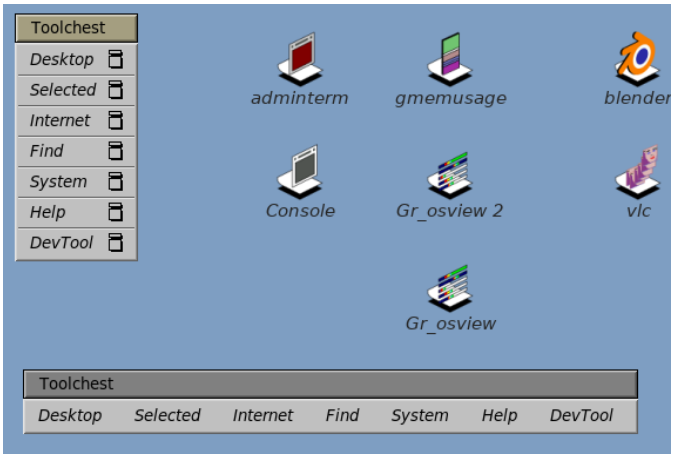
Toolchest

The Toolchest is an utility menu application that serves as the primary access point for desktop user interfaces. For example, users can access interfaces for everyday tasks such as customizing their desktop, accessing applications and Web tools, backing up and restoring their files, and finding information (help) on a variety of desktop topics. By default, the Toolchest is located in the upper left corner.

Toolchest in Classic Mode



Toolchest in Modern Mode

Extract from IRIX 6.5.x man page

Synopsis

```
$ toolchest [-horizontal | -vertical] [-decal | -nodecal]
```

Description

The toolchest program displays a list of buttons, each of which can invoke a useful command or present a submenu of additional buttons. The standard set of menu buttons in the IRIX Interactive Desktop toolchest is Desktop, Selected, Find, System, and Help. Each menu provides lists of useful commands for running top-level window management programs (in conjunction with 4Dwm), desk utilities, search utilities, system administration functions, or documentation-displaying programs.

Options

-horizontal	This option specifies that the toolchest should display the top level horizontally, rather than vertically.

-vertical	This option specifies that the toolchest should display the top level vertically. This is the default, but the option can be used to override saved resources.
-decal	This option specifies that the toolchest should display decals. For a vertical toolchest, a decal is displayed on each top level button that contains a menu. For a horizontal toolchest, one decal is displayed to the left of the toolchest. For an icon toolchest, no decals are ever displayed. This option is on by default.
-nodecal	This option specifies that the toolchest should not display decals.
-hidetitle	This option eliminates the title bar of the toolchest. This was the default in earlier releases but with the availability of desks, the titlebar is enabled by default and shows the name of the current desk.
-showtitle	This option shows the titlebar of the toolchest. This option is on by default.
-title	This option sets the title in the toolchest. If it is not set, the name "Toolchest" is used.

Users can change which buttons and/or menus are displayed by the toolchest program, as well as the command lines which will be launched by button selections.

The toolchest program reads a description of its buttons and menus from a menu description file when the program starts. The standard menu set is described by the system toolchest file **/opt/MaXX/etc/system.chestrc**, but users can customize their toolchest menu by providing either an auxiliary toolchest file named .chestrc or a user-customized toolchest file named .chestrc in their home directory. The system toolchest file includes the auxiliary file, and thus adds its contents to the default system menu, while the user-customized toolchest file overrides the system toolchest file. It is suggested that the auxiliary toolchest file be used for customization if possible, as future changes to the system toolchest will then be incorporated automatically. If a user-customized toolchest file is to be used, it is suggested that the system toolchest file be copied to **$HOME/.chestrc** as a starting point and then the user can modify the .chestrc file.

Remote toolchests launched via /usr/sbin/accessworkstation or from the Toolchest–Desktop–AccessFiles–Another User/By Remote Login) use the resource file /opt/MaXX/etc/remote.chestrc to show a limited number of menu choices.

Invoking toolchest with no arguments causes the user or system menu description file to be read. When one or more file names are specified to toolchest, the menu set is completely described by the list of files. Instead of files, directory names may also be specified, in which case all files in that directory that end with .chest are included, in alphabetical order. Specifying command line arguments is most useful in designing the menu layout; once this has been done, the user can concatenate the files named on the command line into a single file, name it .chestrc and put it in their home directory.

The menu tree is always rooted at a menu named "**Toolchest**"; thus the Menu **Toolchest { ... }** must be provided somewhere in the toolchest description. For example, if a menu named "**Tools**" with menu items "Gedit" and

"Gmemusage" needs to be added to the toolchest, the following should go in .chestrc :

```
Menu Tools
{
no-label f.separator
"Gedit" f.checkexec.sh.le "/usr/bin/gedit"
"Gmemusage" f.checkexec.sh.le "/opt/MaXX/bin/gmemusage"
}

Menu ToolChest

{
no-label f.separator
"Tools" f.menu Tools
}
```

Toolchest buttons and menus may be operated with either left or right mouse buttons (mouse buttons 1 or 3).

The default shell to run the commands is determined in the same way as is used by mwm, 4Dwm and 5Dwm. If the environment $MWMSHELL is defined, toolchest uses that. Otherwise, if the $SHELL is defined, it uses that. Otherwise, it uses /bin/sh. In the interests of performance, however, alternate "shell" versions of the execute commands are provided that force the use of /bin/sh regardless of the shell specified through the environment. Since /bin/sh is faster than some other shells, this can speed up launching commands. These alternate forms, used in the system toolchest, are described below, and are highly recommended unless you
have commands that are shell specific.

Files

```
/opt/MaXX/etc/system.chestrc
~/.chestrc
```

Resources

"Toolchest" is the resource class name for toolchest. "toolchest" is the resource instance name for toolchest and takes precedence over "Toolchest" when used to specify resources. However, for historical reasons, when invoked at system startup (from the Xsession.dt file), toolchest is invoked with the -name Toolchest option.

Customization

The actual location of the system-wide class resource file may depend on certain environment variables. The toolchest should not be resized. Do not set a size with geometry. The toolchest selects an optimal size based on the

font. To get a different sized toolchest, set the Toolchest*fontList: to some other font in your resources. For example, a smaller toolchest can be created with:

Change Font size in Classic Look and Feel

1. Edit the file $HOME/.maxxdesktop/Xdefaults.d/Xdefault.classic

```
toolchest*fontList:  -adobe-helvetica-bold-r-normal--10-*
```

2. Update the Desktop settings and restart toolchest

```
$ update-desktop
$ killall toolchest
$ toochest &
```

Change Font size in Modern Look and Feel

1. Edit the file **$HOME/.maxxdesktop/Xdefaults.d/Xdefault.modern**

```
*toolchest*renderTable:  xft
*toolchest*xft*fontType:  FONT_IS_XFT
*toolchest*xft*fontName:  Sans
*toolchest*xft*fontStyle:  Oblique
*toolchest*xft*fontSize:  10
```

2. Update the Desktop settings and restart toolchest

```
$ update-desktop
$ killall toolchest
$ toochest &
```

Menu Description Format

The format for the menu description file is a subset of that described for the Motif Window Manager, mwm, with a few extensions. toolchest recognizes the keyword menu and the operators f.menu, f.title, f.exec, f.separator, and f.label. In addition, the new operators f.checkexec, f.checkexpr, f.exec.sh, f.checkexec.sh, f.checkexec.sh.le, and f.checkexpr.sh have been added.

mwm-compatible Operators

The menu keyword is followed by a menu name field, then by a set of curly braces containing one or more menu description lines. Each such line has a label field, an operator field, and may have a target field.

The **f.title** operator defines a title label to be placed in the menu.

The **f.separator** operator causes a horizontal line to be drawn below the previous label.

The **f.menu** operator causes a subsidiary menu to be invoked when its label is selected. The menu may also contain the keyword dynamic which is used internally for several Desktop applications to dynamically update the toolchest's contents. There is currently no publicly available method for other applications to take advantage of this capability.

The **f.label** operator defines a label to be placed in the menu.

The **f.exec** operator defines a command to be executed behind a command button. If f.exec is used, no validation on the executability will be performed. If a more robust treatment at run-time is desired, refer to the extension operators described below. It is also possible to improve performance by using the shell versions of the exec operators, also defined below. Prior to executing the command, toolchest will load the environment defined in $HOME/.maxxdesktop/desktopenv. This file can be modified by selecting Desktop>Cusomize>Utilities from the Toolchest. Any commands that contain double or single quotes should be protected from the shell by preceding them with a backslash character. For example, if you wish to execute the command echo "Testing the toolchest" then you would add an entry to the chest's resource file as follows:

```
"Test" f.exec "echo \"Testing the toolchest\""
```

Unlike in mwm, the same menu may be named twice. Subsequent references to the menu add items to the menu. The primary purpose of declaring a menu twice is to add items to a system menu in a private file. For example, in the auxiliary toolchest file, a user could add a new pane to the top level by adding items to the ToolChest menu.

Extension Operators

In addition to the mwm-compatible operators, a set of extension operators are available.

The **f.checkexec** operator defines a command to be executed behind a command button. When selected, it behaves exactly like f.exec above. However, when the toolchest menus are being built, a validation check is run. If the command (the first argument in the command string) begins with a rooted path name (begins with "/"), then that path is checked for the existence and executability of the file. If the file does not exist, or if it is not executable, then this entry will be grayed out in the toolchest. Thus, the toolchest user will not be able to pick a menu item that will fail to execute. For example,

```
"Magnifier" f.checkexec "/usr/sbin/mag"
```

insures that /usr/sbin/mag is available to be run. If it has not been installed, for instance, the "Magnifier" item will be

grayed out.

The **f.checkexpr** operator defines a command to be executed behind a command button. When selected, it behaves exactly like f.exec above. However, when the toolchest menus are being built, a validation check is run. The f.checkexpr operator takes two double-quote-delimited expressions. The first expression is evaluated when the toolchest menus are being built; if this shell command expression fails (returns a nonzero status), then this entry will be grayed out in the toolchest. Since an arbitrary shell expression may be provided for this evaluation, a large degree of care may be exercised by the button programmer interested in protecting users from environmental dependencies which may lie within the actual command line itself. The second expression given to f.checkexpr defines the command to be executed when its button is selected, just as for f.checkexec and f.exec. For example, the command

"Flip Logo" f.checkexpr "test -x /usr/demos/bin/flip -a -r
/usr/demos/data/models/logo.bin" "/usr/demos/bin/flip
/usr/demos/data/models/logo.bin"

provides a test expression that insures the executability of the command and the readability of the critical data file. If either of these files has been deleted or lacks the required permissions, the "Flip Logo" item will be grayed out.

For improved performance, the various forms of f.exec all have shell versions of them, made by appending .sh to their names (i.e., f.exec.sh, f.checkexec.sh, and f.checkexpr.sh). Use of these versions forces the command to be run in /bin/sh instead of in $MWMSHELL or $SHELL. For certain shells, /bin/sh is faster. It is highly recommended that these shell forms be used (the system toolchest uses them). However, the older forms are provided for compatibility with older versions of toolchest.

Note that if you wish to see the sparkle launch effect when you launch from your menus, you need to add .le to the checkexec command. (i.e, f.checkexec.sh.le).

Supported Actions

Name	Description	Use MLaunch	Visual Effect	Validate Presence
f.menu	Menu			
f.separator	Separator			
f.exec	exec command	NO		
f.exec.sh	exec with sh command	NO		
f.exec.le	exec command + effect	NO	YES	
f.checkexec	use mlaunch + check	YES		
f.checkexec.sh	use mlaunch + check in SH	YES		YES

f.checkexec.sh.le	use mlaunch + check in SH	YES	YES	YES
f.dmb.le		YES	YES	YES
f.quit	Close Toolchest			
f.nop	No operation			

Nested Menus

The name on a menu line can be referenced by a f.menu operator from another menu description; this defines a cascading menu relationship.

Top level buttons displayed in the toolchest window are defined in the "Toolchest" menu. Multiple references to the same menu are not supported. Menu names should be unique. Consider the following (partial) definition:

```
menu ToolChest
{
"System" f.menu system
no-label f.separator
"Windows" f.menu windows
no-label f.separator
"Tools" f.menu tools
no-label f.separator
"Demos" f.menu demos
}

menu tools
{
"Tools" f.title
"Shell" f.checkexec.sh.le "xterm"
no-label f.separator
"Showmap" f.checkexec.sh.le "/usr/sbin/showmap"
"Makemap" f.checkexec.sh "/usr/sbin/makemap"
"Clocks" f.menu clocks
}

menu clocks
{
"" f.title
```

```
"Square Clock" f. checkexec. sh. le "/usr/sbin/clock"
"Analog Clock" f. exec. sh "xclock"
"Round Clock" f. checkexec. sh. le "oclock -bg black -fg \"dark red\""
"Digital Clock" f. exec. sh "xclock -digital"
}
```

...

Note that the "clocks" menu is cascaded from the "tools" menu, and that "tools" is cascaded from the "ToolChest" menu. So in this case the toolchest has the following top level buttons:

```
-----------
| System  |
|---------|
| Windows |
|---------|
| Tools   |
|---------|
| Demos   |
-----------
```

and selecting the "Tools" button will pop up a menu that looks something like this:

```
--------------
|   Tools    |
|============|
|   Shell    |
|------------|
| Showmap    |
| Makemap    |
| Clocks ->  |
--------------
```

Sample .chestrc file

The following sample .chestrc file adds a menu of my favorite things to the top level. This menu includes the program atlantis as well as a games sub-menu.

```
menu User
```

```
{
"My Favorite Things" □f.menu □mystuff
}

menu mystuff
{
"dolphins" □□□□f.exec "/usr/demos/bin/atlantis"
"Test Program" □□□f.exec "source ~/.variables;~/testprog"
"games" □□□□f.menu mygames
}

menu mygames
{
"flight simulator" □□f.exec /usr/demos/bin/flight
"arena" □□□□f.exec /usr/demos/bin/arena
}
```

NOTE
When the desktop is configured off with /etc/chkconfig desktop off, an alternate toolchest file called /usr/lib/X11/nodesktop.chestrc is used; it contains fewer entries and functions.

For more information about the entire IRIX Interactive Desktop environment, see the IID (1) man page.

Revision #21
Created 27 June 2020 00:32:50 by Eric Masson
Updated 1 May 2021 16:34:01 by Eric Masson