Appendix B: The $\kappa\lambda\epsilon\iota\omega$ configuration program

B.1. Introduction

$\kappa\lambda\epsilon\iota\omega$ reads many of the assumptions the program uses from a *configuration file*. This file can be modified by a utility program distributed with the software, which is known as *$\kappa\lambda\epsilon\iota\omega$ configurator*. The configurator in turn expects directives defining what default assumptions $\kappa\lambda\epsilon\iota\omega$ should use, which are prepared as an ASCII file and compiles them into the configuration file $\kappa\lambda\epsilon\iota\omega$ actually uses.

This configuration can be changed by the user; in some cases a behaviour of the program will arise, which for the uniniate may look like a completely different piece of software. Indeed, one of the intentions in providing this facility was the possibility to create end user configurations, which require significantly less understanding of the underlying logic than $\kappa\lambda\epsilon\iota\omega$ as it is distributed.

The person using the configuration facilities, however, is expected to understand the system quite well and to have a good understanding of the concept of environment variables.

The configuration system is currently rapidly expanding: that is, many additional aspects of $\kappa\lambda\epsilon\iota\omega$s behaviour will in the near future become controlled by the logic described here. This appendix does therefore only describe the basic logic of the system: the current state of the implementation can be gotten directly out of the distributed files.

B.2 Inspecting the current configuration

To get a report on the configuration $\kappa\lambda\epsilon\iota\omega$ uses currently on your installation, type the command `showconfig` at the command line. A report on the configuration in the command format discussed in section B.4 will be displayed. `showconfig` is copied by the installation program into the same directory which also holds the executable $\kappa\lambda\epsilon\iota\omega$ program.

B.3 Where does $\kappa\lambda\epsilon\iota\omega$ expect the configuration to be?

When $\kappa\lambda\epsilon\iota\omega$ is called, it looks for a file with the name `kl@@cfg`. The double at-signs shall prevent the user accidentally to create another file with that name. If that file is missing, the program cannot execute. If its contents have been changed in any other way but the one described here, the results are unpredictable.

When installed, $\kappa\lambda\epsilon\iota\omega$ establishes a copy of this file in the directory where the executable program is kept.

When $\kappa\lambda\epsilon\iota\omega$ is called, it tries to find a file with this name according to the following rules:

1) If the directory in which the program is activated contains a file `kl@@cfg` that is used.
2) Otherwise the system checks whether an environment variable `KLEIOCONFIG` has been defined and whether it names a directory, which contains `kl@@cfg`. If both conditions are met that file is used.
3) On UNIX systems only: next the HOME-directory of the user is checked. If that directory contains `kl@@cfg` that file is used.
4) Finally the system checks whether an environment variable `KLEIO` has been defined and whether it names a directory, which contains `kl@@cfg`. If both conditions are met that file is used.
5) If no file `kl@@cfg` has been found the system stops.

This mechanism has been provided to enable installations, where different users and/or different databases can use one version of the software with significantly different types of behaviour.

B.4 Changing the configuration

Before you change the configuration, please make a copy of the current version of the `kl@@cfg` *file. If it gets damaged, κλειω cannot longer be activated.*

To change a configuration file, you use a program called `config` which by the installation process has been placed in the same directory as the executable κλειω program. This program expects to be called with the name of a *configuration script*. A configuration script is an ASCII file containing a series of configuration parameters.

If you specify the name of a configuration script without an extension, `.cfg` will be added to the name. Calling

```
config local
```

directs the κλειω configurator therefore to look for a configuration script in a file named `local.cfg`.

B.4.1 Understanding configuration scripts

When the system is installed on your machine a configuration script `default.cfg` is copied into the directory which contains the executable κλειω program. This file can be inspected with any editor or wordprocessor.

To understand the syntax of the configuration language, we reprint the beginning of the default script, as distributed with versions 5.1.1/6.1.1.

```
# Section Default:
# This section controls various default sizes, characters
# and numerical quantities used by the system
################################################################################
# Subsection Default:Display:
# This subsection controls the sizes used for composing output pages.
################################################################################
# Subsection Default:Display:Printer:
# This subsection controls the sizes used to compose output pages to be sent
# into a file which is afterwards to be printed.  Such files are composed
# when by any means the output is directed to a file instead of the screen.
# ==============================================================================
# Parameter Rows=
# This parameter defines the number of rows a page on the printer is expected
# to have.
# It expects a number as value.
Default:Display:Printer:Rows=62.
# ==============================================================================
# Parameter Columns=
# This parameter defines the number of columns a page on the printer is
# expected to have.
```

```
# It expects a number as value.
Default:Display:Printer:Columns=132.
################################################################################
# Subsection Default:Display:Screen:
# This subsection controls the sizes used to compose output pages to be sent
# to the screen.
# ==============================================================================
# Parameter Rows=
# This parameter defines the number of rows the screen is expected to have.
# It expects a number as value.
Default:Display:Screen:Rows=22.
# ==============================================================================
# Parameter Columns=
# This parameter defines the number of columns the screen is expected to have.
# It expects a number as value.
Default:Display:Screen:Columns=80.
################################################################################
# Subsection Default:Charset:
# This subsection controls the signal characters used by kleio.  Please be
# aware that redifining these characters does not only affect data during
# the reading of raw data, but also the characters used to delimit various
# elements of the command language, like parameters.
# ==============================================================================
# Parameter Dollar=
# This parameter defines data signal character one, defaulting to a dollar sign.
# It expects a single character as value.
Default:Charset:Dollar=$.
# ==============================================================================
# Parameter Slash=
# This parameter defines data signal character two, defaulting to a slash.
# It expects a single character as value.
Default:Charset:Slash=/.
# ==============================================================================
# Parameter Equal=
# This parameter defines data signal character three defaulting to an
# equal sign.
# It expects a single character as value.
Default:Charset:Equal==.
# ==============================================================================
# Parameter Hash=
# This parameter defines data signal character four, defaulting to a hash sign.
# It expects a single character as value.
Default:Charset:Hash=#.
# ==============================================================================
```

```
# Parameter Percent=
# This parameter defines data signal character five, defaulting to a percent
# sign.
# It expects a single character as value.
Default:Charset:Percent=%.
# ===============================================================================
# Parameter Left=
# This parameter defines data signal character six, defaulting to a left
# angular bracket.
# It expects a single character as value.
Default:Charset:Left=<.
# ===============================================================================
# Parameter Right=
# This parameter defines data signal character seven, defaulting to a right
# angular bracket.
# It expects a single character as value.
Default:Charset:Right=>.
# ===============================================================================
# Parameter Semicolon=
# This parameter defines data signal character eight, defaulting to a semicolon.
# It expects a single character as value.
Default:Charset:Semicolon=;.
# ===============================================================================
# Parameter Colon=
# This parameter defines data signal character nine, defaulting to a colon.
# It expects a single character as value.
Default:Charset:Colon=:.
# ===============================================================================
# Parameter Backslash=
# This parameter defines data signal character ten, defaulting to a backslash.
# It expects a single character as value.
Default:Charset:Backslash=\.
# ===============================================================================
# Parameter Question=
# This parameter defines flag character one, defaulting to a question mark.
# It expects a single character as value.
Default:Charset:Question=?.
# ===============================================================================
# Parameter Exclamation=
# This parameter defines flag character two, defaulting to an exclamation mark.
# It expects a single character as value.
Default:Charset:Exclamation=!.
# ===============================================================================
#################################################################################
```

From this file the following rules are probably immediately apparent:

1) A line starting with a hash sign `#` is a comment line.
2) A line starting with anything else is assumed to be a valid parameter definition.
3) Parameter definitions start with a keyword describing the section of the configuration to be modified, followed by a variable length chain of keywords describing subsections, being separated by colons and finally a parameter name ending with an equal sign.
4) After the equal sign a parameter value follows, which has to be specified according to the type of parameter used.
5) Independent of the type of parameter the parameter definition *must* end with a full stop.

B.4.2 Writing configuration scripts

To write your own alternate configuration script, you should copy such parameters as you require into an ASCII file, delete everything that does not need to be changed and modify the parameter values of those parameters, which you want to change. So, if you want to configure your system in such a way, that it assumes your printer to have lines of 80 characters, you would produce a configuration file containing the single line:

```
Default:Display:Printer:Columns=80.
```

B.4.3 Compiling configuration scripts

When you call the `config` program, it will look for a `kl@@cfg` file to modify according to your configuration script. It will use the same logic in searching for that file which has been described in section B.3 above.

You will normally proceed in the following way therefore:

1) Copy the `kl@@cfg` file from the kleio directory into the directory where you want the new one to reside. On a DOS system for example by something like:

```
copy \kleio\bin\kl@@cfg \mydatabase
```

2) Now you go into the new directory and write the shell script, as described in section B.4.2.
3) And compile it by a call like

```
config local
```

4) Now the new configuration will take effect whensoever you call $\kappa\lambda\epsilon\iota\omega$ from the directory `\mydatabase`. *The old configuration will still remain valid, when the system is called from anywhere else.*
5) Check now, whether the new configuration behaves as expected. If this is so,
6) you might copy the new configuration file into the default directory to become activated by all future calls to $\kappa\lambda\epsilon\iota\omega$, for example by:

```
copy \mydatabase\kl@@cfg \kleio\bin
```

When you get any error messages during compilation of your configuration script it is very strongly recommended, that you recopy the original configuration file, as in step 1 above, before you run the corrected shell script again.