TABLE OF CONTENTS

1. wmii_newlisp/actions.nlsp [Generics]

[Top] [Generics]
NAME
actions.nlsp
SYNOPSIS
A replacement wmii handler in newlisp. This file contains all the code related to window manager "actions" handlers. These are window manager events that user visible
AUTHOR
Luis R. Anaya
COPYRIGHT

1.1. actions.nlsp/Action [Functions]

[Top] [actions.nlsp] [Functions]
NAME
Action
DESCRIPTION
Evaluates the macro and executes the selected action.
INPUTS
a − action − A list containing the action to perform.

OUTPUT
Action is executed.

1.2. actions.nlsp/action−items [Functions]

[Top] [actions.nlsp] [Functions]
NAME
action−items
DESCRIPTION
Contains all the actions and its bound command name. The bound command is extracted and piped through dmenu/wimenu
INPUTS
a − action − Used for variable substitution for macro creation
OUTPUT
Case statement mapping an action name to a piece of logic.

1.2.1. action−items/exec [Generics]

[Top] [action−items] [Generics]
NAME
exec
DESCRIPTION
Executes dmenu, used to obtain the name of an executable and spawn it to background.

1.2.2. action−items/exec−term [Generics]

[Top] [action−items] [Generics]
NAME
exec−term
DESCRIPTION
Executes dmenu, used to obtain the name of an executable runs it in a terminal.

1.2.3. action−items/quit [Generics]

[Top] [action−items] [Generics]
NAME
quit
DESCRIPTION
Quits from the window manager

1.2.4. action−items/status [Generics]

[Top] [action−items] [Generics]
NAME
status
DESCRIPTION
Forked on start, it is used to write the status in the lower right hand corder of the window manager. The default definition is in newlisp/wmiirc

1.3. actions.nlsp/Actions [Variables]

[Top] [actions.nlsp] [Variables]
NAME
Contains list of available actions
SOURCE

```
(setq Actions (key-list (action-items a)))
```

2. wmii_newlisp/events.nlsp [Generics]

[Top] [Generics]
NAME
events.nlsp
SYNOPSIS
A replacement wmii handler in newlisp.
This file contains all the code related to windoe manager "events" handlers. These are window manager events that *not* user visible
AUTHOR
Luis R. Anaya
COPYRIGHT

2.1. events.nlsp/Event [Functions]

[Top] [events.nlsp] [Functions]
NAME
Event
DESCRIPTION
Evaluates the macro and executes the selected event.
INPUTS
e − event − A list containing the event to perform.
OUTPUT
Event is executed.

2.2. events.nlsp/event-items [Functions]

[Top] [events.nlsp] [Functions]
NAME
event−items
DESCRIPTION
Contains all the events and its bound command. The bound command is obtained from the event loop handler
INPUTS
e − event − Used for variable substitution for macro creation
OUTPUT

Case statement mapping an event name to a piece of logic.

2.2.1. event−items/CreateTag [Generics]

[Top] [event−items] [Generics]
NAME
CreateTag
DESCRIPTION
Creates a workspace.

2.2.2. event−items/DestroyTag [Generics]

[Top] [event−items] [Generics]
NAME
DestroyTag
DESCRIPTION
Destroys a workspace.

2.2.3. event−items/FocusTag [Generics]

[Top] [event−items] [Generics]
NAME
FocusTag
DESCRIPTION
Brings a workspace into view.

2.2.4. event−items/Key [Generics]

[Top] [event−items] [Generics]
NAME
Key
DESCRIPTION
Key handler − not in use.

2.2.5. event−items/LeftBarDND [Generics]

[Top] [event−items] [Generics]
NAME
LeftBarClick LeftBarDND
DESCRIPTION
Handles drag and drop into the lower left corner of the window.

2.2.6. event−items/Notice [Generics]

[Top] [event−items] [Generics]
NAME
Notice
DESCRIPTION
Writes notice in the notice bar which is in the lower right hand corner of the window manager.

2.2.7. event−items/NotUrgentTag [Generics]

[Top] [event−items] [Generics]
NAME
NotUrgentTag
DESCRIPTION
Removes an asterisk into a workspace when an urgent message is expired.

2.2.8. event−items/Termination [Generics]

[Top] [event−items] [Generics]
NAME
List termination ""
DESCRIPTION
List terminator. Used by the key−list function to indicate the last option.

2.2.9. event−items/UnFocusTag [Generics]

[Top] [event−items] [Generics]
NAME
FocusTag
DESCRIPTION
Brings a workspace out of view.

2.2.10. event−items/Unresponsive [Generics]

[Top] [event−items] [Generics]
NAME
Unresponsive
DESCRIPTION
Handles unresponsive clients by presenting a dialog and asking the disposition of the client.

2.2.11. event−items/UrgentTag [Generics]

[Top] [event−items] [Generics]
NAME
UrgentTag
DESCRIPTION
Adds an asterisk into a workspace when an urgent message is detected by the window manager.

2.3. events.nlsp/Events [Variables]

[Top] [events.nlsp] [Variables]
NAME
Contains list of available events.
SOURCE

```
(setq Events (key−list (events−items e)))
```

3. wmii_newlisp/keys.nlsp [Generics]

[Top] [Generics]
NAME
keys.nlsp
SYNOPSIS
A replacement wmii handler in newlisp. This file contains all the code related to key bindings and operations.
AUTHOR
Luis R. Anaya
COPYRIGHT

3.1. keys.nlsp/Key [Functions]

[Top] [keys.nlsp] [Functions]
NAME
Key
DESCRIPTION
Evaluates the macro and executes the selected key stroke.
INPUTS
k − key stroke − A list containing the key stroke and arguments.
OUTPUT
Key stroke command is executed.

3.2. keys.nlsp/key−items [Functions]

[Top] [keys.nlsp] [Functions]
NAME
key−items
DESCRIPTION
This macro builds the list of the operations that relate to key bindings.
INPUTS
k − key stroke − Contains a list with the key stroke and arguments
OUTPUT
Cast statement containing the key stroke and operations.

3.2.1. key−items/Mod−a [Generics]

[Top] [key−items] [Generics]
NAME
Mod−a
DESCRIPTION
Opens the wmii action menu. It reads the choices from the Action list and executes dmenu. The actions are defined in action.nlsp.
NOTE Colors are hard coded, they need to be configurable. This is originally defined with wimenu, but in this case, dmenu was used because of convenience.

3.2.2. key−items/Mod−Control−Down [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Control−Down
DESCRIPTION
Select the stack below.

3.2.3. key−items/Mod−Control−t [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Control−t
DESCRIPTION
Toggles all other key bindings.

3.2.4. key−items/Mod−Control−Up [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Control−Up
DESCRIPTION
Select the stack above.

3.2.5. key−items/Mod−d [Generics]

[Top] [key−items] [Generics]
NAME
Mod−d
DESCRIPTION
Set column to default mode. Default mode is stacked mode in which all clients are of the same size.

3.2.6. key−items/Mod−Down [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Down
DESCRIPTION
Selects the client on the bottom.

3.2.7. key−items/Mod−e [Generics]

[Top] [key−items] [Generics]
NAME
Mod−e
DESCRIPTION
Opens the program menu, gets input from the user and executes the selected program. NOTE Colors are hard coded, they need to be configurable. This is originally defined with wimenu, but in this case, dmenu was used because of convenience.

3.2.8. key−items/Mod−f [Generics]

[Top] [key−items] [Generics]
NAME
Mod−f
DESCRIPTION
Toggle selected client into full screen state.

3.2.9. key−items/Mod−Left [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Left
DESCRIPTION
Selects the client on the left.

3.2.10. key−items/Mod−m [Generics]

[Top] [key−items] [Generics]
NAME
Mod−m
DESCRIPTION
Set column to maximum mode. Maximum mode places the client to occupy all available space. Clients are overlay one on top of the other. The number of clients is listed in the upper right hand corner of the client’s title frame.

3.2.11. key−items/Mod−Return [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Return
DESCRIPTION
Launch a terminal as defined in WMII_TERM.

3.2.12. key−items/Mod−Right [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Right
DESCRIPTION
Selects the client on the right.

3.2.13. key−items/Mod−s [Generics]

[Top] [key−items] [Generics]
NAME
Mod−s
DESCRIPTION
Set column to stack mode. Stack mode overlaps clients one on top of the other. Window decorations are visible on all clients within a workspace.

3.2.14. key−items/Mod−Shift−c [Generics]

[Top] [key−items] [Generics]

NAME
Mod−Shift−c
DESCRIPTION
Kills selected client.

3.2.15. key−items/Mod−Shift−Down [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Shift−Down
DESCRIPTION
Moves the selected client to the bottom.

3.2.16. key−items/Mod−Shift−Left [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Shift−Left
DESCRIPTION
Moves the selected client to the left.

3.2.17. key−items/Mod−Shift−q [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Shift−q
DESCRIPTION
Dismantles the window manager and quits. Ths is a divergence from the original script. It is just for convenience.

3.2.18. key−items/Mod−Shift−Right [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Shift−Right
DESCRIPTION
Moves the selected client to the right.

3.2.19. key−items/Mod−Shift−Space [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Shift−Space
DESCRIPTION
Toggle client between floating and managed layers.

3.2.20. key−items/Mod−Shift−t [Generics]

[Top] [key−items] [Generics]
NAME
Mod−Shift−t
DESCRIPTION

Opens the wmii tag menu. It reads the defined workspaces and permits the user to create one with a custom name.

NOTE Colors are hard coded, they need to be configurable. This is originally defined with wimenu, but in this case, dmenu was used because of convenience.

3.2.21. key−items/Mod−Shift−Up [Generics]

[Top] [key−items] [Generics]

NAME

Mod−Shift−Up

DESCRIPTION

Moves the selected client to the top.

3.2.22. key−items/Mod−Space [Generics]

[Top] [key−items] [Generics]

NAME

Mod−Space

DESCRIPTION

Toggles between floating and managed layers.

3.2.23. key−items/Mod−t [Generics]

[Top] [key−items] [Generics]

NAME

Mod−t

DESCRIPTION

Opens the wmii tag menu. It reads the defined workspaces and permits the user to write the one that the user wishes to swap to.

NOTE Colors are hard coded, they need to be configurable. This is originally defined with wimenu, but in this case, dmenu was used because of convenience.

3.2.24. key−items/Mod−Up [Generics]

[Top] [key−items] [Generics]

NAME

Mod−Up

DESCRIPTION

Selects the client on the top.

3.2.25. key−items/Tags−0−to−9 [Generics]

[Top] [key−items] [Generics]

NAME

Tags−0−to−9

DESCRIPTION

Loop that creates key for workspace operations.

NOTES

Creates bindings to move clients among workspaces and create new ones with a number from 0 to 9.

3.3. keys.nlsp/Keys [Variables]

[Top] [keys.nlsp] [Variables]
NAME
Contains list of key bindings.
SOURCE

```
(setq Keys (convert-to-newline-string (key-list (key-items
k))))
```

4. wmii_newlisp/menu.nlsp [Generics]

[Top] [Generics]
NAME
menu.nlsp
SYNOPSIS
A replacement wmii handler in newlisp. This file contains all the code related to window manager "menu" handlers. This file defines menu operations that are run through wmii9menu.
AUTHOR
Luis R. Anaya
COPYRIGHT

4.1. menu.nlsp/Menu [Functions]

[Top] [menu.nlsp] [Functions]
NAME
Menu
DESCRIPTION
Evaluates the macro and executes the selected menu action.
INPUTS
m − menu action − A list containing the menu action to perform.
OUTPUT
Menu action is executed.

4.2. menu.nlsp/menu−items [Functions]

[Top] [menu.nlsp] [Functions]
NAME
menu−items
DESCRIPTION
Contains all the actions and its bound mouse action.
INPUTS
m − menu selection − Contains the variable used to create the case statement.
OUTPUT

Case statement mapping the mouse event to an operations is created.

4.2.1. menu−items/ClientMouseDown [Generics]

[Top] [menu−items] [Generics]
NAME
ClientMouseDown
DESCRIPTION
It handles operations done when the mouse is clicked and held; Used for window operations.
NOTES
It can be improved by not having the options hard coded which should provide flexibility on use. As of now, wi_fmenu presents a menu on right click only.

4.2.2. menu−items/LeftBarClick [Generics]

[Top] [menu−items] [Generics]
NAME
LeftBarMouseClick
DESCRIPTION
It handles operations done when the mouse is clicked on the lower left hand corner.
Used for workspace selection (workspace switching).
NOTES
It can be improved by not having the options hard coded which should provide flexibility on use. As of now, wi_fmenu presents a menu on right click only.

4.2.3. menu−items/LeftBarMouseDown [Generics]

[Top] [menu−items] [Generics]
NAME
LeftBarMouseDown
DESCRIPTION
It handles operations done when the mouse is clicked and held on the lower left hand corner.
Used for workspace operations.
NOTES
It can be improved by not having the options hard coded which should provide flexibility on use. As of now, wi_fmenu presents a menu on right click only.

4.3. menu.nlsp/menu−operations [Functions]

[Top] [menu.nlsp] [Functions]
NAME
menu−operations
DESCRIPTION
Contains all the actual operations that get executed when a menu item is selected form wmii9menu.
INPUTS
m − menu selection − Contains a list with the menu selection and arguments
OUTPUT

Selected operations is executed.

4.3.1. menu−operations/Client−3−Delete [Generics]

[Top] [menu−operations] [Generics]
NAME
Client−3−Delete
DESCRIPTION
Closes the selected window. Not forced.

4.3.2. menu−operations/Client−3−Fullscreen [Generics]

[Top] [menu−operations] [Generics]
NAME
Client−3−Fullscreen
DESCRIPTION
Places the selected window in undecorated fullscreen mode.

4.3.3. menu−operations/Client−3−Kill [Generics]

[Top] [menu−operations] [Generics]
NAME
Client−3−Kill
DESCRIPTION
Kills the selected window. Forced

4.3.4. menu−operations/LBar−1−Click [Generics]

[Top] [menu−operations] [Generics]
NAME
LBar−1−Click
DESCRIPTION
Selects the workspace by clicking on it in the lower left had corner of the screen.

4.3.5. menu−operations/LBar−3−Delete [Generics]

[Top] [menu−operations] [Generics]
NAME
LBar−3−Delete
DESCRIPTION
Closes all the clients and deletes the workspace by *right* clicking on it in the lower left had corner of the screen.

4.4. menu.nlsp/Menus [Variables]

[Top] [menu.nlsp] [Variables]
NAME
Contains list of available menu actions
SOURCE

```
(setq Menus (key−list (menu−items m)))
```

5. wmii_newlisp/wmii.nlsp [Generics]

[Top] [Generics]
NAME
wmii.nlsp
SYNOPSIS
A replacement wmii handler in newlisp.
This file contains all the utility functions used by the scripts for various purposes. Each of them is is documented in its section.
AUTHOR
Luis R. Anaya
COPYRIGHT
(c) 2012 by Luis R. Anaya

5.1. wmii.nlsp/clear_operation_list [Generics]

[Top] [wmii.nlsp] [Generics]
NOTES
The following operation lists are assigned "nil" on start. They are initialized when the keys.nlsp, menu.nlsp, actions.nlsp and events.nlsp are loaded later on in the code.
SOURCE

```
(setq Keys nil)
(setq Actions nil)
(setq Menus nil)
(setq Events nil)
```

5.2. wmii.nlsp/control [Generics]

[Top] [wmii.nlsp] [Generics]
NOTES
The following gets the current values for font, normal and focus color values.
SOURCE

```
(setq wmiifont (wi_readctl "font"))
(setq wmiinormcol (wi_readctl "normcolors"))
(setq wmiifocuscol (wi_readctl "focuscolors"))
```

5.3. wmii.nlsp/convert−to−newlines−string [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
convert−to−newline−string

DESCRIPTION
Converts a list into a newline separated string.
INPUTS
A string of list
OUTPUT
A newline separated string.
NOTES
Naming convention needs to be fixed.

5.4. wmii.nlsp/convert−to−padded−string [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
convert−to−padded−string
DESCRIPTION
Converts a list into a space separated string.
INPUTS
A string of list
OUTPUT
A space separated string.
NOTES
Naming convention needs to be fixed.

5.5. wmii.nlsp/key−list [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
key−list
DESCRIPTION
Parses macro output to get the keys. Used all different handlers to make the available operations known.
INPUTS
Output from the macro.creation script.
OUTPUT
List with all available operations.

5.6. wmii.nlsp/loading−routines [Generics]

[Top] [wmii.nlsp] [Generics]
NOTES
The following loads all the required scripts. It iterates through WMII_CONFPATH and loads them into memory.
SOURCE

```
(dolist (dirtoload (parse (env "WMII_CONFPATH") ":"))
;;
;; Load system in order.
;;
```

```
        (catch
         (load (append dirtoload "/events.nlsp" )) 'result)
        (catch
         (load (append dirtoload "/actions.nlsp" )) 'result)
        (catch
         (load (append dirtoload "/keys.nlsp" )) 'result)
        (catch
         (load (append dirtoload "/menu.nlsp" )) 'result)
)
```

5.7. wmii.nlsp/start [Generics]

[Top] [wmii.nlsp] [Generics]
NOTES
The following is used to determine that the correct script and tells the window manager to start execution using these for management.
SOURCE

```
(if (nil? scriptname)
    (setq scriptname wmiiscript))

(if (=  scriptname wmiiscript)
    (! (append "echo Start " wmiiscript " | wmiir write /event
2>/dev/null" )))
```

5.8. wmii.nlsp/wi_eventloop [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_eventloop
DESCRIPTION
Main event loops. "Tails" from the wmii event file and gets the event from the window manager.
INPUTS
None
OUTPUT
Main execution.

5.8.1. wi_eventloop/keys [Generics]

[Top] [wi_eventloop] [Generics]
NOTES
Write the keystrokes there were defined in keys.nlsp.
SOURCE

```
  (! (append "echo \"" Keys "\"| wmiir write /keys"))
```

5.8.2. wi_eventloop/launch_event_process [Generics]

[Top] [wi_eventloop] [Generics]
NOTES
Create the pipes and launch the process the reads the events.
SOURCE

```
  (map set '(myin bcout) (pipe))
  (map set '(bcin myout) (pipe))

  (setq wmiir_exec (first (exec "which wmiir")))
  (setq pid (process (append wmiir_exec " read /event ") bcin
bcout))
```

5.8.3. wi_eventloop/process_event [Generics]

[Top] [wi_eventloop] [Generics]
NOTES
Select the operation based on:

- Keystroke
- Quit event
- Event that has been defined in event.nlsp
- Action defined in action.nlsp
- Menu command defined in menu.nlsp

SOURCE

```
                    (cond
                     ((= "Key" commando ) (Key (nth 1 a-line)))
                     ((= "Quit" commando)
                      (begin
                        (destroy pid)
                        (destroy statuspid)
                        (exit 0)
                        ))
                     ((not (nil? (find commando Events)))
                      (Event a-line))
                     ((not (nil? (find commando Menus)))
                      (Menu a-line)) )
                    ))
   ))
)
```

5.8.4. wi_eventloop/read_events [Generics]

[Top] [wi_eventloop] [Generics]
NOTES

Reading events consists of:

- Iterate through the output stream. and store into a 320 byte buffer.
- Split the buffer by each event that are separated by newlines.
- Iterate through those events.
- Parse events by space and process them.

SOURCE

```
  (while (= 1 1)
    (read myin a-stream 320)
    (setq a-stream (parse a-stream "\n"))

    (dolist (a-line a-stream)
            (setq a-line (parse (trim (trim a-line "\n"))))

            (if (not (= a-line '()))

                (begin
                  (setq commando (first a-line))
```

5.9. wmii.nlsp/wi_fatal [Functions]

[Top] [wmii.nlsp] [Functions]

NAME

wi_fatal

DESCRIPTION

Writes a fatal message and terminates.

INPUTS

Argument list to print.

OUTPUT

Message written and process is terminated.

NOTES

Not in use − not operational.

5.10. wmii.nlsp/wi_fmenu [Functions]

[Top] [wmii.nlsp] [Functions]

NAME

wi_fmenu

DESCRIPTION

It is used to run the pop−up menu wmii9menu and display the selection. After a selection is obtained, it is sent to menu_operations for execution.

INPUTS

loc − Location in which the menu was clicked. Text, currentlly only "Client" and "LBar" are in use.

cid − Client id, either window id or workspace id.

carg − Client arguments. Pressed button.

options − Menu options to be displayed. (first option) is default.
OUTPUT
Menu executes and an answer is obtained from the user.
NOTES
Only right click raises the menu.

5.11. wmii.nlsp/wi_notice [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_notice
DESCRIPTION
Writes a notice in xmessage and returns.
INPUTS
Argument list to print.
OUTPUT
Notice is written.
NOTES
Not in use − not operational.

5.12. wmii.nlsp/wi_processexists [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_processexists
DESCRIPTION
Runs "ps" and checks if a given processid is running or not.
INPUTS
Process id
OUTPUT
true if running, nil if not.

5.13. wmii.nlsp/wi_proglist [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_proglist
DESCRIPTION
Gets a sorted list of programs in a given directory.
INPUTS
d − Directory to obtain the list.
OUTPUT
Sorted list of files from a directory.
NOTES
Not in use.

5.14. wmii.nlsp/wi_readctl [Functions]

[Top] [wmii.nlsp] [Functions]

NAME
wi_readctl
DESCRIPTION
Reads a control attribute from the ixp file system and writes the current value.
INPUTS
k − Control Attribute
OUTPUT
Value

5.15. wmii.nlsp/wi_runcmd [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_runcmd
DESCRIPTION
Executes tags operations.
INPUTS
c − Command to execute.
OUTPUT
Command assigned to a given workspace.
NOTES
This can be repurposed for custom scripts. It needs to be fixed and tested.

5.16. wmii.nlsp/wi_script [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_script
DESCRIPTION
Executes a shell script. Not in use. Not operational
INPUTS
s − Script to execute
OUTPUT
Scripts executed
NOTES
This can be repurposed for custom scripts. It needs to be fixed and tested.

5.17. wmii.nlsp/wi_selclient [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_selclient
DESCRIPTION
Gets the selected client.
INPUTS
None
OUTPUT
Selected client.

5.18. wmii.nlsp/wi_seltag [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_seltag
DESCRIPTION
Gets the selected workspace.
INPUTS
None
OUTPUT
Selected workspace.

5.19. wmii.nlsp/wi_tags [Functions]

[Top] [wmii.nlsp] [Functions]
NAME
wi_tags
DESCRIPTION
Gets the list of current workspaces.
INPUTS
None
OUTPUT
List of current workspaces

6. wmii_newlisp/wmiirc [Generics]

[Top] [Generics]
NAME
wmiirc
SYNOPSIS
A replacement wmii handler in newlisp.
This file contains all the global configuration file.
AUTHOR
Luis R. Anaya
COPYRIGHT

NOTES
Ideally this should contain configuration, but some of these routines should be in wmii.nlsp to be not user facing.

6.1. wmiirc/cofigurable_parameters [Generics]

[Top] [wmiirc] [Generics]
NOTES
User configurable parameters
SOURCE

```
(setq wmiiscript "wmiirc")
```

```
# Configuration Variables
(setq MODKEY "Mod1")
(setq UP "k")
(setq DOWN "j")
(setq LEFT "h")
(setq RIGHT "l")
```

6.2. wmiirc/colors [Generics]

[Top] [wmiirc] [Generics]
NOTES
Color, fonts and terminal definitions are here.
SOURCE

```
(setq WMII_NORMCOLORS "\#bbbbbb \#222222 \#444444" )
(setq WMII_FOCUSCOLORS "\#eeeeee \#005577 \#005577 " )
(setq WMII_BACKGROUND "\#0C0C0C")

(setq WMII_FONT "-*-fixed-medium-r-*-*-13-*-*-*-*-*-*-*")

(setq WMII_TERM "xterm")
```

6.3. wmiirc/create_config [Generics]

[Top] [wmiirc] [Generics]
NOTES
Create configuration file in home directory
SOURCE

```
(if (not (directory? WMII_HOME))
    (begin
      (make-dir WMII_HOME)
      (setq message (append
          "Welcome to wmii,\n\n"
          "Most of wmii's default key bindings make use of the
\n"
          "Windows key, or equivalent. For keyboards lacking
such\n"
          "a key, many users change this to the Alt key.\n\n"
          "Which would you prefer?"))
      (setq res (exec (format "wihack -type DIALOG xmessage
-nearmouse -buttons Windows,Alt -print -fn %s %s " WMII_FONT
message)))
      (if (and (= res "Alt")
               (= MODKEY "Mod1"))
          (setq fp (open (append WMII_HOME "/wmiirc_local")
"w"))
          (write-line fp (format "(setq MODKEY \"%s\")"
```

```
MODKEY))
          (close fp)
          (! (append "chmod +x " WMII_HOME "/wmiirc_local"))
          )
      )
)
```

6.4. wmiirc/exception_handling [Generics]

[Top] [wmiirc] [Generics]
NOTES
Exception handling routine, currently stubbed.
SOURCE

```
(define (ctrlC-handler)
  (destroy pid)
  (exit)
)



;(signal SIGINT 'ctrlC-handler)
;(signal SIGKILL 'ctrlC-handler)
;(signal SIGQUIT 'ctrlC-handler)
;(signal SIGABRT 'ctrlC-handler)
;(signal SIGSTOP 'ctrlC-handler)
;(signal SIGTERM 'ctrlC-handler)
```

6.5. wmiirc/history [Generics]

[Top] [wmiirc] [Generics]
NOTES
History file location. Not in use
SOURCE

```
# Menu history
(setq hist (append WMII_HOME "/history"))
(setq histnum 5000)
```

6.6. wmiirc/home_directory [Generics]

[Top] [wmiirc] [Generics]
NOTES
Determine home directory
SOURCE

```
(if (not (nil? (env "WMII_CONFPATH")))
```

```
    (setq WMII_HOME (first (parse (env "WMII_CONFPATH" )
":")))
)
```

6.7. wmiirc/ixp_defaults [Generics]

[Top] [wmiirc] [Generics]
NOTES
Default values in the ixp file system.
SOURCE

```
# Column Rules − the easy way
(! "
wmiir write /colrules <<!
/gimp/ −> 17+83+41
/.*/ −> 62+38 # Golden Ratio
!
")

# Tagging Rules − Same
(! "
wmiir write /tagrules <<!
/MPlayer|VLC/ −> ~
!"
)
```

6.8. wmiirc/operational_global_variables [Generics]

[Top] [wmiirc] [Generics]
NOTES
Operational variables. Not for user use.
SOURCE

```
(setq WMII_HOME nil)

(setq pid nil)
(setq statuspid nil)

(setq Keys nil)
(setq Actions nil)
(setq Events nil)


(constant 'SIGINT 2)
(constant 'SIGKILL 9)
(constant 'SIGQUIT 3)
(constant 'SIGABRT 6)
```

```
(constant 'SIGSTOP 19)
(constant 'SIGTERM 15)
```

6.9. wmiirc/remaining [Generics]

[Top] [wmiirc] [Generics]
NOTES
The remaining code is non user configurable. It:

- set up status and colors.
- loads libraries
- load workspaces
- runs event loop.

6.10. wmiirc/status [Functions]

[Top] [wmiirc] [Functions]
NAME
status
DESCRIPTION
Displays status in the lower right hand corner. It defaults to load and date.
INPUTS
None
OUTPUT
Load "|" Date
NOTES
Other status types and variables can be used. It gets forked by newlisp and runs on the background.

6.11. wmiirc/user_configuration [Generics]

[Top] [wmiirc] [Generics]
NOTES
Load user configuration. Not really too useful for now. But it is there.
SOURCE

```
(define (local_events) 1)

(catch
 (load (append WMII_HOME "/wmiirc_local")) 'result)
```

Generated from ./wmii_newlisp/ with ROBODoc V4.99.38 on Sun May 06 2012 21:09:43