BcdaMenu Documentation

Release 0.g10e1dc1.dirty

Pete Jemian

Sep 18, 2017

Contents

Creates a GUI menu button to start common beam line software

CHAPTER 1

Provides

- **bcdamenu** : the button menu

CHAPTER 2

Package Information

- **author** Pete R. Jemian
- **email** jemian@anl.gov
- **copyright** 2017-, Pete R. Jemian
- **license** ANL OPEN SOURCE LICENSE (see LICENSE.txt file)
- **documentation** http://bcdamenu.readthedocs.io
- **source** https://github.com/BCDA-APS/bcdamenu
- **PyPI** https://pypi.python.org/pypi/bcdamenu
- **version** 2017.9.0
- **release** 0.g10e1dc1.dirty
- **published** Sep 18, 2017

2.1 Usage

typical

```
user@linux > bcdamenu path/to/settings_file.ini &
```

bash starter file

```
#!/bin/bash
bcdamenu path/to/settings_file.ini &
```

usage

```
user@linux > bcdamenu
usage: BcdaMenu [-h] settingsfile
BcdaMenu: error: too few arguments
```

help

```
user@linux > bcdamenu -h
usage: BcdaMenu [-h] settingsfile

BcdaMenu: Creates a GUI menu button to start common beam line software

positional arguments:
  settingsfile  Settings file (.ini)

optional arguments:
  -h, --help    show this help message and exit
```

2.2 Contents

2.2.1 Settings File

The **BcdaMenu** GUI is configured by the content provided in a settings file which name is specified on the command line when starting the program. For example, this Linux command:

```
bcdamenu path/to/menus_settings.ini &
```

Version "2017.3.0" format of the Settings file

The settings file version *2017.3.0* uses the *.ini* format, a structure similar to what you would find in Microsoft Windows INI files. This format was chosen for its minimal approach to language markup. The examples provided should guide you to the syntax. For more details, see the documentation for the Python ConfigParser. The web has many explanations of this informal format.

Settings file elements

The settings file consists of sections which are lines starting with "*[*" and ending with "*]*", such as *[section_name]*. In **BcdaMenu**, these sections are single words with no embedded white space.

Within a section, one or more lines are given with the syntax of *key = value* (or *key: value*). It is expected by the *.ini* format that any key is unique *within* its section. In **BcdaMenu**, the *key* has two parts. First an integer is given that is used to sort the menu's items in order'. The integer is not required to be strictly increasing from 1. Gaps and negative numbers are also allowed. Keep the integers between -9999 .. 9999 to avoid any potential misunderstandings. You *will not* have that many menu items.

Refer to the *Example settings file* section for an example settings file. As the examples show, both *key* and *value* are quite flexible strings. A *key* should not contain either the "*:*" or "=" separator characters. The comment characters allowed by the *.ini* format should also be avoided within either *key* or *value* content.

{BcdaMenu] section This section expects the following keys and values. Other keys and values will be ignored.

title The window title (default: *BCDA Menu*)

version The version of the settings file format. Presently, the only allowed value is *2017.3.0*, the settings format of the initial release. If this format ever changes, this key will be used to identify how to handle the different syntax of the settings file of the new version.

menus Names of the sections below with menu specifications. Its value may have more than one menu name, separated by white space.

menu sections As referenced by *menus* or *submenu* keys. Each menu section must have a one-word name with no internal white space (to simplify the parsing of names in the *[BcdaMenu]* section. All menu (and submenu) sections must be unique with the settings file. If the same name is used in more than one section, a *configparser.DuplicateSectionError* exception will be raised.

other sections will be ignored

Menu Sections

Each popup menu is configured by the keys and values of a menu section.

title Use this as the title on the popup button or submenu name. If the *title* key is not present, the name of the menu section will be used as the text of the popup button.

separator draws a horizontal separator line in the menu

numbered lines specification of a menu item

- syntax: *index label = command*

where:

index an integer used to sort the list numerically (increasing)

label text that appears in the menu

- *title* is reserved word and cannot be used as a menu item label
- *separator* is reserved word and cannot be used as a menu item label
- *submenu* Specifies that a submenu will be inserted at this position in the menu. For *submenu*, the *command* gives the name of the menu section to be used.

  NOTE Do not use the same submenu more than once. This will raise a *ConfigFileKeyError* exception when reading the settings file.

command Operating system command to be executed when menu item is chosen. This is a complete command in the operating system syntax that will be started as a separate task by **BcdaMenu**.

Linux: There is no need to append ' &' to run the commands in the background.

Shortcut keys

Shortcut keys are not supported for any menu items.

Example Settings File

The settings file in the source code distribution (`download`, also shown below) is an example demonstrating the various features used by **BcdaMenu**.

file

The example settings file (highlighted lines show the sections, lines 1, 6, & 19 and the specification of the popup menus, line 4) is shown next.

```
1  [BcdaMenu]
2  title = BcdaMenu: 9-ID-C USAXS
3  version = 2017.3.0
4  menus = USAXS linux
5
6  [USAXS]
7  title = 9-ID-C USAXS
8  1 SAXS Imaging tool = /APSshare/epd/rh6-x86/bin/python /home/beams/USAXS/Apps/USAXS_
   ↪dataworker/Main.py
9  2 sample and detector XY position tool = wxmtxy.csh
10 3 separator =
11 4 USAXS Q calculator = qToolUsaxs.csh
12 5 9-ID-C USAXS controls (MEDM) = start_epics
13 6 Save Instr. status to Elog = saveToElog.csh
14 7 PyMca = /APSshare/bin/pymca
15 8 USAXS sample stage tool = /home/beams/USAXS/Apps/wxmtusaxs/wxmtusaxs
16 9 separator =
17 10 submenu = example_submenu
18
19 [linux]
20 1 Xload = xload
21 2 FireFox = firefox
22
23 [example_submenu]
24 title = this is an example of a submenu
25 1 comment = echo "this is not a comment"
26 # this is a comment
```

screens

This settings file produces a GUI titled *9-ID-C USAXS menu* with two user menus: *USAXS* and *linux*. The following screen views are from Linux.

Fig. 2.1: GUI using example settings file.

This is the *USAXS* menu:

This is the *linux* menu:

The "Help" popup menu button

The *Help* popup menu button is controlled by the program and is not configurable by the user through the settings file.

This is the *Help* menu:

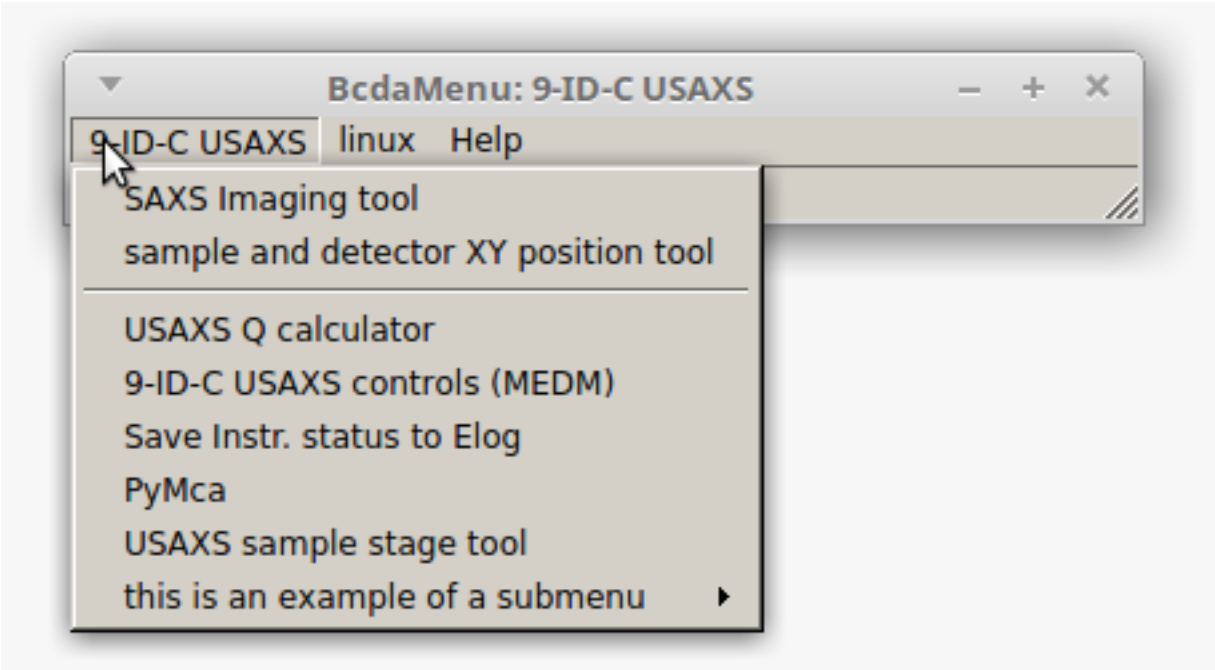


Fig. 2.2: GUI using example settings file, showing the *USAXS* menu.

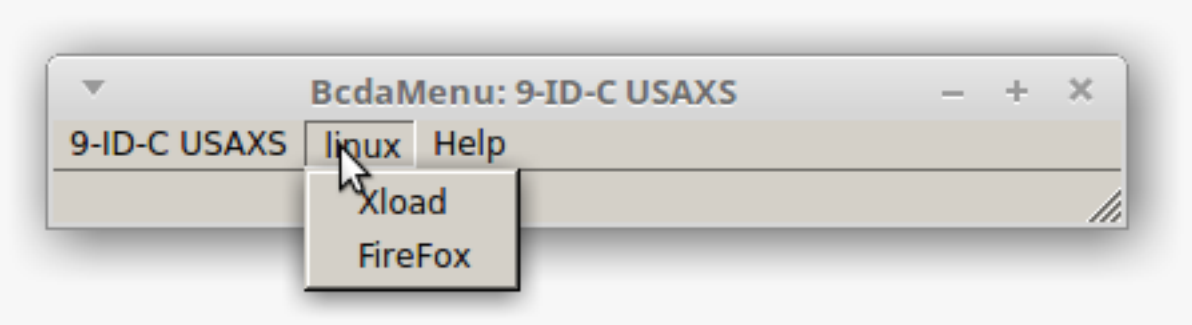


Fig. 2.3: GUI using example settings file, showing the *linux* menu.

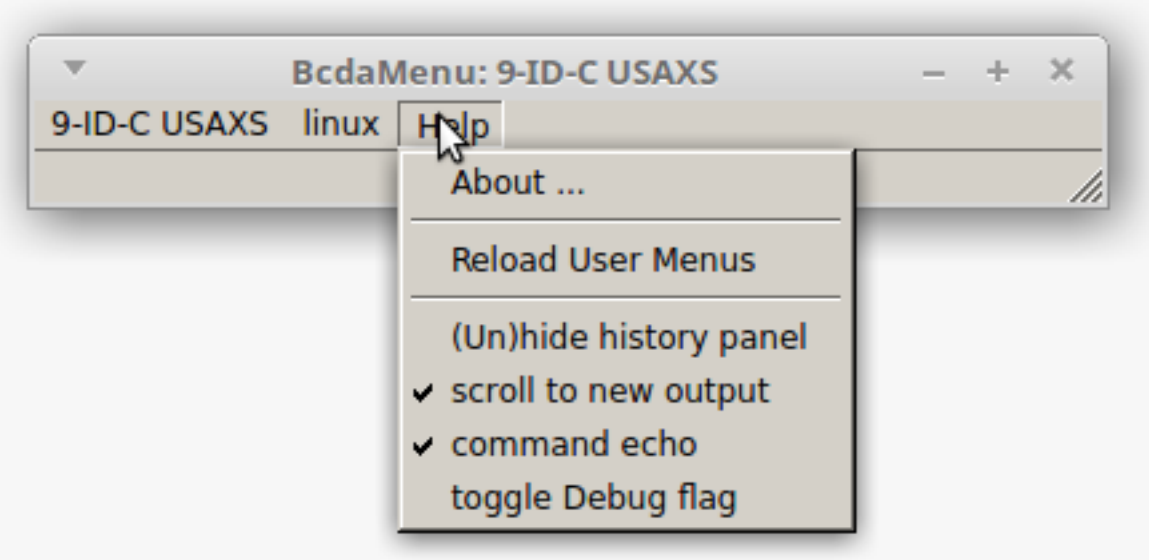


Fig. 2.4: GUI showing the *Help* menu.

These items are available in the *Help* popup menu:

- *About . . .* : prints to the console basic information about this program
- *Reload User Menus* : reloads the settings file (use this when editing/revising that file)
- *(Un)hide history panel* : if checked, show a history panel with command output
- *scroll to new output* : if checked, always scroll when new output is received
- *command echo* : if checked, show command in history when executed
- *toggle Debug flag* : if checked, turns on diagnostic output

2.2.2 Examples

Linux: variety of possibilities

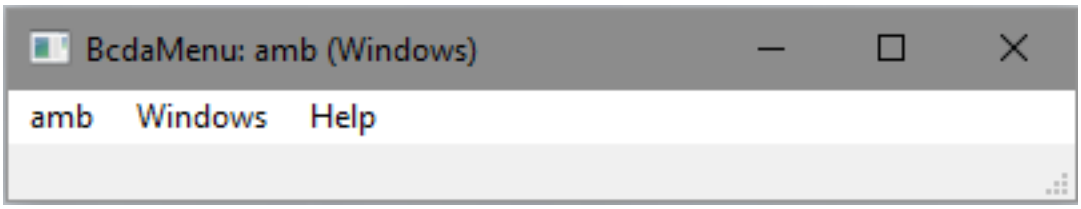
an example from Linux with a number of possibilities

```
# bcdamenu.ini
#
# settings file for BcdaMenu GUI

[BcdaMenu]
title = BcdaMenu: jemian@gov
version = 2017.3.0
menus = IOC BlueSky linux

[IOC]
title = IOCs on gov
20 submenu   = iocgov
42 submenu   = iocprj

[iocgov]
title = IOC: gov (synApps 5.8)
```

```
## synApps 5.8 IOC has start/stop/status/console features
1 caQtDM iocgov   = cd /home/oxygen/JEMIAN/sandbox/ioc/gov; ./start_caQtDM.sh
# 2 screen editor = # this is not supported yet
10 separator   =
8 start IOC    = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; ./gov.sh␣
↪start
14 console iocgov = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; gnome-
↪terminal -e "./gov.sh console"
15 status iocgov  = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; ./gov.sh␣
↪status
23 stop iocgov    = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; ./gov.sh␣
↪stop

[iocprj]
title = IOC: prj on gov (synApps 5.6)
# synApps 5.6 did not have console & process management
101 caQtDM   = cd /home/oxygen/JEMIAN/sandbox/ioc/prj; ./start_caQtDM
103 start    = /home/oxygen/JEMIAN/bin/start_ioc_prj.sh
114 console  = cd /home/oxygen/JEMIAN/sandbox/ioc/prj/iocBoot/iocLinux; gnome-
↪terminal  -e "screen -r"

[linux]
1 edit settings file = /bin/nedit-client /home/oxygen/JEMIAN/bin/bcdamenu.ini
2 type settings file = /bin/cat /home/oxygen/JEMIAN/bin/bcdamenu.ini
44 xload = xload
45 Ku'damm clock = blnuhr

[BlueSky]
title = NSLS-II BlueSky
14 BlueSky console = cd /home/oxygen/JEMIAN/Documents; gnome-terminal -e "/home/
↪oxygen/JEMIAN/bin/use_bluesky.sh bluesky"
20 submenu = BlueSky-mongodb
90 submenu = BlueSky-documentation

[BlueSky-mongodb]
title = mongodb viewer
22 start mViewer server (console) = cd /home/oxygen/JEMIAN/Apps/mViewer; gnome-
↪terminal -e "./start_mviewer.sh 8086"
24 mongodb viewer in web browser (chrome) = /bin/google-chrome http://localhost:8086/
↪index.html
25 mongodb viewer hints = echo "host: gov.aps.anl.gov   port: 27017  others leave␣
↪blank"

[BlueSky-documentation]
title = web documentation
92 BlueSky documentation = /bin/google-chrome http://nsls-ii.github.io/bluesky
93 NSLS-II Software documentation = /bin/google-chrome http://nsls-ii.github.io
```

Windows: caQtDM & PyMca

an example from Windows showing one absolute path and one default path




```
[BcdaMenu]
title = BcdaMenu: amb (Windows)
version = 2017.3.0
menus = amb Windows

[amb]
#title =
1 CaQtDM = "D:\Program Files\caQTDM\bin\windows-x64\caQtDM.exe"
3 separator =
7 PyMca = PyMca

[Windows]
10 edit settings file = "D:\Apps\Sublime Text\sublime_text.exe" C:\Users\Pete\.
↪bcdamenu.ini
```

2.2.3 History panel in main window

With version 2017.4.0, the *BcdaMenu* was transformed from a *button window* style to a *main window* style of application. This change gained several features:

- keyboard motion between the various menus and menu items
- a status line to report program information (like the *unimenu* predecessor of *BcdaMenu*)
- a main window panel to gather any command output and show history

The history panel (like real history) cannot be changed after it has been written. Both *stderr* and *stdout* from any command are combined and reported in the window. There are options (under the *Help* menu) to control what is written to the history.

Since the original program did not have a history panel, the default is to not display the history panel. Again, the *Help* menu has an item to show/hide the history.

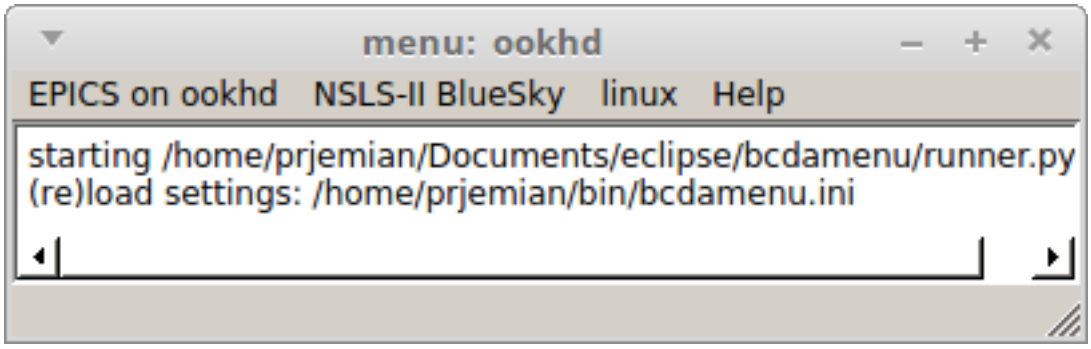


Fig. 2.5: History panel is shown.

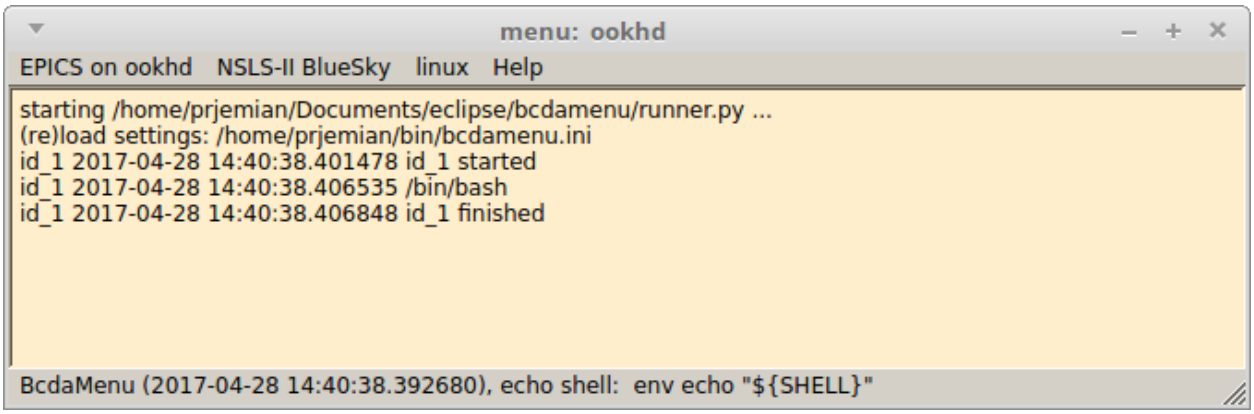


Fig. 2.6: History panel is shown with debugging turned on. This shows when commands are started and stopped. All lines are time stamped with debugging turned on.

2.2.4 Installation

Install this program from the Python Package Index (PyPI) using the *pip* command:

```
pip install bcdamenu
pip install --no-deps bcdamenu
```

The `--no-deps` option tells *pip* not to download and attempt to build newer versions of other required packages.

Requirements

- **Python 2.7**
- **PyQt4**

Your system will need to have the package **PyQt4.QtGui** already installed. A Python 2.7 distribution, such as Anaconda Python provides this package.

This program was developed on a Windows workstation and tested with various Linux distributions (RHEL7, Linux Mint, Raspberry Pi). It is expected to run on any host that provides the standard Python 2.7 packages *and* **PyQt4**.

2.2.5 Source : `about`

source code: about

show the About box

Usage (example)

```
ui = about.InfoBox(self)

ui.setTodoURL(__issues__)
ui.setDocumentationURL(__url__)
ui.setLicenseURL(__license_url__)
ui.setTitle(config_file_parser.MAIN_SECTION_LABEL)
ui.setVersionText("software version: " + __version__)
ui.setSummaryText(__doc__.strip())
ui.setAuthorText(__author__)
```

```
ui.setCopyrightText(__copyright__)

ui.show()
```

InfoBox([parent, settings])	a Qt GUI for the About box
loadUi(ui_file[, baseinstance])	load a .ui file for use in building a GUI

class bcdamenu.about.**InfoBox**(*parent=None*, *settings=None*)

a Qt GUI for the About box

setTodoURL(url)	set the URL for the issue tracker
setDocumentationURL(url)	set the URL for the documentation
setLicenseURL(url)	set the URL for the software license text
setTitle(text)	set the title in the About box
setVersionText(text)	set the version text in the About box
setSummaryText(text)	set the description in the About box
setAuthorText(text)	set the author list in the About box
setCopyrightText(text)	set the copyright string in the About box

closeEvent(*event*)

called when user clicks the big [X] to quit

doDocsUrl()

show documentation URL in default browser

doIssuesUrl()

show issues URL in default browser

doLicense()

show the license URL in default browser

doUrl(*url*)

opening URL in default browser

setAuthorText(*text*)

set the author list in the About box

setCopyrightText(*text*)

set the copyright string in the About box

setDocumentationURL(*url*)

set the URL for the documentation

setLicenseURL(*url*)

set the URL for the software license text

setSummaryText(*text*)

set the description in the About box

setTitle(*text*)

set the title in the About box

setTodoURL(*url*)

set the URL for the issue tracker

setVersionText(*text*)

set the version text in the About box

bcdamenu.about.**loadUi**(*ui_file*, *baseinstance=None*, ***kw*)
load a .ui file for use in building a GUI

Wraps *uic.loadUi()* with code that finds our program's *resources* directory.

See http://nullege.com/codes/search/PyQt4.uic.loadUi

See http://bitesofcode.blogspot.ca/2011/10/comparison-of-loading-techniques.html

inspired by: http://stackoverflow.com/questions/14892713/how-do-you-load-ui-files-onto-python-classes-with-pyside?lq=1

Basic Procedure

1. Use Qt Designer to create a .ui file.
2. Create a python class of the same type as the widget you created in the .ui file.
3. When initializing the python class, use uic to dynamically load the .ui file onto the class.

Here is an example:

```
from PyQt4 import QtGui
import resources

UI_FILE = 'plainTextEdit.ui'

class TextWindow(QtGui.QDialog, form_class):

    def __init__(self, title, text):
        QtGui.QDialog.__init__(self, parent)
        resources.loadUi(UI_FILE, baseinstance=self)
        self.setWindowTitle(title)
        self.plainTextEdit.setPlainText(text)

import sys
app = QtGui.QApplication(sys.argv)
win = TextWindow('the title', __doc__)
win.show()
sys.exit(app.exec_())
```

2.2.6 Source : config_file_parser

source code: config_file_parser

parse the configuration file

readConfigFile(file_name)	
ConfigFileError	general exception from *config_file_parser*
ConfigFileKeyError	exception with a key in the configuration file
clearKnownMenuNames()	keep a list of all known menus so a recursive configuration will be found
MenuBase([parent, order])	base class for menu definitions
Menu([parent, sectionName])	specifications of a menu or submenu
MenuItem([parent, label])	specification of one item in a a menu (or submenu)

Continued on next page

Table 2.3 – continued from previous page

MenuSeparator([parent])	specification of a separator line in a menu

exception bcdamenu.config_file_parser.**ConfigFileError**
general exception from *config_file_parser*

exception bcdamenu.config_file_parser.**ConfigFileKeyError**
exception with a key in the configuration file

class bcdamenu.config_file_parser.**Menu**(*parent=None*, *sectionName=None*)
specifications of a menu or submenu

setTitle(title)	set the text of this menu's title
readConfiguration(config)	read the menu's section from the config file

readConfiguration(*config*)
read the menu's section from the config file

Parameters **config** (*obj*) – instance of ConfigParser()

setTitle(*title*)
set the text of this menu's title

class bcdamenu.config_file_parser.**MenuBase**(*parent=None*, *order=None*)
base class for menu definitions

class bcdamenu.config_file_parser.**MenuItem**(*parent=None*, *label=None*)
specification of one item in a a menu (or submenu)

setCommand(command)	set the text of the command to be executed when this menu item is selected
setLabel(label)	set the text to appear in the menu (called in constructor)

setCommand(*command*)
set the text of the command to be executed when this menu item is selected

setLabel(*label*)
set the text to appear in the menu (called in constructor)

class bcdamenu.config_file_parser.**MenuSeparator**(*parent=None*)
specification of a separator line in a menu

bcdamenu.config_file_parser.**clearKnownMenuNames**()
keep a list of all known menus so a recursive configuration will be found

2.2.7 Source : launcher

source code: launcher

BcdaMenu: Creates a GUI menu button to start common software

MainButtonWindow([parent, settingsfilename])	the widget that holds the menu button
CommandThread()	run the command as a subprocess in its own thread, report any output
read_settings(ini_file)	read the user menu settings from the .ini file

Continued on next page

Table 2.6 – continued from previous page

gui([settingsfilename])	display the main widget
timestamp()	ISO8601-compliant date & time string
main()	process any command line options before starting the GUI

class bcdamenu.launcher.**CommandThread**
run the command as a subprocess in its own thread, report any output

Usage

```
process = CommandThread()
process.setName(process_name)
process.setDebug(self.debug)
process.setSignal(self.process_responded)
process.setCommand(command)
process.start()
```

See https://docs.python.org/3.3/library/subprocess.html

Methods

run()	print any/all output when command is run
execute()	run the command in a shell, reporting its output as it comes in
setCommand(command)	user's command to be run
setDebug(value)	*True* to output more diagnostics
setSignal(signal)	designate the signal to use when subprocess output has been received

execute()
run the command in a shell, reporting its output as it comes in

run()
print any/all output when command is run

setCommand(*command*)
user's command to be run

setDebug(*value*)
True to output more diagnostics

setSignal(*signal*)
designate the signal to use when subprocess output has been received

class bcdamenu.launcher.**MainButtonWindow**(*parent=None*, *settingsfilename=None*)
the widget that holds the menu button

receiver(label, command)	handle commands from menu button
reload_settings_file()	(re)load the settings file and (re)create the menu(s)
build_user_menus(config)	build the user menus
showStatus(text[, isCommand])	write to the status bar
historyUpdate(text)	record history where user can see it
toggleAutoScroll()	change whether (or not) to keep new output in view
toggleDebug([debug_state])	change whether (or not) to output diagnostic information

Continued on next page

Table 2.8 – continued from previous page

toggleEcho()	change whether (or not) to echo command before running it
hide_history_window()	toggle the visibility of the history panel
about_box()	display an About box
closeEvent(event)	

about_box()
display an About box

build_user_menus(*config*)
build the user menus

hide_history_window()
toggle the visibility of the history panel

historyUpdate(*text*)
record history where user can see it

receiver(*label*, *command*)
handle commands from menu button

reload_settings_file()
(re)load the settings file and (re)create the menu(s)

showStatus(*text*, *isCommand=False*)
write to the status bar

toggleAutoScroll()
change whether (or not) to keep new output in view

toggleDebug(*debug_state=None*)
change whether (or not) to output diagnostic information

toggleEcho()
change whether (or not) to echo command before running it

bcdamenu.launcher.**gui**(*settingsfilename=None*)
display the main widget

bcdamenu.launcher.**main**()
process any command line options before starting the GUI

bcdamenu.launcher.**read_settings**(*ini_file*)
read the user menu settings from the .ini file

bcdamenu.launcher.**timestamp**()
ISO8601-compliant date & time string

2.2.8 Change History

Production

2017.9.0

- #43 no output to command history
- #42 only one separator line is added, even if multiple are described

2017.4.0

- #35 spawns commands in separate subprocess threads
- #31 capture all console output to history panel
- #30 changed buttons into menubar, added history panel
- #13 About box added
- #12 **submenu** feature added!

2017.3.5

- #6 source code now documented on ReadTheDocs

2017.3.3

- #27 reload settings file now sets window title
- #26 add option to lay out buttons vertical (horizontal is the default)
- #25 reload settings file now clears previous buttons
- #24 started an Examples section in the documentation

2017.3.2

- #21 allow command line help to be shown on MS Windows
- #20 show version on command line help
- #18 change internal handling of user menu specs from dictionary to list
- #17 preserve capitalization of menu items

2017.3.0

- initial release: 2017.03.21

Development

2017-03-20

- created GitHub repo from *menuUSAXS.py*: https://github.com/BCDA-APS/bcdamenu

2.2.9 License

```
   the author and organization's name.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

3. Neither the names of UChicago Argonne, LLC or the Department of Energy
   nor the names of its contributors may be used to endorse or promote
   products derived from this software without specific prior written
   permission.

4. The software and the end-user documentation included with the
   redistribution, if any, must include the following acknowledgment:

   "This product includes software produced by UChicago Argonne, LLC
   under Contract No. DE-AC02-06CH11357 with the Department of Energy."

*****************************************************************************

DISCLAIMER

THE SOFTWARE IS SUPPLIED "AS IS" WITHOUT WARRANTY OF ANY KIND.

Neither the United States GOVERNMENT, nor the United States Department
of Energy, NOR uchicago argonne, LLC, nor any of their employees, makes
any warranty, express or implied, or assumes any legal liability or
responsibility for the accuracy, completeness, or usefulness of any
information, data, apparatus, product, or process disclosed, or
represents that its use would not infringe privately owned rights.

*****************************************************************************
```

2.3 Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index

T