For: jan
Printed on: Tue, Nov 17, 1992 15:45:53

Document: xmfm-manual
Last saved on: Tue, Nov 17, 1992 15:45:38

The Xmfm Users Guide

Jan Newmarch

Faculty of Information Science
and Engineering
University of Canberra
PO Box 1, Belconnen
ACT, Australia
email: jan@ise.canberra.edu.au

Chapter 1 Overview

`xmfm` is a file manager designed to run under the X Window System. It is built using the Motif toolkit, and has the Motif look–and–feel. `xmfm` displays panes of executable files (programs), ordinary files and the directories within a particular directory. It uses icons to represent different types of files, allowing easy distinction between files.

Any one of these files can be selected, which sets a toolbar of possible actions for the selected file. By clicking on one of these toolbar buttons, the action for the selected file is carried out. By double clicking on a file, the default action for that file is executed.

Multiple directories can be displayed in multiple panes. Files can be dragged from one pane to another, to move or copy them [not yet implemented]. New files and directories can be created within any directory pane. The contents of a pane can be replaced with the contents of another directory so that the user can "move" between directories.

The actions associated with files can be configured by the user by using a configuration file. Actions are set by the type of the file (executable, directory or other), and a pattern using shell pattern matching. A default configuration file is supplied. For example, all ordinary files have default actions of editing and viewing the file; a `Makefile` has additional actions of running the program `make`, and running `make` with an argument; any file ending in `.c` has the additional default actions of compiling to object or executable, or of running make on it. Directories have default actions of replacing the current pane, or of creating a new pane.

Each directory has a set of actions independant of any selected file. For example, refreshing the directory contents, or running the `make` program in the current directory. These actions are shown in a further set of buttons. In addition, this can be used for actions that can be applied to almost any type, such as getting information about the file. This can also be configured by each user.

1.1 System requirements

The full functionality requires Motif 1.1 or Motif 1.2. It will not run under Motif 1.0 because it uses `XmStringCreateSimple()` and assumes that `Form` works. It has only been tested on a Sun SparcStation using Solaris 1.0, using the GNU and Sun compilers.

1.2 Installation

You will have received this system through InterNet, through email, or through some tape distribution. This documentation file will have been part of that distribution. To read this, you will already have uncompressed, un–shar'd or un–tar'd the files, and sent this file to a PostScript printer. `xmfm` uses the `imake` system to control building and installation. It uses a file called `Imakefile` for this.

In most cases you will not have to modify the `Imakefile`. The file `DEFINES` tells you about some of the flags that may be set. Other options that may be changed are, for example, the compiler used.

Once the `Imakefile` has been set up to your satisfaction, run `xmkmf` to create a `Makefile`, and then

```
make
make install
make install.man
```

to create executables and install the system. You may need to be logged on as the system supervisor to do this.

Chapter 2 Using xmfm

2.1 xmfm window areas

Each pane of `xmfm` consists of a menu bar along the top, two sets of buttons down the left, and three directory display areas down the right. On the left, the top set of buttons displays actions that can be carried out on selected files, programs and directories. This set is called the 'file toolbar.' The bottom set displays a fixed set of actions that can be performed at any time, and is known as the directory toolbar. The top right pane displays the set of programs (executable files) in the current directory; the middle pane shows the ordinary files; and the bottom pane show the directories. Two directories are nearly always '.' (the current directory) and '..' (the parent directory). These are shown by icons with labels HERE and an upward pointing arrow respectively.

2.2 Manipulating directories

If a directory is selected, a set of options for that directory is displayed in the file toolbar. The supplied configuration treats all directories in the same way, and there are only two options: to open a directory or to move to the directory.

2.2.1 Opening a directory

If a directory is opened, a new pane is created on the desktop. This pane shows the contents of the new directory. The new pane has its own menu, its own file, executable and directory areas, and its own toolbars. Interaction can be done with either the new or the old pane by clicking on it to select it. The separate panes provide separate interaction areas to different directories.

2.2.2 Move to a directory

If a directory is moved to, the current contents of the pane are replaced by the contents of the new directory. All actions for this pane are now performed within this directory. This is the default double–click action.

2.2.3 Home directory

You can move to your home directory, or open a new pane on it. This directory is the one you are placed in when you login, and forms a general reference point for many directory movements.

2.3 Manipulating programs

If a program is selected, a set of options for that program is displayed in the file toolbar. The supplied configuration gives two possible actions for each program: execute the command with no command line arguments, and execute the command with command line arguments. The two buttons for this show:

`execute`:	execute with no command line arguments
`execute...`:	execute with command line arguments

When a command is executed with command line arguments, a prompt dialog is called up to get the arguments.

`xmfm` is initially set up to distinguish between two types of files: those with names beginning with x (such as `xmfm`, `xclock`, `xterm`) and any others. Those beginning with x are assumed to be X Window programs, and are directly executed. The rest are assumed to be non–graphical programs and are executed within an xterm window. While this works correctly for most programs, it will get some wrong. For example, the Motif window manager `mwm` should not be executed within an `xterm`, whereas the command line interface program `xargs` should be run within an `xterm`. Methods for avoiding this are dealt with in the chapter on ‘Configuring xmfm.’

2.4 Manipulating files

If a file is selected, a set of options for that file is displayed in the file toolbar. The supplied configuration distinguishes between a number of different file types, based on a name pattern. The details of this are considered in the chapter on ‘Configuring xmfm.’

Most files can be edited and viewed. When the edit action is selected, the editor `vi` is brought up within an xterm window for this file. When the view action is selected, the program `more` is run on the file within an `xterm` window. The choices of editor and viewer are configurable: see the chapter on ‘Configuring xmfm.’ The default double–click action for many files is to start the editor.

A `Makefile` can be edited and viewed. Two additional actions are to run the program make using the `Makefile`, and to run the program `make` with arguments using the `Makefile`.

2.5 Manipulating panes

2.5.1 Refresh the displayed directory

Programs that you run may make changes to a directory, by, say, removing files. This may make the displayed lists of files incorrect. After each command is executed `xmfm` attempts to spot these changes and to correct for them, but it is unable to spot all changes. The refresh command forces `xmfm` to update the lists of files and redisplay any changes.

2.6 The xmfm menus

The `xmfm` menus are for major actions that generally are not performed as often as the actions that show up in the toolbars. For example, creating a new file or directory, or moving to an arbitrary directory.

2.7 Scenario: the edit/compile/execute/debug cycle

A typical scenario is to attempt to build an executable file from C source files, possibly using a `Makefile`. This involves editing source files, running `make` and running the resultant executable under a debugger. To do this, start `xmfm` in any directory. Move to the directory in which this project is being performed. Select a source file that needs changes and start an editor on it. Perform any edits that are needed, and save the file (you have to remember to save it before doing the next steps). Leave the editor running, because there is a chance that you will have to return to the editor to fix up syntax or semantics errors. Next select the `Makefile` and run `make` on it to build the entire project, or `make...` to invoke `make` with an argument. `make` will run within its own `xterm`, so you can see the results of execution. If there are syntax errors, return to an editor window for the source file, fix any errors, and run make again.

Eventually, an executable file will be created. By this time, it is quite likely that the directory contents will have changed – `make` will have created object modules and the executable file. However, `xmfm` will have caught these changes as they occurred, and will have updated the directory display accordingly.

Select the executable, and press the debug button. A debugger will start, and you can test the program. If an error occurs, you can switch back to an editor window to fix the error, run `make` again, and debug again.

Chapter 3 Drag and drop

`xmfm` supports drag and drop activities. A file, program or directory that is selected may be dragged to various places and dropped there. `xmfm` in fact supports three models, and the one that is available will have been chosen at compile time. In the first model, drag and drop is disabled and does not work.

The second model uses Roger Reynolds' drag and drop library. With this mechanism, a file can be dragged by the left mouse button. A visual indication occurs when the mouse is moved out of the icon for the selected file. The file is ''dropped'' by releasing the left button.

The third model uses the Motif 1.2 drag and drop library. With this mechanism, a file can be dragged using the middle button. A visual indication is given as soon as the mouse button is pressed. This model is not recommended at present because it severely slows down the rate at which the display is refreshed when moving between directories.

The action that takes place depends on where the icon is dropped.

3.1 Dropped on a toolbar button

When the mouse button is pressed, the actions for the selected file show up in the toolbar. If the icon is dropped on any of the toolbar buttons, the action for that button takes place.

3.2 Dropped on a directory [not yet implemented]

If an icon is dropped on a directory button, a copy is made in that directory.

3.3 Dropped on a program [not yet implemented]

If an icon is dropped on a program, `xmfm` tries to see if it can run the program with the selected file as an argument. A search is made of the actions list for the program for an action which expects a command line argument. The selected file is then used as `$1` for this command. If no action exists, the program is executed with the selected file as the only argument. `xmfm` attempts to determine whether or not a program needs to be run in an `xterm`.

3.4 Dropping on different panes [not yet implemented]

`xmfm` allows you to drag files from one directory pane and drop them on another. Here are some possible uses for this:

To copy a file from one directory to another, open panes on both the source and destination directories. Drag files from one to the other.

Open a pane in `/bin` (this will take some time). Dragging files and dropping them on programs here will execute them.

Chapter 4 Configuring xmfm

4.1 Configuration file

When you start `xmfm` it will check for the existence of a configuration file. This file is `.xmfmrc` in your home directory. If the file does not exist, `xmfm` will use the default one. The default file is `/usr/lib/X11/app-defaults/xmfmrc`. It will read configuration information from the file. If you wish to modify the configurations used by xmfm, copy the default file to your Home directory as `.xmfmrc`.

The file `.xmfmrc` may be changed by you to alter the behaviour of `xmfm` by customising the file patterns and the actions for each pattern.

Part of the default configuration file is:

```
directory_pattern   *
file_pattern                [^.]*
executable_pattern  *
{         type              f
          pattern   [mM]akefile
          pixmap    makefile.xbm
          description Makefile
          {
                    label     make
                    run_in_xterm
                    pause_after_exec
                    action    "make -f $0"
          }
          {
                    label     edit
                    run_in_xterm
                    action    "vi $0"
          }
}
{         type      f
          pattern   *.c
          pixmap    ccode.xbm
          description "C source code"
          {
                    label     edit
                    run_in_xterm
                    action    "vi $0"
          }
          {
                    label     "to object"
                    run_in_xterm
                    pause_after_exec
```

```
                action    "cc -o ${0%.c} $0"
        }
}
{       type      f
        pattern   *
        pixmap    file.xbm
        {
                label     edit
                run_in_xterm
                action    "vi $0"
        }
        {
                label     view
                run_in_xterm
                action    "view $0"
        }
}
{       type      x
        pattern   x*
        pixmap    prog.xbm
        program "X program"
        {
                label     execute
                action    "$0"
        }
}
{       type      x
        pattern   *
        pixmap    prog.xbm
        {
                label     execute
                run_in_xterm
                pause_after_exec
                action    "$0"
        }
}
{       type      d
        pattern   .
        pixmap    folder.xbm
        description "Current directory"
        {
                label     refresh
                action    "$cd $0"
        }
        {
                label     open
```

```
                action    "$opendir $0"
        }
}
{       type      d
        pattern   ..
        pixmap    dotdot.xbm
        description "parent directory"
        {
                label     chdir
                action    "$cd $0"
        }
        {
                label     open
                action    "$opendir $0"
        }
}
{       type      d
        pattern   *
        pixmap    folder.xbm
        {
                label     chdir
                action    "$cd $0"
        }
        {
                label     open
                action    "$opendir $0"
        }
}


{       type      p
        pattern   *
        pixmap    ""
        {
                label     make
                run_in_xterm
                pause_after_exec
                action    make
        }
        {
                label     information
                run_in_xterm
                pause_after_exec
                action    "ls -l $0"
        }
        {
```

```
                    label     remove
                    action    "rm $0"
          }
}
```

4.2 Format of the configuration file

The configuration file is made up of a preamble part which is a set of lines applicable to the whole application, and parts enclosed in curly brackets { }. Each of these parts gives the patterns and actions for classes of files. Within each section a preamble part gives information on each class. This is followed by sets of actions, also enclosed in curly brackets.

4.3 Format of preamble

The preamble section can be used to specify resources that apply to the entire application. At the moment, this consists of the initial patterns that are used to filter the display of files. This consists of lines (in any order)

```
file_filter          pattern
directory_filter     pattern
executable_filter    pattern
```

The pattern uses the file pattern matching of shells such as the Korn shell, zsh and bash. The following table gives the patterns;

`*`	match any sequence of characters
`?`	match any single character
`[...]`	match any one of the enclosed characters
`[^...]`	match anything except one of the enclosed characters

For example, the pattern for files is `[^.]*`, which means to match anything that does not begin with a '.'. (This is the usual display for the Unix file lister `ls`.)

4.4 Format of file class

The format is

```
{         type pattern
          name pattern
          pixmap
          description
          { action specification }
          { action specification }
          ...
}
{...}
```

4.5 Type patterns

`xmfm` distinguishes three types of file: directories of type 'd', executable files of type 'x', and ordinary files (i.e. everything else) of type 'f'. The type specification is one of

```
type d
type x
type f
```

In addition, xmfm uses the file type ‘p’ to describe the global actions in the lower toolbar.

4.6 Name patterns

The names of files, directories and executables may be given using Bourne shell patterns. That is any character stands for itself, except for the following pattern matching characters:

`*` matches any string (including the null string)

`?` matches any single character

`[...]` matches any one of the enclosed characters

For example:

`[Mm]akefile` matches either `Makefile` or `makefile`

`*.c` matches any file ending in `.c`

`*` matches any file

When a file is selected by the user, an attempt is made to match it to each pattern of the configuration list. The first match found with the correct file type is used to set the action list.

For example, given the list

```
{         type f
          name [Mm]akefile
          ... }
 {        type f
          name *.c
          ... }
 {        type f
          name *
          ...
 }
```

A `makefile` or `Makefile` matches the first pattern, any ordinary file ending in `.c` matches the second line, and any other ordinary file matches the third line.

4.7 Pixmaps

`xmfm` uses bitmaps to show file types, as well as showing the name of the file. The bitmap used is specified by the form

```
pixmap pixmap-name
```

xmfm uses a standard Motif algorithm system to locate bitmap files. In the simplest case, the following files are looked for:

```
pixmap-name
$HOME/bitmaps/XmFm/pixmap-name
```

```
$HOME/bitmaps/pixmap-name
$HOME/pixmap-name
/usr/lib/X11/bitmaps/XmFm/pixmap-name
/usr/lib/X11/bitmaps/pixmap-name
/usr/include/X11/pixmap-name
```

If shell environment variables `$LANG`, `$XMBMLANGPATH` or `$XAPPLRESDIR` are set, extra paths are added to this. See the Motif Programmers Reference Manual under the entry for `XmGetPixmap`.

4.8 Description

The format of the description line is

```
description text
```

where text is any sequence of characters. If this contains spaces, then it should be enclosed in double quotes. This is used by the information button to give additional information.

4.9 Actions

Each file has an associated set of actions. These are listed after the file name patterns. The default action (the one that will be executed on a double click) is first in the list. Each action specification consists of a a name, a set of runtime options and an action.

4.9.1 Action label

This is the label that will appear in a toolbar button, and should be chosen to convey in a short space what the action is. For example, ‘edit’, ‘vi’, ‘compile’, etc. If the action needs a prompt to complete the command, a simple convention is to follow the action name with three dots, as in ‘move to...’. For example,

```
label edit
label view
```

Multiline labels may be created by enclosing the whole phrase in quotes and carrying it onto the next line. For example,

```
label "to
executable"
```

4.9.2 Runtime options

The runtime options control:

– whether the command is run in a terminal emulator window or not. if the word `run_in_xterm` appears in the action specification, then it is run in a terminal emulator.

– whether the command requests a <return> key press after termination. This only has meaning if the command is run in a terminal emulator, and is used to control if the window is maintained after the action has completed. For example, when a text editor completes, there is no need to maintain a window for it. On the other hand, when a compile completes, the output window should be left on–screen until the user signals that they no longer need it. Otherwise, all error messages, etc would just disappear! If the word `pause_after_exec` appears in the action specification, then this option is in effect.

– whether or not the user needs to enter a prompt string. For example, if the action is to move or rename the file, the user needs to be prompted for the new name. If a prompt string appears in the action specification, then a prompt will be used. For example:

```
prompt "Enter the name of the new file"
```

Note that if the string contains more than one word, it must appear in double quotes.

4.9.3 Actions

Finally, the last part of an action specification is the action itself. This is a normal Unix command. It must appear in double quotes if it contains more than one word. Before executing it, any patterns are expanded out. The patterns are simple forms of the patterns used in the newer shells such as the Korn shell, the Bourne Again Shell and the Z shell. The basic patterns are:

`$0` the name of the file selected

`$*` the entire prompt string (if there is no prompt, this is the NULL string)

`$1` the first word in the prompt string

`$2` the second word in the prompt string

`$9` the ninth word in the prompt string

The backslash character '\' can be used to escape any character.

More complex patterns use the positional parameters `$0` to `$9`, but they can be modified

`${1}` the first word in the prompt string

`${2}` the second word in the prompt string

`${9}` the ninth word in the prompt string

If the positional parameter is followed by '#pattern' for some pattern, the pattern is removed from the start of the value of the parameter. If the positional parameter is followed by '%pattern' for some pattern, the pattern is removed from the end of the value of the parameter. For example:

To compile C source files, the following pattern will remove the trailing '.c' from the selected file.

```
cc -o ${0%.c} $0
```

RCS encoded files are terminated by ',v'. To extract a file from RCS, the command

```
co -l ${0%,v}
```

will checkout the selected file.

SCCS file begin with 's,'. To use SCCS on such a file,

```
sccs ${0#s,}
```

To move a file using a pattern

```
mv $0 $1
```

will use the first name in the prompt string as the destination file.

4.10 Builtin actions

A number of actions are performed internally by `xmfm`. The actions currently recognised are

`$chdir`	change to the selected directory
`$opendir`	open a new pane pane on the selected directory
`$information`	show the full details on the selected file or directory in a popup window, such as size, ownership, etc. File permissions can be edited.

4.11 xmfm Resources

`xmfm` is built using the Motif and Xt libraries, and so has full support for the resource database mechanism. For example, to control the display used and the foreground and background colours, run `xmfm` by the command line

```
xmfm -display pandonia:0 -fg red -bg green
```

Further use of the resource database requires knowledge of the ‘‘widget tree’, that is the tree of objects that make up `xmfm`. This tree can be shown dynamically using the program `editres` if `xmfm` has been compiled appropriately. The widget tree for `xmfm` is:

```
XmFm  xmfm
    XmMainWindow  main
        XmSeparatorGadget  Separator1
        XmSeparatorGadget  Separator2
        XmSeparatorGadget  Separator3
        XmForm  form
            XmLabel  directory_label
            XmRowColumn  files_toolbar
                XmPushButton  file_toolbar_button
            XmRowColumn  dirs_toolbar
                XmPushButton  dir_toolbar_button
            XmPanedWindow  pane
                XmPanedWindow  exec_rc
                    XmLabel  executable_filter_label
                    XmScrolledWindow  exec_sw
                        XmDrawingArea  ScrolledWindowClipWindow
                            XmRowColumn  executables_pane
                                XmDrawnButton  dumm
                                XmDrawnButton  exec_button
                        XmScrollBar  VertScrollBar
```

```
                XmScrollBar  HorScrollBar
            XmSash  sash
            XmSeparatorGadget  separator
            XmSash  sash
            XmSeparatorGadget  separator
        XmPanedWindow  file_rc
            XmLabel  file_filter_label
            XmScrolledWindow  file_sw
                XmDrawingArea  ScrolledWindowClipWindow
                    XmRowColumn  files_pane
                        XmDrawnButton  file_button
                XmScrollBar  VertScrollBar
                XmScrollBar  HorScrollBar
            XmSash  sash
            XmSeparatorGadget  separator
            XmSash  sash
            XmSeparatorGadget  separator
        XmPanedWindow  dir_rc
            XmLabel  dir_filter_label
            XmScrolledWindow  dir_sw
                XmDrawingArea  ScrolledWindowClipWindow
                    XmRowColumn  dirs_pane
                        XmDrawnButton  dir_button
                XmScrollBar  VertScrollBar
                XmScrollBar  HorScrollBar
            XmSash  sash
            XmSeparatorGadget  separator
            XmSash  sash
            XmSeparatorGadget  separator
        XmSash  sash
        XmSeparatorGadget  separator
        XmSash  sash
        XmSeparatorGadget  separator
        XmSash  sash
        XmSeparatorGadget  separator
XmRowColumn  menu_bar
    XmCascadeButton  File
    XmCascadeButton  Goto
    XmCascadeButton  Run
    XmCascadeButton  Filter
    XmCascadeButton  Help
    XmMenuShell  popup_menu_pane
        XmRowColumn  menu_pane
            XmPushButton  New file...
            XmPushButton  New dir
            XmPushButton  Copy...
```

```
            XmPushButton  Move...
            XmPushButton  Close
            XmPushButton  Exit
        XmRowColumn  menu_pane
            XmPushButton  Chdir Home
            XmPushButton  Chdir Directory...
            XmPushButton  Open Home
            XmPushButton  Open Directory...
        XmRowColumn  menu_pane
            XmPushButton  Command...
            XmPushButton  Xterm
        XmRowColumn  menu_pane
            XmPushButton  Files
            XmPushButton  Executables
            XmPushButton  Directories
        XmRowColumn  menu_pane
            XmPushButton  XmFm
            XmPushButton  Mouse
            XmPushButton  Keyboard
            XmPushButton  Filters
            XmPushButton  Bugs
            XmPushButton  About
```

Resources of individual objects can then be controlled from resource files such as `.Xdefaults`. For example, the set the foreground colour of every button in the files toolbar, add this line to `.Xdefaults`:

```
xmfm.main.form.files_toolbar.file_button.fore-
ground:  red
```

(This should all appear on one line with no hyphenation.) This may be simplified by use of resource patterns, e.g.

```
xmfm*file_button.foreground:  red
```

The full set of resources may be found in the Motif Programmers Reference Manual, or queried interactively using editres. The tree given above also shows the class of each object (for example `XmMainWindow` or `XmPushButton`) under which the Reference Manual describes the options.