send the results to the `<output file>`.

The following syntax:

`kleio - <output file>`

instructs $\kappa\lambda\epsilon\iota\omega$ to expect commands from the default input device – usually the keyboard – and send the processed results to the `<output file>`.