Password Gorilla

A password manager coded in Tcl/Tk

Part 1: Application related aspects

- Features
- Screenshots
- Taking over the project „Password Gorilla“
- Maintaining the sources
- Development tools
- Getting Feedback

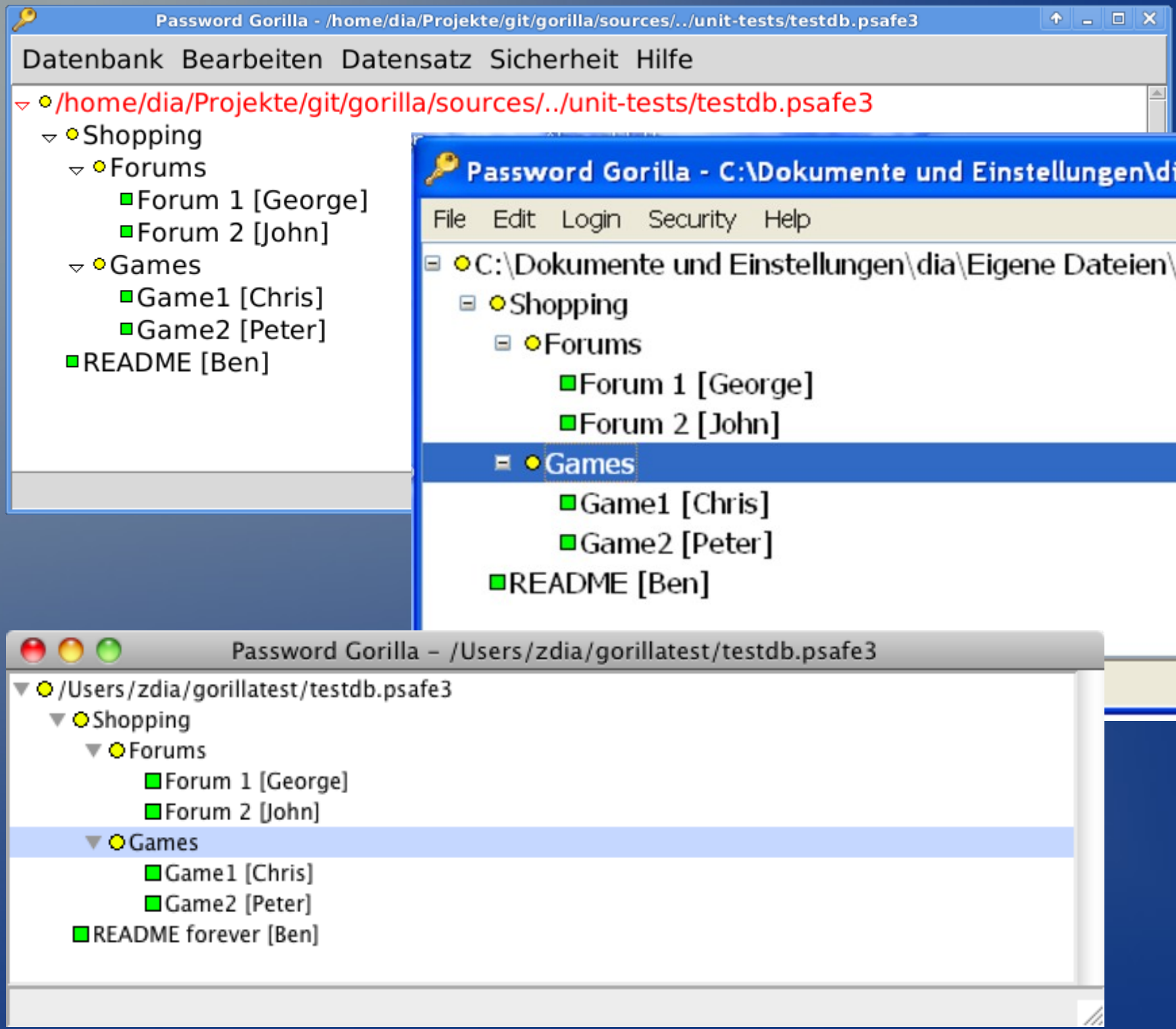
Part 2: Tcl/Tk related aspects

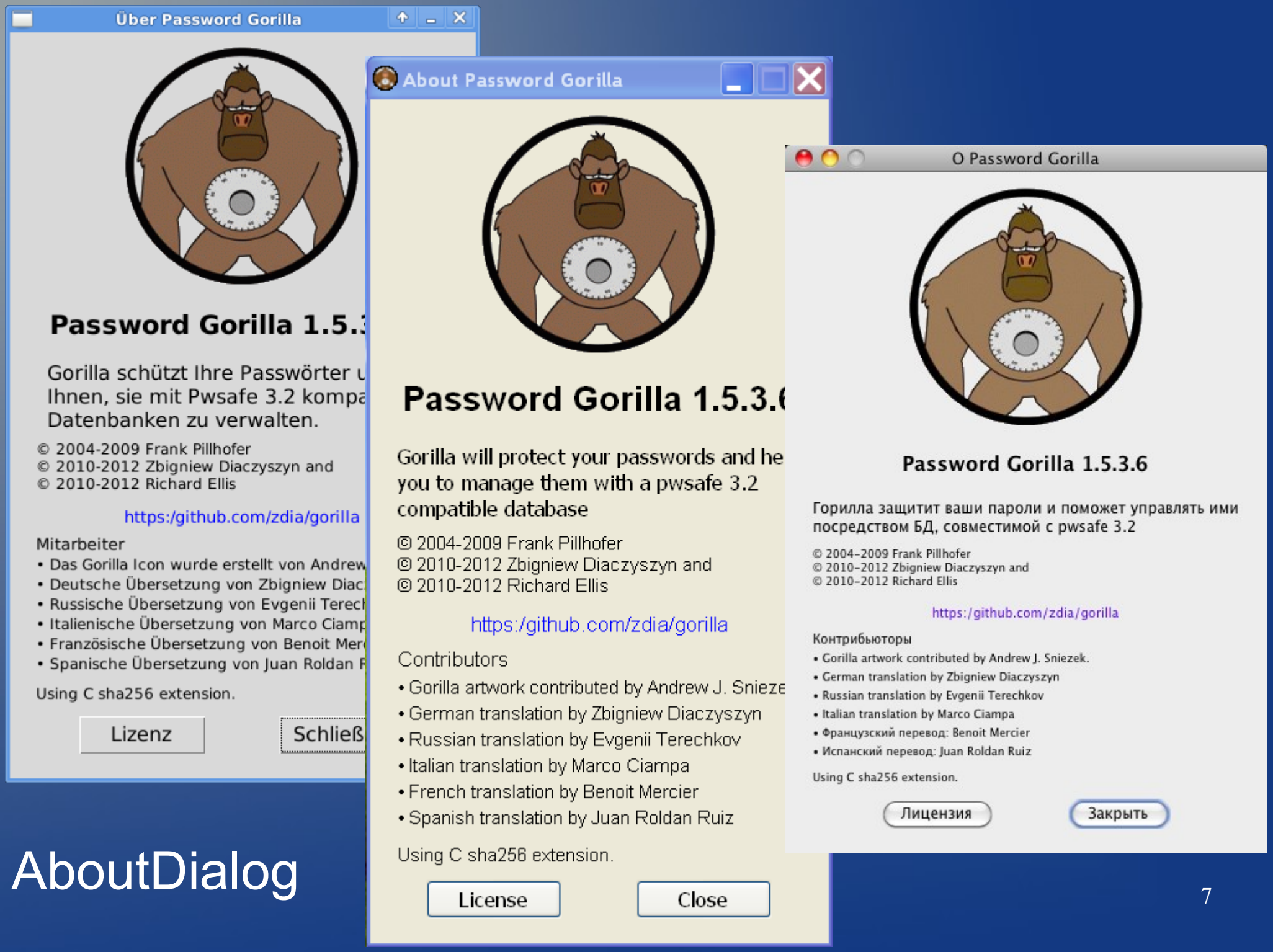
- Localisation with GNU's *gettext* utility
- Keith Vetter's Hypertext Help System
- Preview: The Android port
- Creating executables with tclkits
- Managing the OSX port
- Critcl v3: Speeding up the encryption algorithms
- Unit-tests with *tcltest*
- Documentation with Ashok Nadkarni's *Ruff*

Features

- Cross-platform password manager
- Copy-paste service
- Integrated random password generator
- Cross-platform: Linux, FreeBSD, Windows and MacOS X
- Compatibility to actual Password Safe 3.2 databases
- SHA256 protection for the master password
- Content encryption with Bruce Schneier's Twofish algorithm
- Prevention of brute force attacks by key stretching.
- Integration of a hypertext help system
- Localization support
- Starpack versions for Linux, Windows, OSX
- Dependency for use with sources: Tk 8.5

Part 1: Application related aspects


Password Gorilla - /home/dia/Projekte/git/gorilla/sources/../unit-tests/testdb.psafe3
Datenbank Bearbeiten Datensatz Sicherheit Hilfe
/home/dia/Projekte/git/gorilla/sources/../unit-tests/testdb.psafe3
Shopping
Forums
Forum 1 [George]
Forum 2 [John]
Games
Game1 [Chris]
Game2 [Peter]
README [Ben]
Password Gorilla - C:\Dokumente und Einstellungen\dia\Eigene Dateien\
File Edit Login Security Help
C:\Dokumente und Einstellungen\dia\Eigene Dateien\testdb.psafe3
Shopping
Forums
Forum 1 [George]
Forum 2 [John]
Games
Game1 [Chris]
Game2 [Peter]
README [Ben]
Password Gorilla – /Users/zdia/gorillatest/testdb.psafe3
/Users/zdia/gorillatest/testdb.psafe3
Shopping
Forums
Forum 1 [George]
Forum 2 [John]
Games
Game1 [Chris]
Game2 [Peter]
README forever [Ben]



AboutDialog

Taking over the project

- Frank Pillhofer (2004-2009)
  - Version 1.4.7
  - Itcl, Bwidget
- Zbigniew Diaczyszyn (2010)
  - Version 1.5.x
  - Bwidget replaced by Ttk widgets
  - msgcat mechanism added
  - Translation into German language
- Richard Ellis (2010)
  - Improving constantly code, adding various modules

Maintaining the sources

- Git
  - Encourages cloning
  - Fully fledged local repository
  - Simple and effective local branches
  - GUI programs: gitk, „git gui“
- Github Server
  - Download area
  - Wiki (Textile)
  - Automatic code tarballs
  - Social coding
  - Issue management

Development Tools

- Texteditor *Geany*
  - Syntax highlighting (Tcl/Tk)
  - Code folding
  - Symbol name auto-completion
  - Construct completion/snippets
  - Symbol lists (for code navigation)
  - Simple project management
  - Plugin interface (see Plugins)
- Tkcon
  - Console access to live running PWGorilla process
  - *Edit* command to inspect/change procedures or variables
  - Source and change the running system in real time

Statistics: platform dependency

Total PWGorilla lines of code: 9,088

out of the above:

Linux specific lines:	5	0.05% of total	(1)
Windows specific lines :	12	0.13% of total	(2)
MacOSX specific lines:	109	1.19% of total	(3)
Total platform specific:	126	**1.39%** of total	

(1) X11 clipboard selection
(2) alternative for /dev/urandom; samba mount bug; clipboard select;
(3) scrollbar, menubar, autopath, menu shortcuts, About menu, mouse-button, event loop issues, path issues, file preselection, open browser

Getting feedback

Editor's Opinion

Password Gorilla - a powerful and highly efficient way in which you can securely store all of your login credentials. This versatile and easy to use piece of software is an excellent tool for any computer user that wants to secure all of their accounts.

Part 2: Tcl/Tk related aspects

Localization with *gettext*

Source code format:

```
puts [ mc "Need %s" $file ]
```

Create a Portable Object Template:

```
xgettext -kmc -o gorilla.pot -L Tcl gorilla.tcl ?...?
```

Result in file *gorilla.pot*:

```
#: ../../sources/gorilla.tcl:193
#, tcl-format
msgid "Need %s"
msgstr ""
```

Localization with *gettext*

```
puts [ mc "Need %s" $file ]
```

source code

```
xgettext -kmc -o gorilla.pot -L Tcl gorilla.tcl
```

```
msgid "Need %s"
msgstr ""
```

gorilla.pot

Gettext: Creating locale .po file

```
msgid "Need %s"
msgstr ""
```

gorilla.pot

Text editor

```
msgid "Need %s"
msgstr "Benötige %s"
```

de.po

Gettext: Getting de.msg file

Merge existing <lang>.po with new *gorilla.pot*:

msgmerge --update <lang>.po --backup=simple gorilla.pot"

Create a Tcl *.msg language file:

msgfmt --tcl -l<lang> -d <path-to-msgs-dir> <lang>.po

(lang = en, de, fr, ru ...)

Result in file *de.msg*:

::msgcat::mcset de "Need %s" "Ben\u00f6tige %s"
::msgcat::mcset de "File" "Datenbank"
...

Gettext: Getting de.msg file

```
msgid "Need %s"
msgstr "Benötige %s"
```

de.po

msgfmt --tcl -lde -d <path-to-msgs-dir> de.po

```
::msgcat::mcset de "Need %s" "Benötige %s"
::msgcat::mcset de "File" "Datenbank"
...
```

de.msg

Gettext: Tweaking .msg files

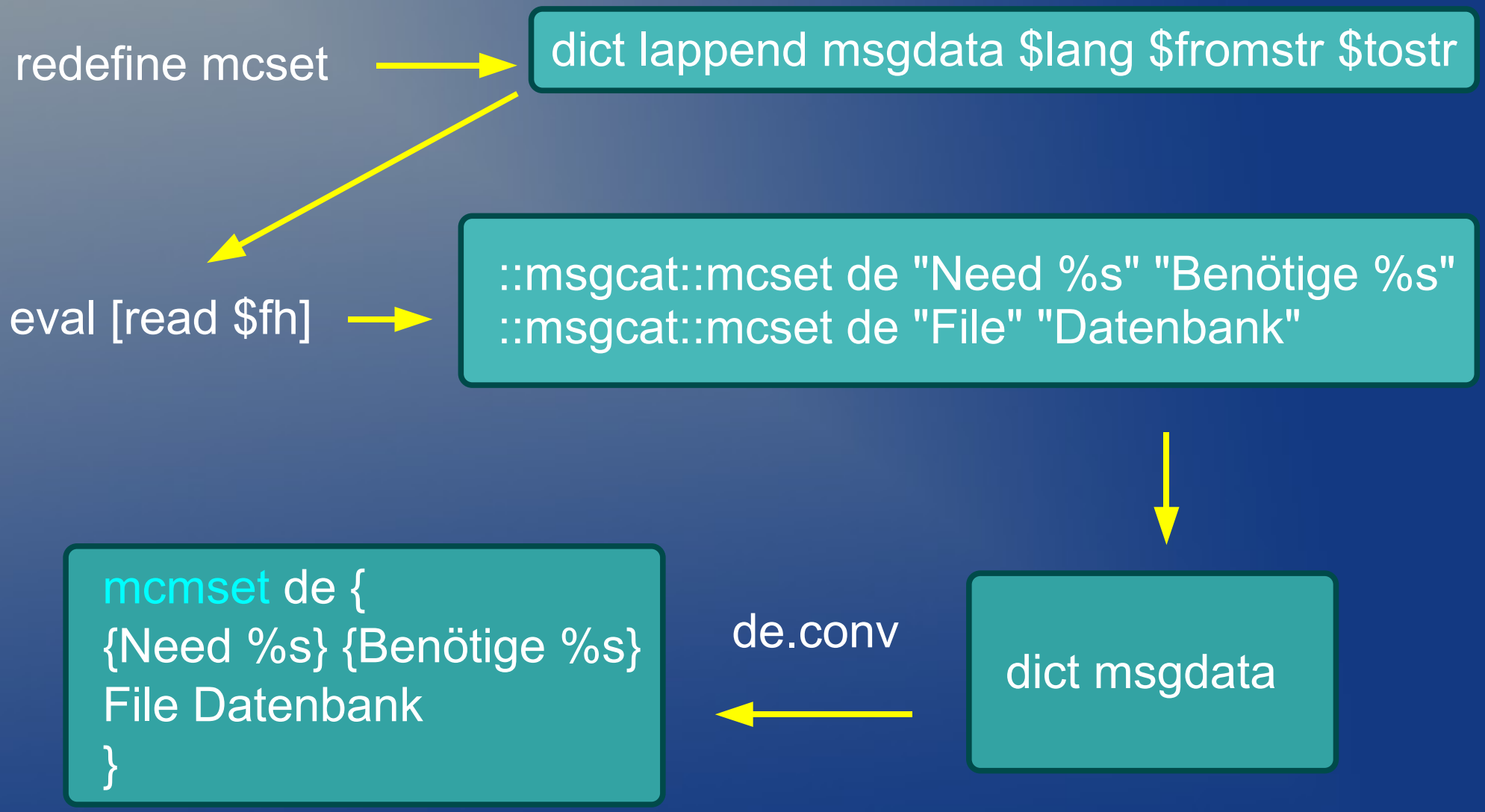
Redefine *mcset*:

```
proc mcset { lang fromstr tostr } {
    variable msgdata
    dict lappend msgdata $lang $fromstr $tostr
  }
```

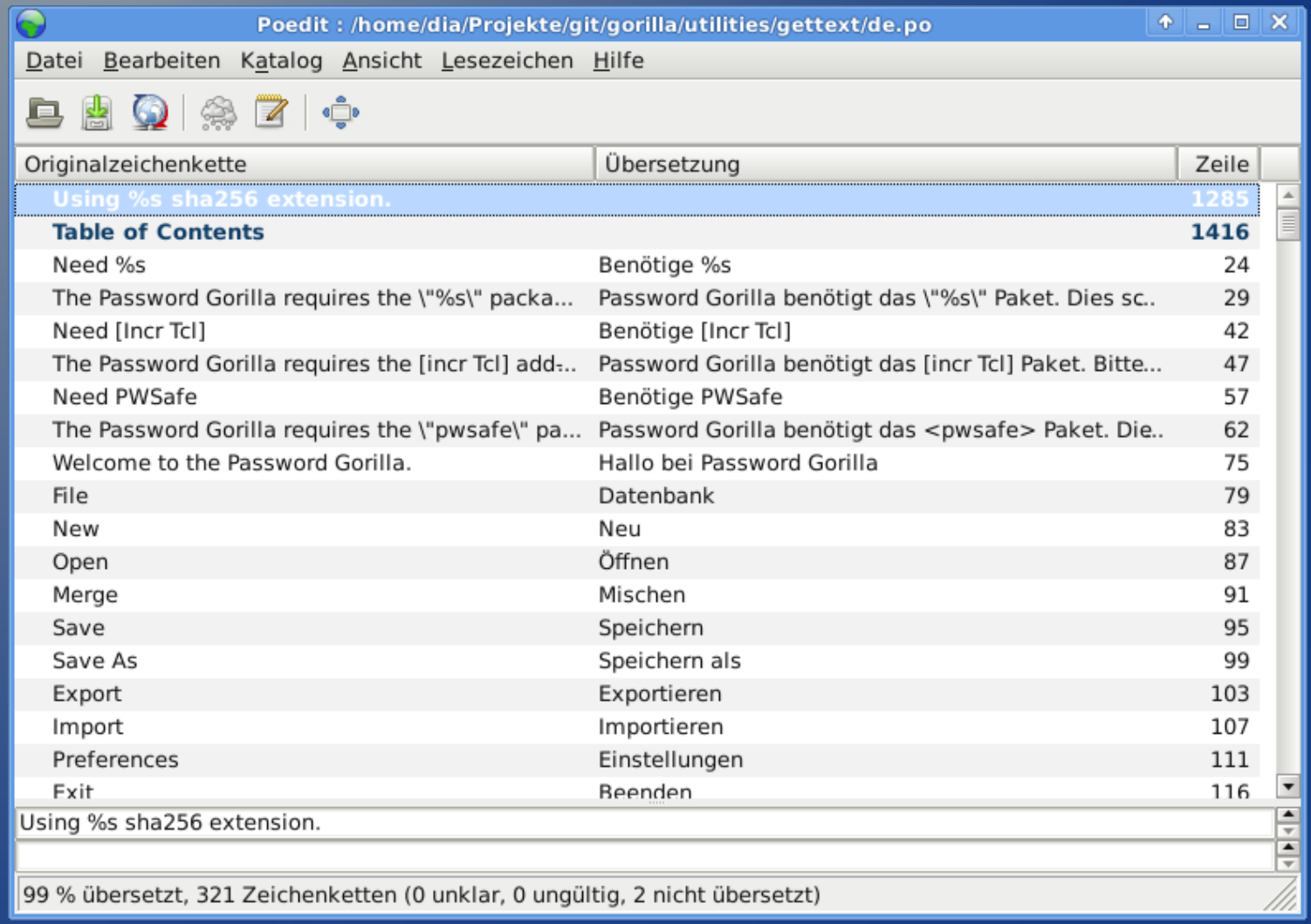
Use the created dictionary to build the final *de.msg:*

```
mcmset de {
{Need %s} {Benötige %s}
File Datenbank
...
}
```

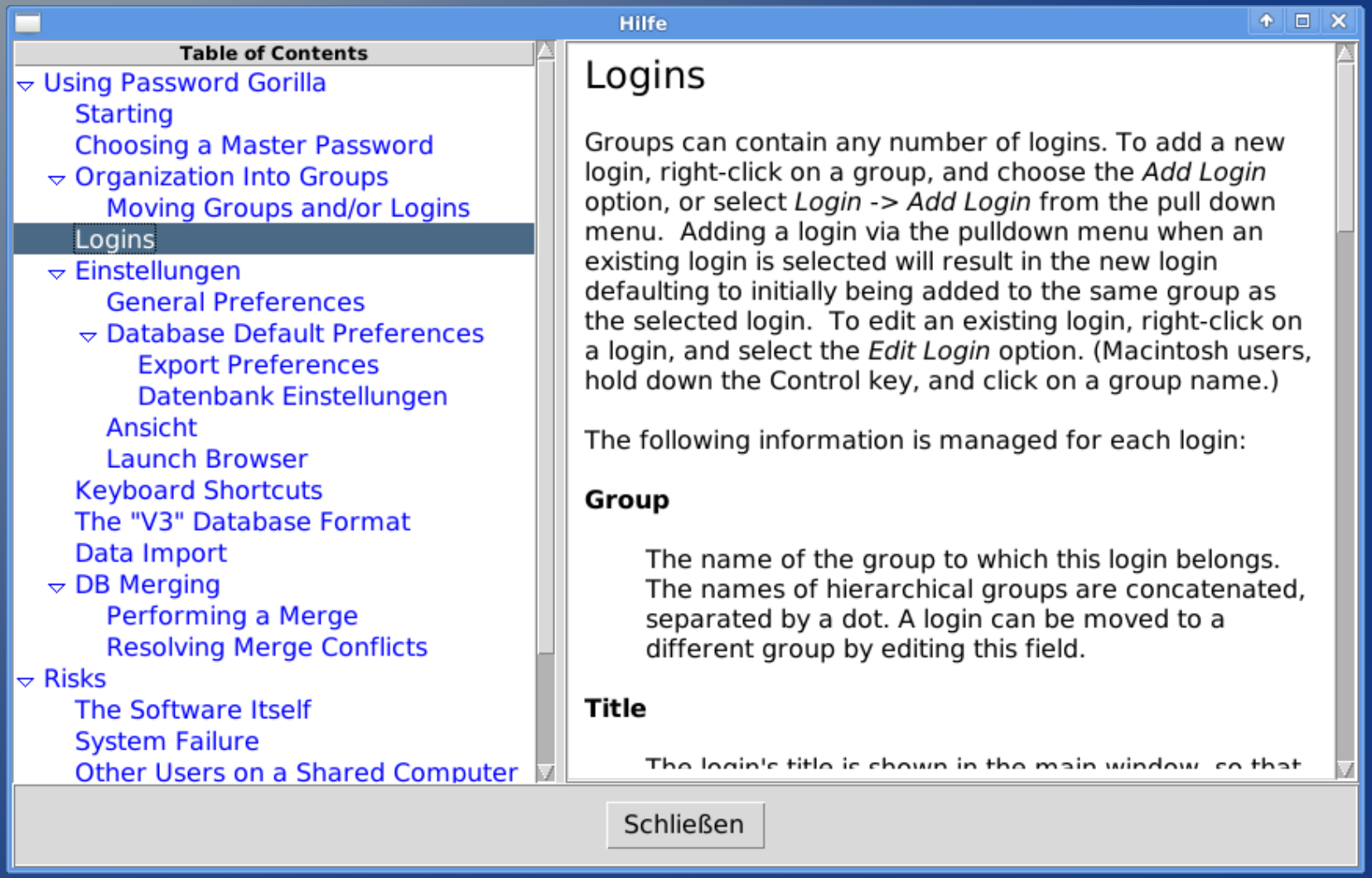
Tweaking .msg files



Gettext: Editor *poedit*



The Hypertext Help System



Help system text example

---
title: Logins

Groups can contain any number of logins. To add a new login, right-click on a group, and choose the "Add Login" option, or select "Login" -> "Add Login" from the pull down menu. [...]

The following information is managed for each login:

'''Group'''

 | The name of the group to which this login belongs. The names of hierarchical groups are concatenated, separated by a dot. A login can be moved to a different group by editing this field.

Using The Help System

```
source viewhelp.tcl

proc gorilla::Help {} {
    ::Help::ReadHelpFiles $::gorillaDir $::gorilla::preference(lang)
    ::Help::Help Overview
}
```

Tcltest: Final Test Run

```
$ gorilla --tcltest
++++ csv-import.test-1.2 PASSED
++++ csv-import.test-1.3 PASSED
++++ csv-import.test-1.4 PASSED
...
++++ csv-import.test-1.13 PASSED
++++ csv-import.test-1.1 PASSED
++++ csv-import.test-2.1 PASSED
++++ csv-import.test-2.2 PASSED
++++ csv-import.test-2.3 PASSED
++++ csv-export.test-1.0 PASSED
++++ lock-database-1.1 PASSED
++++ backup-1.1 PASSED
++++ backup-1.2 PASSED
RunAllTests.tcl:  Total   20  Passed    20  Skipped   0    Failed 0
```

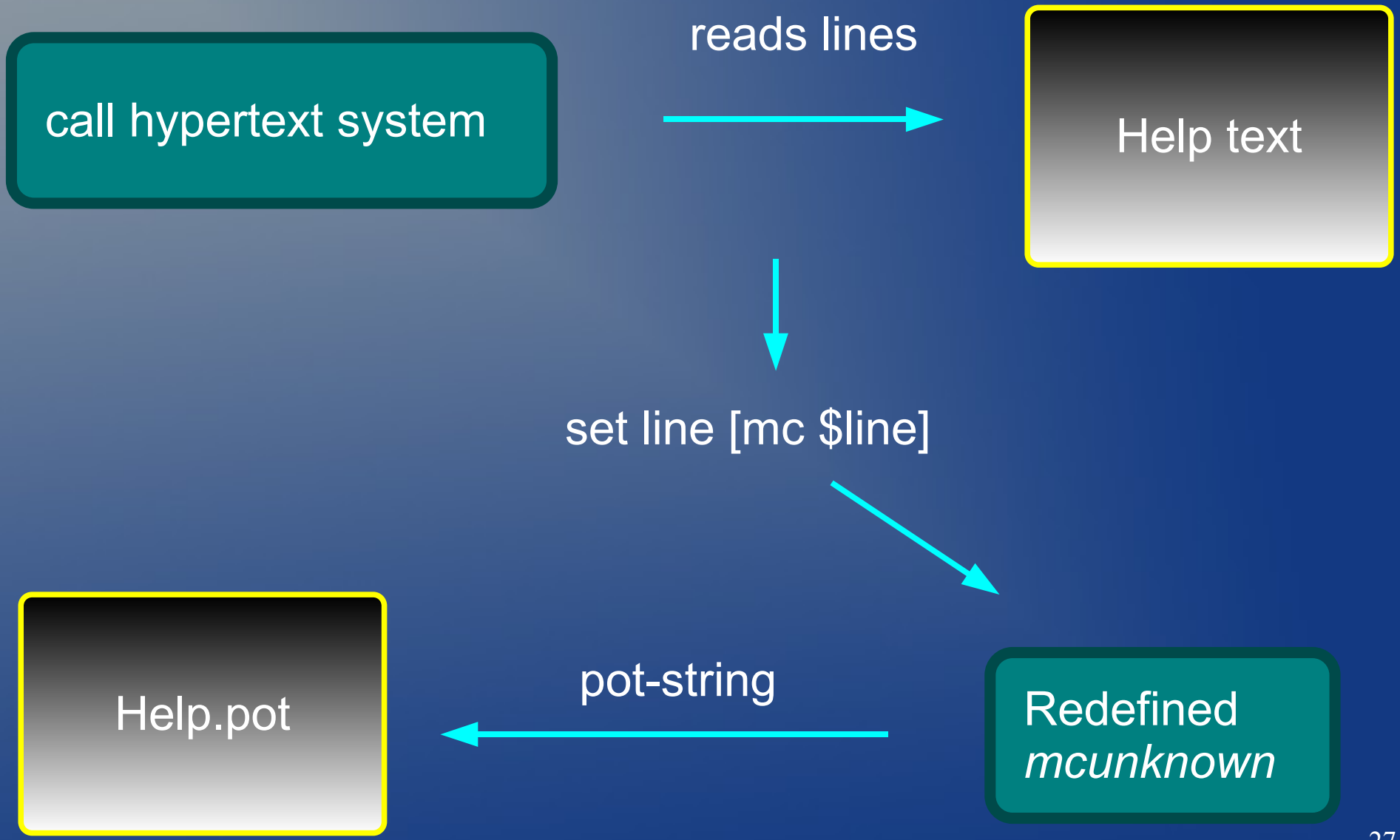
Modifying The Help System

It is based on a treeview and a text widget

Example for adding an indented paragraph with mark „|“:

```
$w.t tag config bar -lmargin1 $l2 -lmargin2 $l2
...
if { $op1 eq "|" } {          ; # Bar
        set tag bar
}
```

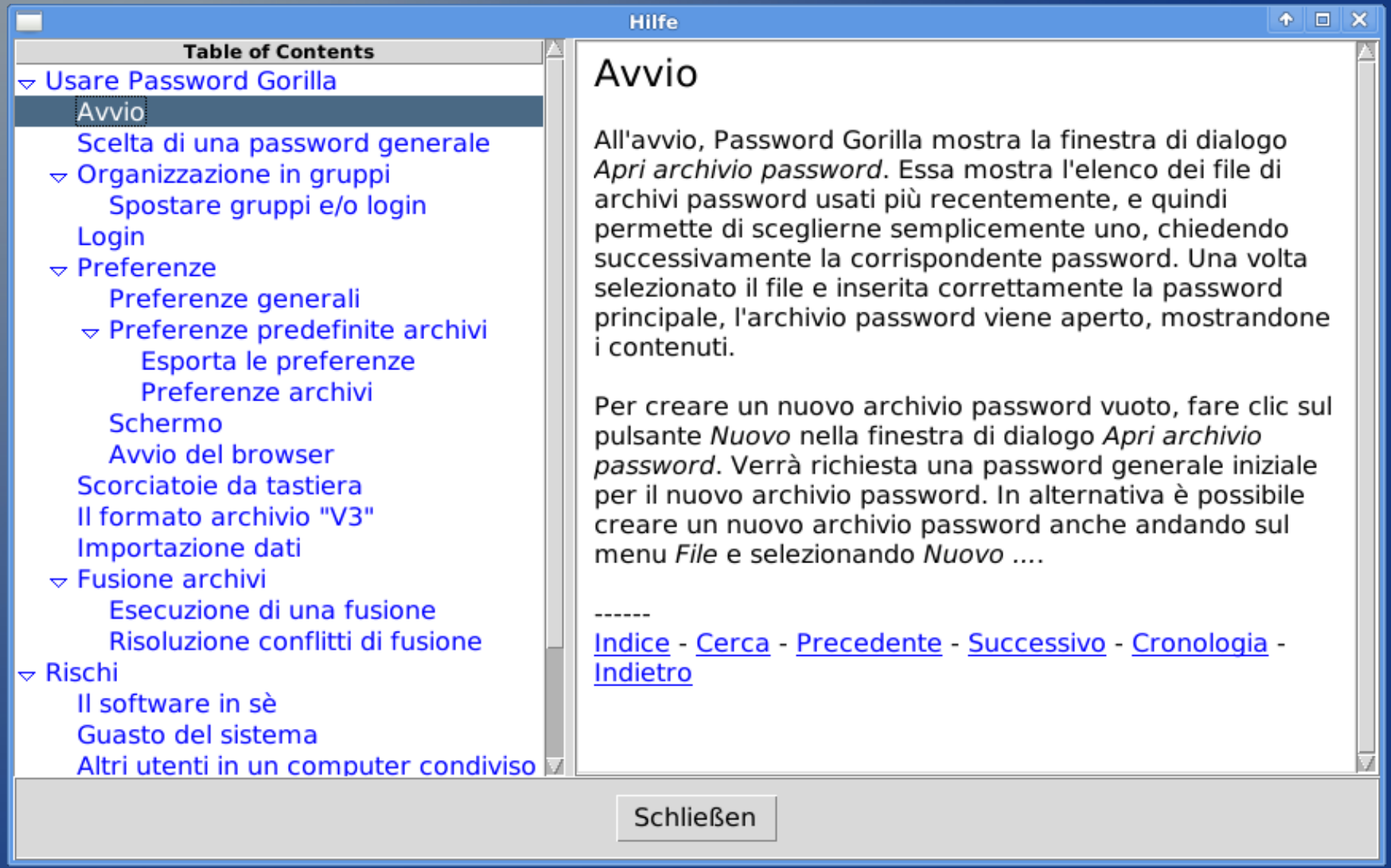
Hypertext: Creating Help.pot



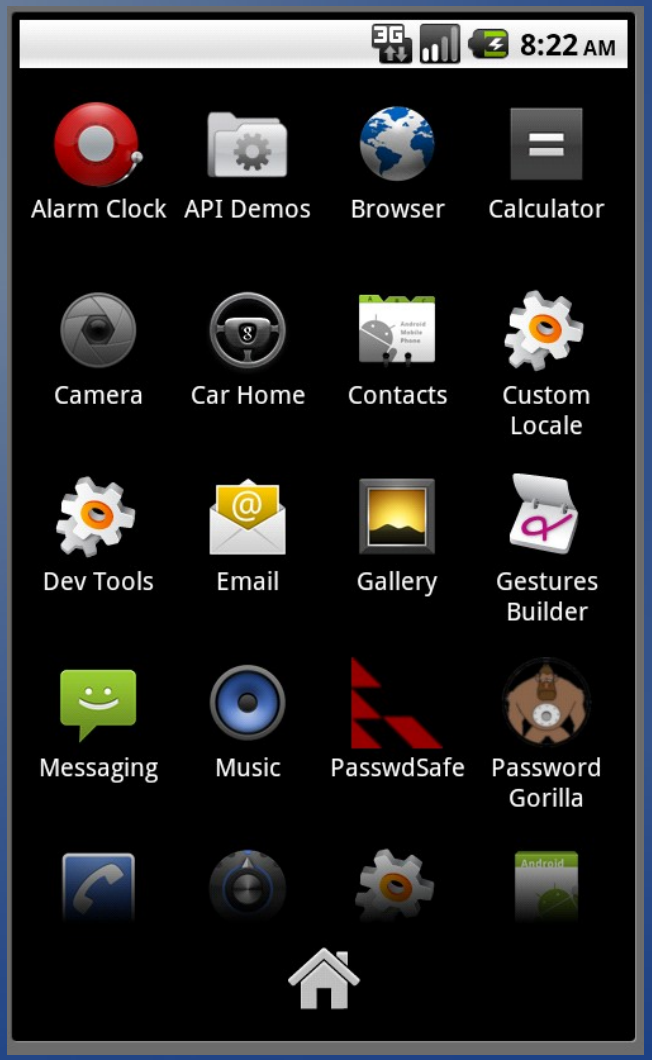
Hypertext and gettext: mcunknown

```
proc ::msgcat::mcunknown {locale src_string} {

    global output

    set poStr "msgid \"$src_string\"\nmsgstr \"\"\n"

    # msgid $src_string
    # msgstr ""

    if { $src_string ne "" } { puts $output $poStr }

      return $src_string
}
```

Gettext: Italian Help Text

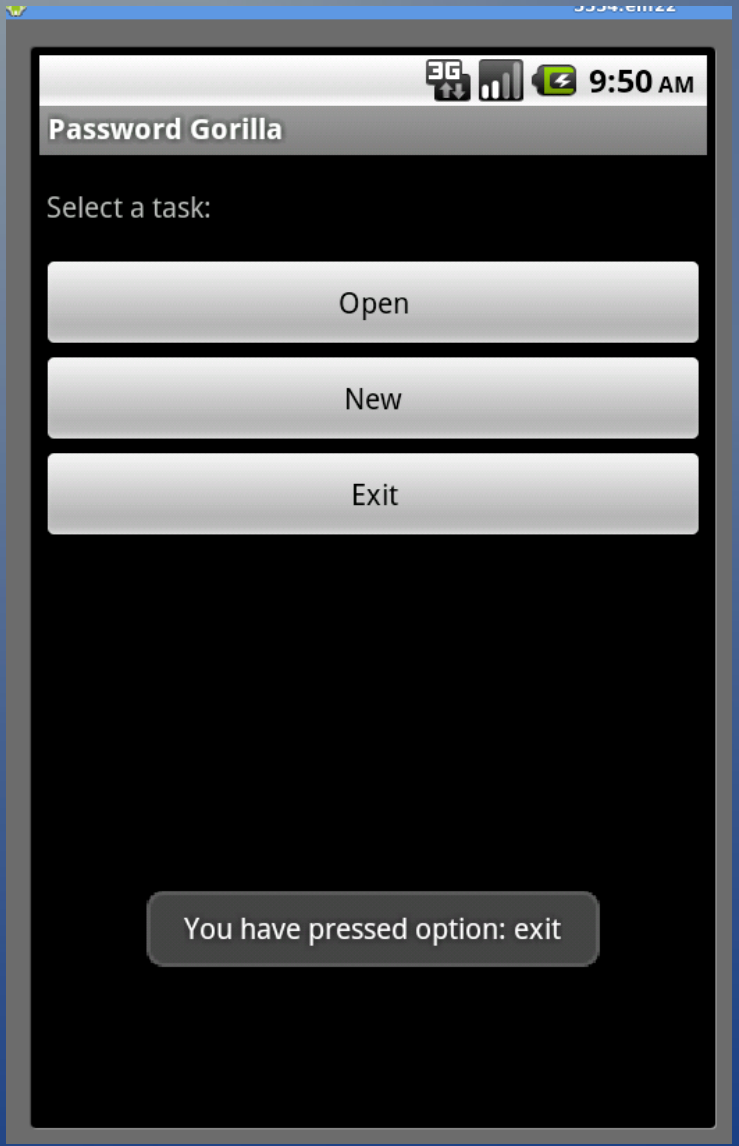


Android: Launch

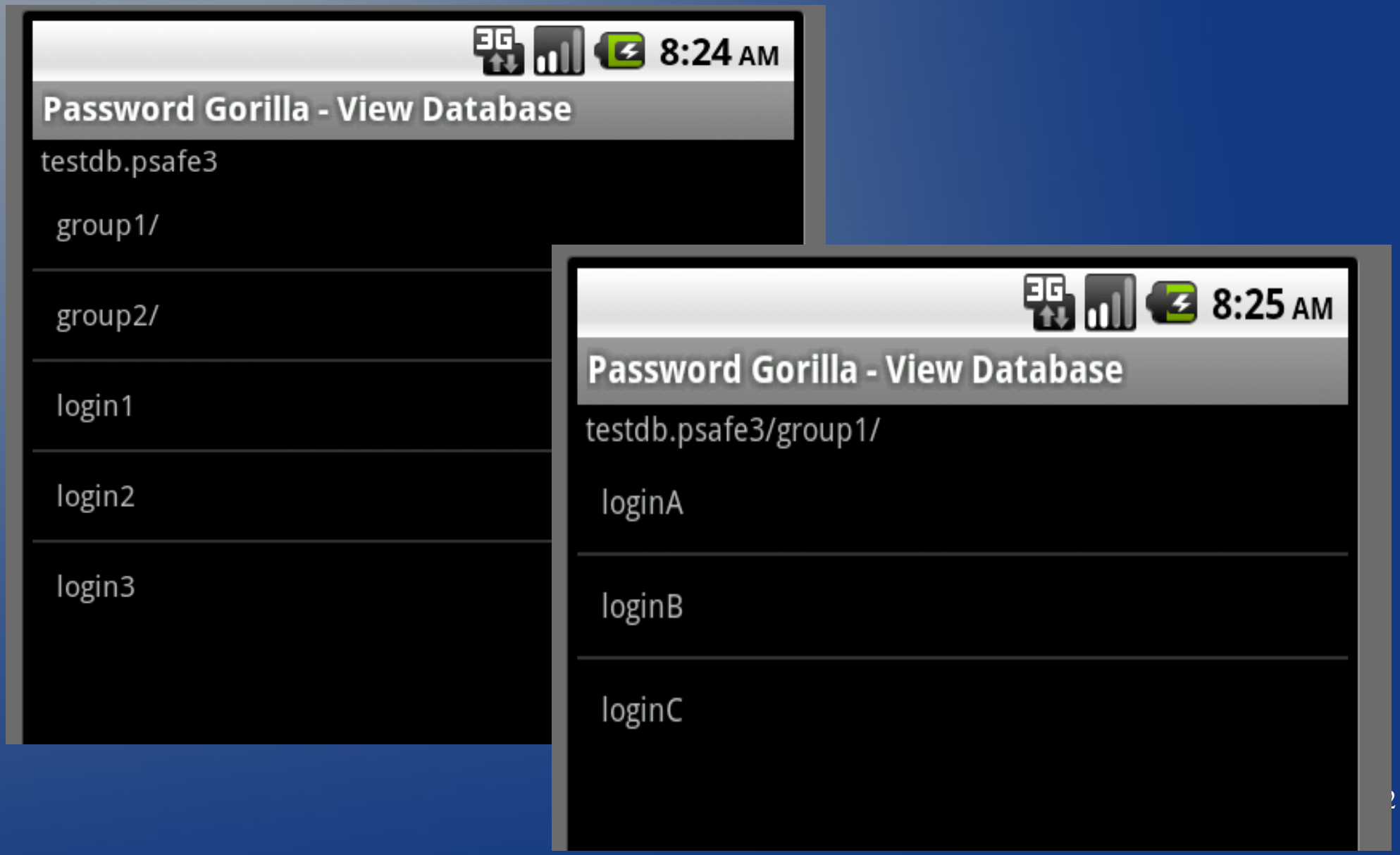


https://github.com/zdia/gorilla/wiki/Gorilla-for-Android

Android: OpenDialog and FileDialog



Android: Content Listing



Hecl: Tcl-like Syntax

Math operations in Polish notation:

```
puts "2 + 2 = [+ 2 2]"
if { < $temp 0 } {   puts "It's freezing" }
```

Command *hash (*instead of array or dict)

```
set foo [hash {a b c d}]
puts [hget $foo a]    ;# prints 'b'
hset $foo c 2
puts [hget $foo c]    ;# prints '2'
```

Hecl: GUI Code Example

```
set context [activity]
set layout [linearlayout -new $context]
$layout setorientation VERTICAL
set layoutparams [linearlayoutparams -new { \
    FILL_PARENT WRAP_CONTENT}]

set tv [textview -new $context -text {Hello World} \
            -layoutparams $layoutparams]

$layout addview $tv
[activity] setcontentview $layout
```

Hecl: The command *java*

```
hecl> java java.lang.Integer integer
Integer

# System.out.println( Integer.toHexString(2048) );

hecl> set hex [integer toHexString 2048]
800

# int decimalNumber = Integer.parseInt(800, 16);

hecl> integer parseInt "800" 16
2048
```

End of Presentation for EuroTcl 2012

Creating executables (1)

- Virtual Filesystem gorilla.vfs

```
gorilla.vfs/
|-- lib/
|   `-- app-gorilla/
|-- main.tcl
`-- tclkit.ico*
```

app-gorilla/ is linked to:

```
/home/dia/Projekte/git/gorilla/sources/
```

Creating executables (2)

- Content of main.tcl

```
package require starkit
starkit::startup
package require app-gorilla 1.0
```

- First line in application:

```
package provide app-gorilla 1.0
```

Creating executables (3)

- Final creation command:

```
./tclkit sdx.kit wrap gorilla.exe -runtime <path/tclkit-version>
```

- Actual runtime versions:

```
tclkit-8.5.11-tk-freebsd-ix86
tclkit-8.5.11-linux
tclkit-tk-8.5.11-linux-x86_64
tclkit-8.5.11-win32.exe
tclkit-tk-8.5.11-win32-x86_64
tclkit-8.5.11-macosx-universal
```

The OSX port : Bundle Structure

The application bundle *gorilla.zip*

```
|-- Password Gorilla.app
|   `-- Contents
|       |-- Info.plist
|       |-- MacOS
|       |   `-- gorilla.aqua
|       `-- Resources
|           `-- Gorilla.icns
|
`-- gorilla.zip
```

The OSX port: The OSX Menus

```
If { [tk windowingsystem] == "aqua"} {
    # we have to delete the psn_nr in argv
    if {[string first "-psn" [lindex $argv 0]] == 0} {
        set argv [lrange $argv 1 end]}
    }
    proc ::tk::mac::ShowPreferences {} {
            gorilla::PreferencesDialog
    }
    proc ::tk::mac::Quit {} {
        gorilla::Exit
    }
    proc tk::mac::ShowHelp {} {
        gorilla::Help
    }
}
```

The OSX port: The About Dialog

Enable the event:

```
proc tkAboutDialog {} {
    gorilla::About
}
```

Manage the menu entry:

```
menu .mbar.apple
.mbar add cascade -menu .mbar.apple
.mbar.apple add command -label "[mc "About"] Password Gorilla" \
    -command gorilla::About
```

The OSX port: The File Dialog

Native fileselect dialog does not allow filtering of files:

```
tk_getOpenFile -filetypes [list ] -initialdir $::gorilla::dirName
```

C Extensions with CriTcl

Get CriTcl version 3 (Compiled Runtime In Tcl):

```
$ git clone https://github.com/jcw/critcl.git
```

Build critcl starpack:

```
$ ./build.tcl starpack prefix ?destination?
```

Use it to build extension package *sha256c*:

```
$ critcl -pkg sha256c.tcl
```

Get help:

https://github.com/jcw/critcl/blob/master/embedded/www/toc.htm

Directory structure for CriTcl created libraries:

```
sha256c/
|-- freebsd-ix86/
|   `-- sha256c.so*
|-- linux-ix86/
|   `-- sha256c.so*
|-- linux-x86_64/
|   `-- sha256c.so*
|-- macosx-ix86/
|   `-- sha256c.dylib*
|-- macosx-x86_64/
|   `-- sha256c.dylib*
|-- win32-ix86/
|   `-- sha256c.dll*
|-- win32-x86_64/
|   `-- sha256c.dll*
|-- critcl.tcl
`-- pkgIndex.tcl
```

Documentation Generator *Ruff!*:

- Ruff! (Runtime function formatter) v0.4
- Author: (c) Ashok P. Nadkarni
- http://woof.magicsplat.com/ruff_home
- Written in Tcl
- Generation by runtime introspection
- Comment analysis

Ruff!: Create Gorilla documentation

Initialization in source code for option --*sourcedoc*:

```
package require ruff
set nslist [ namespace children :: ]
::ruff::document_namespaces html $nslist \
    -output gorilladoc.html \
    -recurse true
```

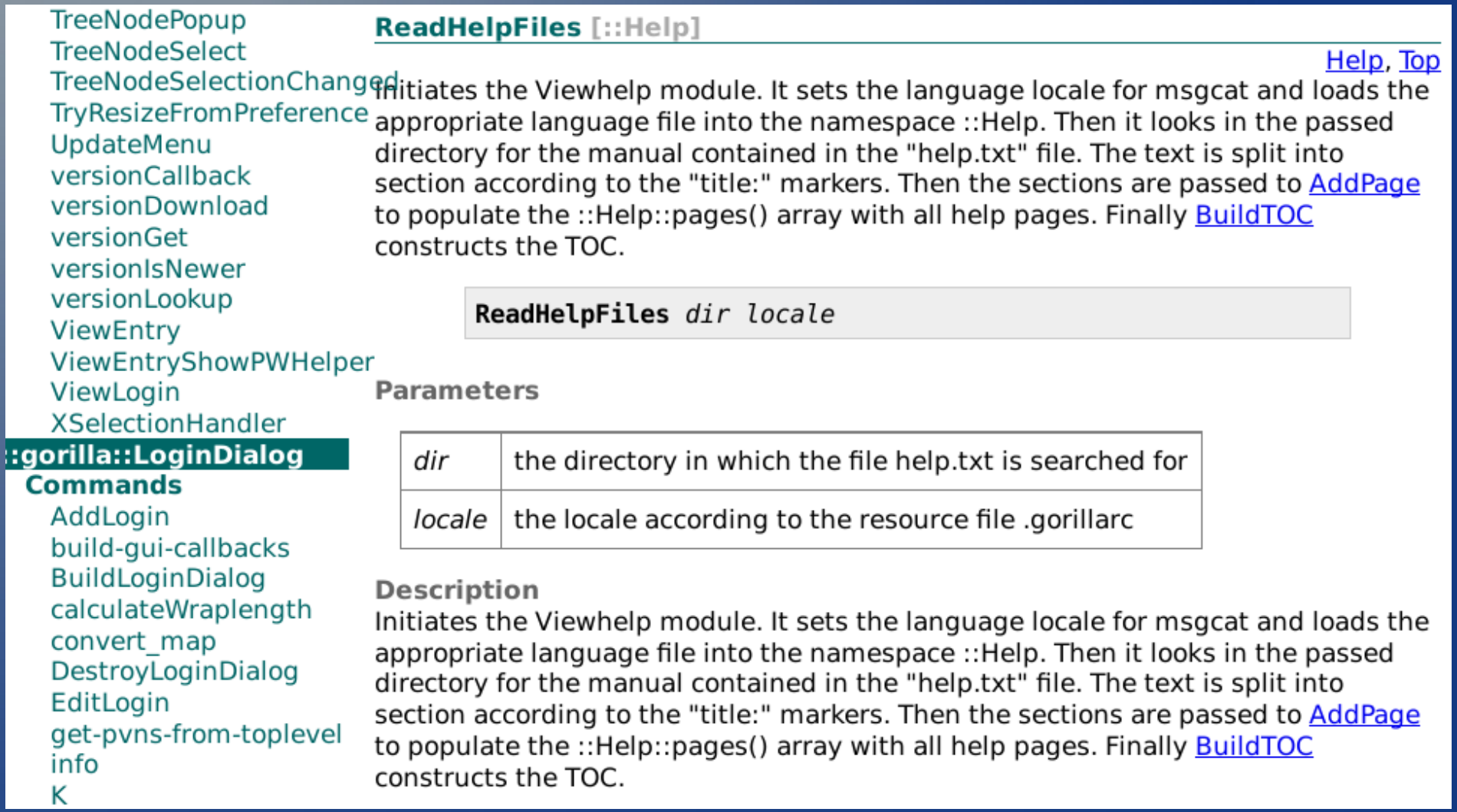
Runtime generation:

```
gorilla --sourcedoc
```

Ruff!: Comment Analysis (Sample)

```
proc ::Help::ReadHelpFiles {dir locale} {
    # Initiates the Viewhelp module.
    # It sets the language locale for msgcat and loads the appropriate
    # language file into the namespace ::Help. [...]
    #
    # dir - the directory in which the file help.txt is searched for
    # locale - the locale according to the resource file .gorillarc
    #
```

Ruff!: Html Output



Tcltest: source code integration

Switch of commandline option:

```
--tcltest {
    # TCLTEST 1 and TEST 1 means:
    # skip the OpenDatabase dialog and load testdb.psafe3
    array set ::DEBUG { TCLTEST 1 TEST 1 }
    }
```

After general and GUI initialization:

```
if { $::DEBUG(TCLTEST) } {
    set argv ""
    source [file join $::gorillaDir .. unit-tests RunAllTests.tcl]
}
```

Tcltest: Directory structure

```
unit-tests
|-- RunAllTests.tcl
|-- backup
|   `-- backup.test
|-- csv-export
|   |-- csv-export.test
|   `-- normexport.csv
|-- csv-import
|   |-- csv-import.test
|   |-- import11.csv
|   |-- import110.csv
|   |-- import17.csv
|-- lock-database
|   `-- lock.test
`-- testdb.psafe3
```

Tcltest: RunAllTests.tcl

```
package require tcltest 2.2

...
# default search path is the actual working directory
tcltest::workingDirectory [file dirname [file normalize [info script]]]
tcltest::singleProcess 1      ;# caller environment will be used
tcltest::verbose { pass }
...
foreach testFile $testList {
    source $testFile
}
...
tcltest::cleanupTests     ;# will give the results and exit
```

Tcltest: A Single Test

```
test $testname-1.2 {File Open Error} \
    -setup { set ::DEBUG(CSVIMPORT) 1 } \
    -body { gorilla::Import unknown.csv } \
    -cleanup { set ::DEBUG(CSVIMPORT) 0 } \
    -result GORILLA_OPENERROR
```

Help system features

- A help system originally based on Tcl Wiki 1194 and Tile
- Author: Keith Vetter (May 2007)
- Hyperlinks to other help pages
- Simple searching ability
- History
- Simple wiki formatting:
  - Nested numeric, bullet and dash lists
  - Bold, italic and unformatted text
- Table of Contents
- Adapted for PWGorilla with msgcat support and Ttk widgets