Rodent Filemanager
User's Guide

September 22, 2011

Abstract

In the unix world, files are not just ordinary files. Shared memory, devices, processes, and practically everything in the unix environment has its place in the file structure. The goal of Rodent Filemanager is to provide a small, fast and powerful filemanager capable of handling this multifacet environment. And the chosen means of doing this in the new multicore environment is with multiple light weight processes running simultaneously in a coordinated fashion.

Contents

1 History

The Xfce Desktop Environment started as a CDE clone using the XForms toolkit (hence the XF and ce in XFce). CDE was the "Common Desktop Environment" developed by HP, Sun, Digital and other big box players for their top notch workstations. Definitely cool. Definitely not for dummies.

CDE had a neat filemanager called dtfile, but this was not open source, so Xfce used a variation of Rasca's "Xtree" called "Xftree" as a filemanager.

Xffm history began when Xfce was running on the Gimp Toolkit (GTK) by extending Xftree's functionality with the "glob" search engine (now "fgr"), the samba network browser "Xfsamba", the differences viewer "Xfdiff" and a set of dtfile icons, among other things.

Then came the migration to GTK-2. For this Xftree was rewritten as Xffm, retaining all the previous functionality and adding more. The dtfile icon set was replaced by Francois' Rodent icon theme. To distinguish Xffm as a part of Xfce, the first release was tagged 4.0.

Then came 4.2 and the love/hate story started. Those new to Xfce could not understand why Xffm had such a steep learning curve, while those familiar to Xftree expected the nerdy behaviour they had become used to.

Anyways, it was decided that Xfce would no longer distribute a filemanager: this way users could choose the new Thunar filemanager (which, as an independent software, would no longer follow the Xfce version numbers).

Cut loose of the xfce umbilical, xffm was adopted by foo-projects, hosted at Stockholm University. The xffm.org domain was assigned and the web page at http://xffm.org/ was put on the line. The next release of Xffm would no longer follow the current Xfce version numbers, but for consistency would not be rolled back.

Xffm-4.5.0, which featured a new user interface (although the old nerdy treeview was still available) was released on 2006-05-24. This remained stable until 2010, when 4.6.0 was released[1]. 4.5.0 had 13500 downloads according to sourceforge data, and the main user group was in Germany, followed by the United States, Russia, China and Poland.

4.6.0 was to ve a new thread-safe, thread-based design (code named "Rodent", for Francois' icon theme) and required a rewrite of the Xffm code. To accomplish this, several goals were set forth:

- Thread based, thread safe design
- Cleaner and simpler build process
- Faster and more powerful
- Unnerdification (this a bit surreal, coming from a nerd)
- Command line functionality (lp-terminal)

[1]Work on 4.6.0 began in early 2006, when Intel gave me a multicore laptop to start doing multicore software (thanks, Intel).

As part of the unnerdification, the old treeview is gone. To keep it simple, the graphic part is as simple as the concept developed by the guys at Palo Alto (you know, the guys who first came up with the idea of a mouse —a species of rodent– for the computer). Nothing more. Simple uncluttered graphics. Along side this graphic part is the keyboard. A place to issue commands. Graphic, but not for dummies.

The first release of Rodent Alpha, (aka xffm-4.6.0) was in November 2010, at a conference at the University of Sonora. The second release was stable enough to be tagged beta software. Rodent Beta (aka xffm-4.6.2) released in April 2011. The third release, Rodent Beta release 2 (aka xffm-4.6.4) was released in the last days of May. The first non beta release is Rodent Gamma.

Rodent Gamma is faster than Beta, has over 95% translation for the main non-English languages in the world, and boasts new features that make this filemanager capable of handling heavy loads. Visit the sourceforge site[2] for download details.

2 Rodent Design

2.1 Structural layout

The Rodent Filemanager source file is extremely simple, consisting of less than 500 lines of code. Yet this expands to more than 40,000 lines of code on a binary tree design modelled on Hegel's comprehensive philosophical framework[3]. If you are not familiar with this manner of thought, fret not, the overall method should become clear as you read on.

On the highest level, Rodent Filemanager is the synthesis of:

- the main application,
  - popup menus
  - lp-terminal area
- helper programs:
  - internal,
  - external.

The main application is divided into two mayor areas, each made up from two components. The nature of the function determines where the code will be contained.

- Dynamic load libraries

[2]http://sourceforge.net/projects/xffm/

[3]Hegel's account of reality revolutionized European philosophy. In particular, he developed the concept that mind or spirit manifested itself in a set of contradictions and oppositions that ultimately integrated and united, without eliminating either pole or reducing one to the other (http://en.wikipedia.org/wiki/Georg_Wilhelm_Friedrich_Hegel).

  - Internal
    - Rfm library (*librfm*: low level library, where functions are namespaced with the *rfm_* prefix);
    - Rodent library (*librodent*: high level library, where functions are namespaced with the *rodent_* prefix)
  - External
    - Gtk+
    - libmagic

- Modules (or plugins).
  - functional plugins (which provide extra filemanager functionality)
  - root level plugins (which provide alternate filestructures)

2.2 Geometric layout

The Rodent filemanager physical layout is composed of the icon and lp-terminal areas.

The first may have icons or empty space. The popup menu is context sensitive, allowing for two basic menus. The icon area is meant to be as simple as possible, in the manner of the concept developed by the guys at Palo Alto. Simple uncluttered graphics (see figure 1). As such, toolbar and file menu functionality is fully handled by popup menus which are only visible when summoned. Display real estate is too valuable to waste on alternate methods to do the same thing.

The lp-terminal area is also simple: it is not a video terminal emulator, but a line printer terminal emulator (hence the lp-terminal nomenclature), such as those used as consoles in several robust computing systems of yore (see figure 2). The lp-terminal area will execute all command in background, and has two types of autocompletion, history[4] and bash[5], and two text areas, one for input and one for output. Thus it is handy to issue commands like "make" and follow the output which is streamed to stdout/stderr[6], but is not adequate to edit a file, for example. There are better graphic tools for the latter.

Rodent filemanager geometric layout can be summarized as follows:

- icon area
  - icon popup menu
  - empty space popup menu
- lp-terminal area

[4]summoned with the CTRL+TAB key combination
[5]summoned with the TAB key
[6]Output to stderr will be colored red, while that to stdout will be black.

Figure 1: First computer with icons and pointer device

- output text area
  - stdout (black font)
  - stderr (red font)
- Input text area
  - history autocompletion
  - bash autocompletion

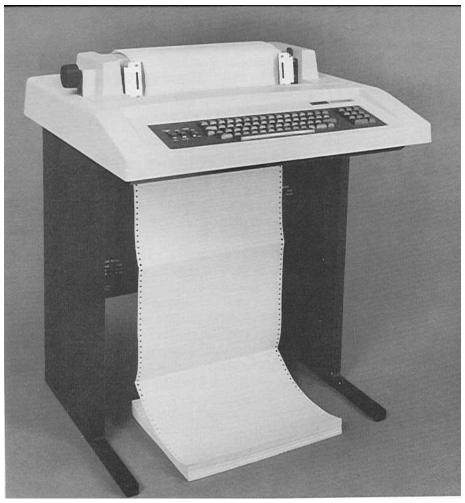


Figure 2: Lp terminal

3 Basics

3.1 Mouse click

Items are activated with a single click. This is a configurable option.

3.2 Drag and drop

The default action for drag and drop is to move the items. This is a configurable option. Holding down the CONTROL key while releasing the mouse button will do a copy. Holding down the SHIFT key will do a move. Holding down both the CONTROL and SHIFT keys will do a symlink. Symlinks constructed in this manner will be done so in a relative file path, not an absolute path.

3.3 Cut and paste

The Rodent pasteboard operates on the localhost. Windows from remote hosts will have a different pasteboard. This change of behavior from previous X-pasteboard in version <= 4.5.0 is required for compatibility with the threaded multicore nature of Rodent.

3.4 Fgr

`fgr` is the command line search tool used by Rodent and Rodent-fgr. The command line tool may be used directly from a terminal or the lp-terminal input line. The *GNU grep* command (or similar BSD tool) must be installed for this application to search for file content. There is no need for the *find* command to be present.

3.5 Application association

To have Rodent associate an application to a particular file mimetype, use the “Open with” element of the popup menu. The selected application will be the default application until the “Open with” element is used again. Other applications associated to the particular mimetype will always appear in the popup menu as individual items.

3.6 Selection

Items may be selected by clicking with the mouse button while holding down the SHIFT or CONTROL key in the conventional manner. SHIFT will select all items between previously selected items, while CONTROL will select if unselected and unselect if previously selected. All other items will retain their selection status. A rubber band box may also be used for selecting multiple items by holding down the mouse button while moving the mouse.

3.7 Keyboard

Keyboard input is processed by the lpterminal. Bash autocompletion is available with the TAB key. History autocompletion is available with CONTROL+TAB key.

3.8 Lp-terminal command history

To access the command history of the lp-terminal you may use the following commands in the lp-terminal:

```
?                Show help about options
!                Show History
!n               Reference Number (Command Line) n.
!STRING          Complete Match STRING.
!?STRING         Anywhere STRING.
STRING<CTRL+TAB> Completion mode: Command Line.
STRING<TAB>      Completion mode: bash.
!!               Clear History (Current)
!!!              Clear History (Disk)
```

All lp-terminal commands will show stdout output in black and stderr output in red in the lp-terminal area. For each command (executed in background) a “exec” button appears next to the lp-terminal input area and which is linked to rodent-ps. This allows the user to view the process in question and send arbitrary signals for control. If the ps-module plugin is not installed, then a dialog which allows for user termination of the process will appear.

3.9 Content emblems

For each folder, a small icon indicating the prevailing mimetype for the files contained in that folder will appear in the upper right hand corner.

3.10 Preview popups

For image, pdf, postscript and text files, placing the mouse on top of the icon will trigger a popup window which will show the content of the file in question in a size big enough to actually see what the file is about. This option is especially useful for sifting through files with non-descriptive names without having to open each one. To make this as fast as possible, move the mouse over several files before halting over each one. This will allow Rodent to do the preview work in the background while the user is examining the output produced. To disable the preview and to show a normal tip with file information, press mouse button 2 (the middle button) while moving the mouse on top of the icon. To disable previews for all files in the current directory, toggle the "show previews" value in the popup menu. To disable all popup windows for all directories, use the settings dialog configuration tool.

3.11 Touch

Rodent provides a graphic front end to GNU touch program. This can be launched from the popup menu on a single or multiple selected items.

3.12 Diff

Rodent also includes a front end to GNU diff command, allowing a side by side comparison of files and patch generation in a variety of formats.

3.13 Icon theme

Rodent includes the updated Rodent icon theme which may be installed as the default icon theme for desktop environments such as Gnome or Kde.

3.14 Translations

Rodent includes translations for 126 languages[7].

[7]af, am, an, ar, as, ast, az, be, be@latin, bg, bn, br, bs, ca, ca@valencia, crh, cs, csb, cy, da, de, dv, dz, el, en_CA, en_GB, eo, es, et, eu, fa, fi, fr, fur, fy, ga, gl, gn, gu, gv, ha, he, hi, hne, hr, hsb, hu, hy, ia, id, ig, io, is, it, ja, ka, kg, kk, km, kn, ko, ku, ky, la, lb, li, lt, lv, mai, mg, mi, mk, ml, mn, mr, ms, nb, nds, ne, nl, nn, no, nso, oc, or, pa, pl, ps, pt, pt_BR, pt_PT, ro, ru, rw, se, si, sk, sl, sq, sr, sr@ije, sr@ijekavian, sr@ijekavianlatin, sr@latin, sv, ta, te, tg, th, tk, tr, ug, uk, ur, uz, uz@cyrillic, vi, wa, xh, yi, yo, zh, zh_CN, zh_HK, zh_TW.Big5, zh_TW, zu.

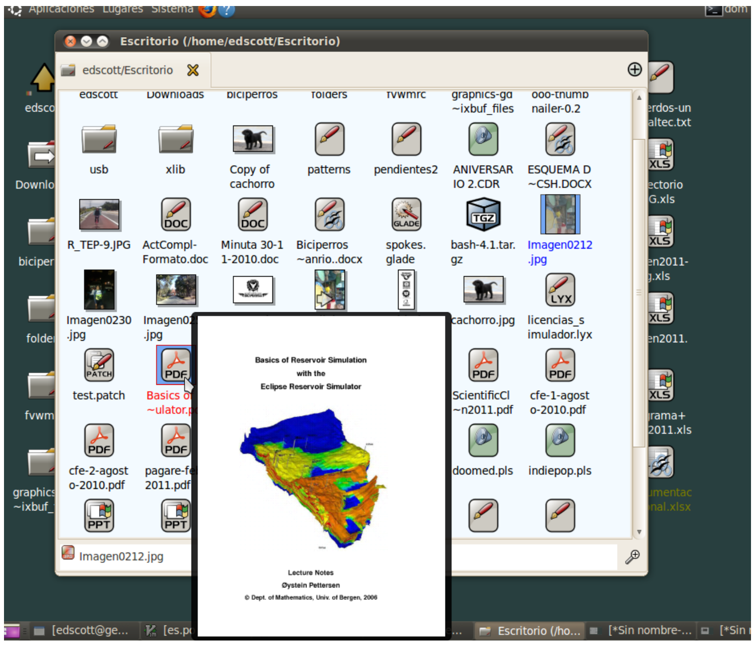

Escritorio (/home/edscott/Escritorio)
edscott/Escritorio
Basics of Reservoir Simulation
with the
Eclipse Reservoir Simulator
Lecture Notes
Øystein Pettersen

Figure 3: Rodent iconview with Rodent icon theme showing a pdf file preview pop up

4 Main application

The rodent filemanager has different modes depending on how it is summoned from the command line.

4.1 rodent

This is the iconview filemanager. If the startup path is not specified on the command line, it will default to the user's home directory. Figure 3 shows the Rodent iconview when the mouse is held over a pdf file. In this case, a popup window will show the user the first page of the document. This makes sense for users who have many pdf files and wish to find one in particular without opening each one one..

4.2 rodent-forked

This is the same iconview filemanager as summoned by the rodent executable, but if the startup path is not specified on the command line, the program will default to the rodent root level (figure 7), where icons for the root file system, the user's home directory, and all installed root level plugins are displayed. Furthermore, the application will execute in background.

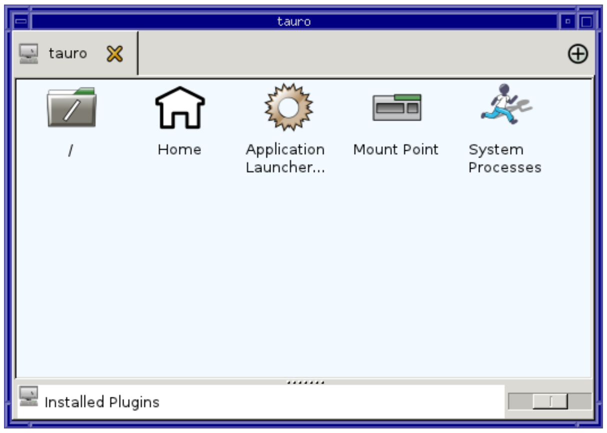


Figure 4: Rodent root level

4.3 rodent-desk

This is the fixed window on the root background. Desktop background color can be customized by the user, as well as the use of an image file in many different formats, such as jpg, gif, png. Only the icons which fit on the desktop will be displayed. As such, if the number of icons exceeds the fit capacity, some will not be shown. In figure we show the Rodent desktop. In this figure the selected icon theme is the default GTK desktop icons. If the *icons* plugin happens to be removed from the system, Rodent will fall back to a limited set of GTK stock icons, which may be a subset of the GTK desktop icons if the program is being executed within a desktop environment.

The deskview, as any other iconview, can be defined in navegation mode, allowing the user to cruise the file system or root plugins as if in an ordinary iconview (this feature is disabled by default, and can be enabled at Personal_settings -> Desktop -> Navegation_mode).

Other characteristics that can be defined for the deskview, shown in figure 6, are:

- Show the desktop on executing any *Rodent* instance (default: yes)
- Show lp-terminal text output window with buttons (default: no)
- Use navegation mode (default: no)
- Define background image (default: none)
- Define background color (default: #283F3F)
- Define top margin where no icons are placed (default: 20 pixels)
- Define bottom margin where no icons are placed (default: 40 pixels)

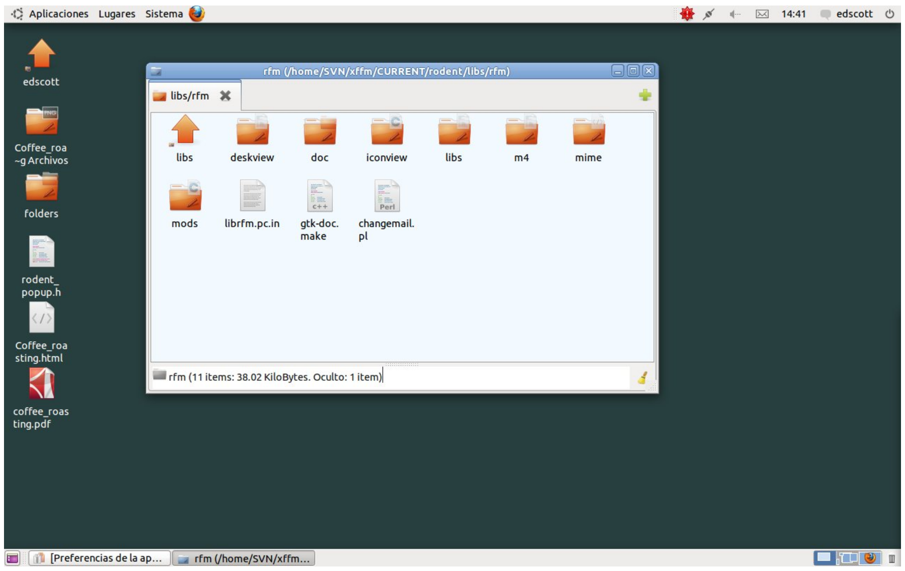
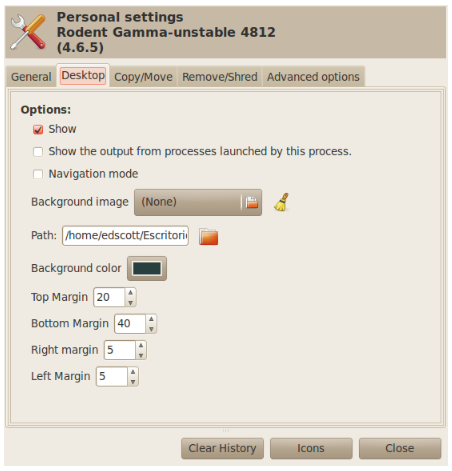


Figure 5: Rodent desktop with gtk stock icons on Ubuntu 9.10



Personal settings
Rodent Gamma-unstable 4812
(4.6.5)

General | Desktop | Copy/Move | Remove/Shred | Advanced options

Options:

- [✓] Show
- [ ] Show the output from processes launched by this process.
- [ ] Navigation mode

Background image (None)

Path: /home/edscott/Escritori

Background color

Top Margin 20

Bottom Margin 40

Right margin 5

Left Margin 5

Clear History | Icons | Close

Figure 6: Rodent-desk user settings dialog

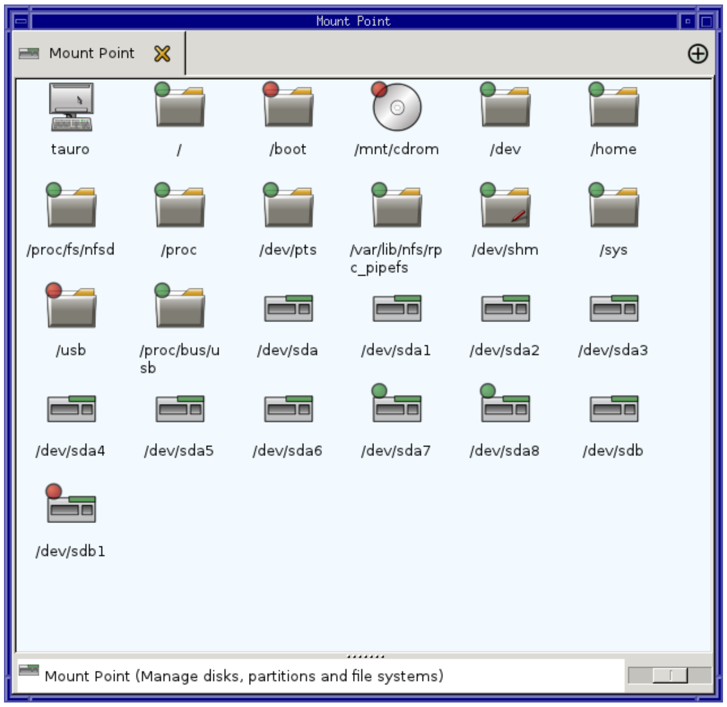


Figure 7: Rodent-fstab

- Define right margin where no icons are placed (default: 5 pixels)
- Define left margin where no icons are placed (default: 5 pixels)

4.4 Fstab plugin (rodent-fstab)

This executable will directly open the *Mount Points* plugin. From this plugin, mount points and devices defined in the system `/etc/fstab` file may be mounted or unmounted. Partitions detected in `/proc/partitions` will also be available for mount or unmount operations. Furthermore, any mount point detected in `/etc/mtab` will also be available (figure 7). Icons for mounted volumes are enhanced with a green ball emblem, while unmounted items have a red ball emblem.

When a usb device is inserted, the detected partition will automatically appear in this window, allowing the user to mount and unmount the device at will. If the fstab configuration do not allow users to mount or unmount a specified volume, Rodent will try to issue the command specified by the user with the *sudo* helper application (section 7.2.5).

When this module is installed, an additional menu element is available in the Rodent icon popup menu which allows the user to mount volumes listed

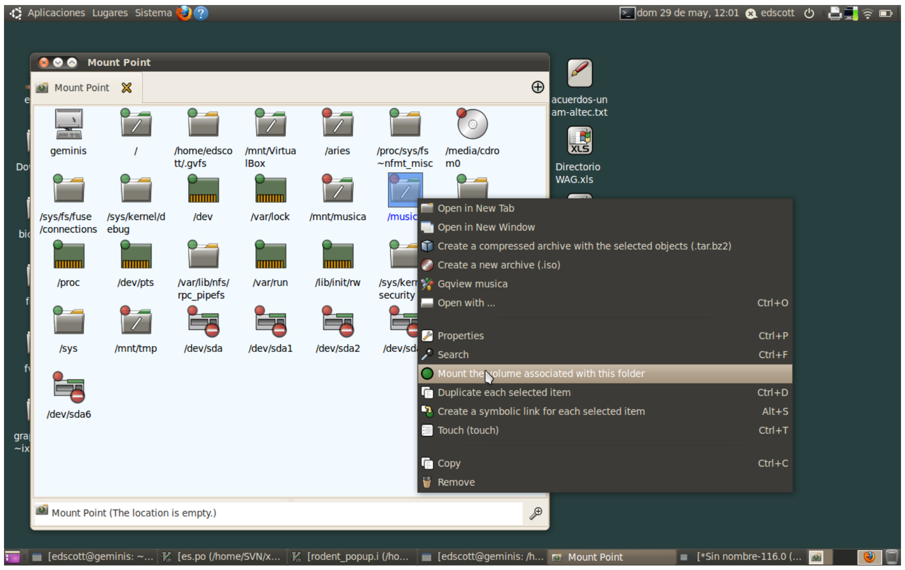

Mount Point
Open in New Tab
Open in New Window
Create a compressed archive with the selected objects (.tar.bz2)
Create a new archive (.iso)
Gqview musica
Open with ... Ctrl+O
Properties Ctrl+P
Search Ctrl+F
Mount the volume associated with this folder
Duplicate each selected item Ctrl+D
Create a symbolic link for each selected item Alt+S
Touch (touch) Ctrl+T
Copy Ctrl+C
Remove
Mount Point (The location is empty.)

Figure 8: Popup menu mount option

in `/etc/fstab` or unmount volumes listed in `/etc/mtab` (figure 8). The menu element will appear automatically if the action is applicable.

4.5 Ps plugin (rodent-ps)

The rodent-ps executable opens the System Processes plugin (figure 9). This plugin shows running processes in the system. The default is to show the parent-child relationship between processes in a file system manner. This behavior may be changed by means of the pop up menu in empty space (figure 10). The user may also select to view all sytem processes or only focus on own processes. Processes may be suspended, continued, terminated or killed by means of the icon area popup menu (figure 11). The complete set of signals that can be issued to any process are:

- `STOP`
- `CONT`
- `INT`
- `HUP`
- `TERM`
- `KILL`
- `USR1`

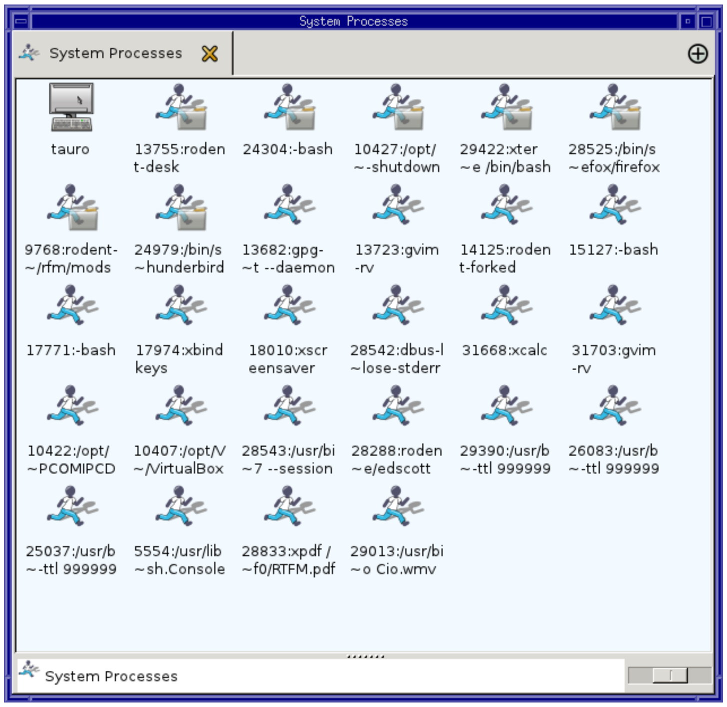


Figure 9: Rodent-ps

- `USR2`
- `SEGV`

The process tree and process information may also be obtained by means of the popup menu. And user processes may be reniced as well.

Furthermore, the window which appears when the mouse hovers over the icon shows further information about the process (figure 12). Whenever the *stop button* is pressed for any Rodent controlled background process, rodent-ps will open at the exact level where the process is running, allowing total control over the process. This is a feature available in versions greater than or equal to 4.6.6. The default behavior in previous versions was to send a `TERM` signal, and if that failed, a `KILL` signal. The old behavior for the *stop button* is now used only if the ps-module is not installed (see `./configure --help` for build configuration options).

The Rodent backend helper program, GNU ps is a contribution of

- Branko Lankester <lankeste@fwi.uva.nl>,
- Michael K. Johnson <johnsonm@redhat.com>,
- Michael Shields <mjshield@nyx.cs.du.edu>,

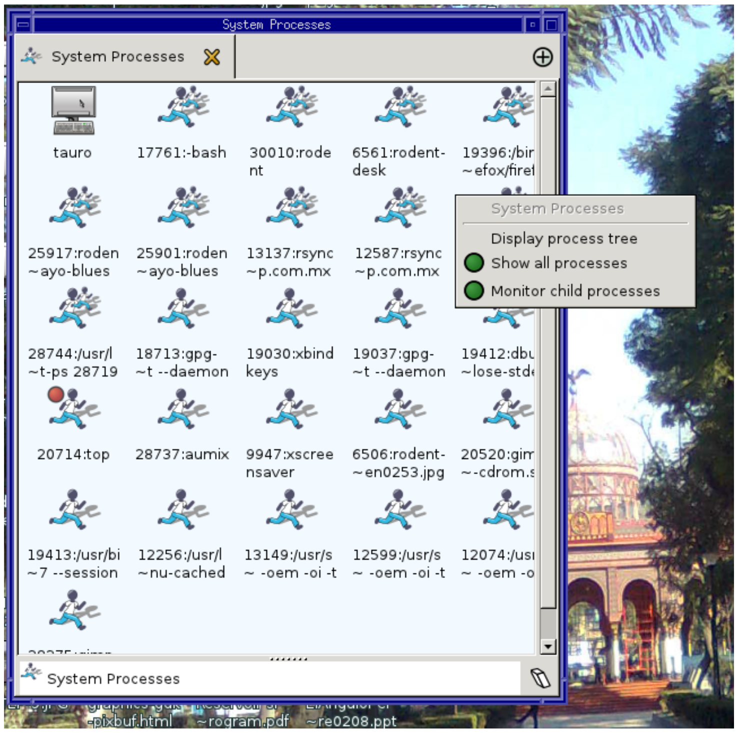


Figure 10: Rodent-ps: empty area popup menu

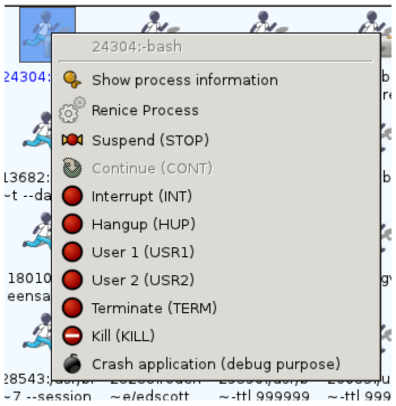
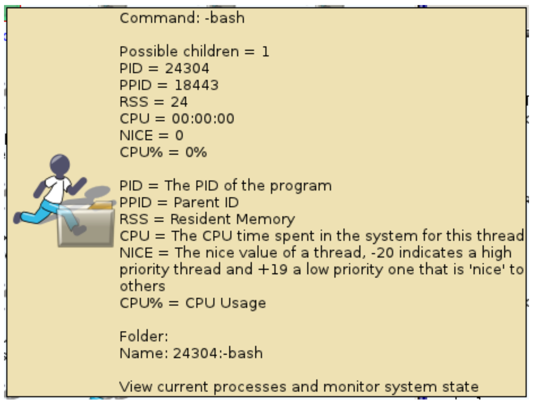


Figure 11: Rodent-ps: icon area popup menu



```
Command: -bash

Possible children = 1
PID = 24304
PPID = 18443
RSS = 24
CPU = 00:00:00
NICE = 0
CPU% = 0%

PID = The PID of the program
PPID = Parent ID
RSS = Resident Memory
CPU = The CPU time spent in the system for this thread
NICE = The nice value of a thread, -20 indicates a high
priority thread and +19 a low priority one that is 'nice' to
others
CPU% = CPU Usage

Folder:
Name: 24304:-bash

View current processes and monitor system state
```

Figure 12: Rodent-ps: tip window

- Charles Blake <cblake@bbn.com>,
- Albert Cahalan <albert@users.sf.net> and
- David Mossberger-Tang

for which we express our gratitude.

4.6 Dotdesktop plugin (rodent-dotdesktop)

The Rodent-dotdesktop executable summons the dotdesktop module in navigation mode.

4.6.1 Navigation mode

This module will create a categories window (figure 13), where each category will contain dot desktop files (launchers) for installed programs which have classified themselves accordingly.

The categories considered by rodent-dotdesktop are all the categories present in installed dot desktop files.

If any of the categories is empty, then the category will not appear. Each launcher icon can execute the referred program in two ways. The simplest is a simple double click (or <`return`> on a selected icon). The second method applies for drag and drop. For instance, if the user desires to open several files with a dot desktop application, all that has to be done is select the icons to be opened and drag them and drop them on the launcher icon. Since each launcher is simply a file, the launcher itself may be dragged and dropped on any filesystem folder or even the desktop folder. The behavior of the icon will be the same as within the Rodent-dotdestop view with one exception. When dot desktop files are located outside the rodent-dotdesktop view, they are also recognized as text files so the user can view and modify them.

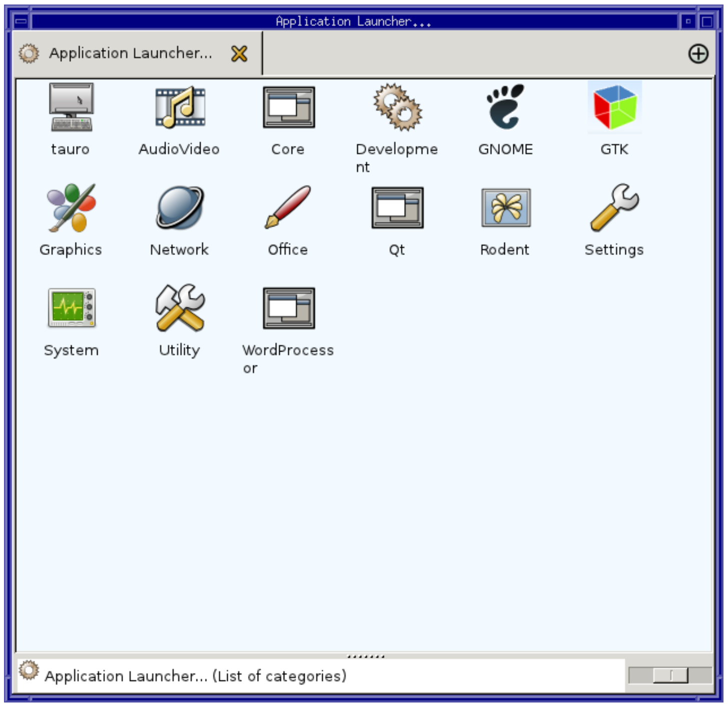


Figure 13: Rodent-dotdesktop

Icon customization The icon which is shown on the dot desktop file is specified within the dot desktop file itself. If this specification is not an absolute path, then the user may customize the icon to be shown by creating the desired icon, with the same name, and placing the icon in the `~/.local/share/pixmaps` directory. This icon will override the default icon.

Mime_information Dot desktop files also have pertinent mime information. This is not used in Rodent Gamma, but will be in the near future.

4.6.2 Popup menu

Furthermore, if the dotdesktop module is installed, the empty space popup menu in Rodent will have the extra submenu *Applications* with a small mouse icon. In this submenu the launchers may be executed directly[8] as from a GNOME, KDE or XFCE panel menu (see figure 14). The advantage of having this *Applications* popup where you need it and when you need it is less stressful to the impatient mind.

The popup has the following fixed categories, following the Gnome layout, although this layout may vary in the future if such modifications are deemed convenient.

- Accessories
- Graphics applications
- System Tools
- Internet and Network
- Games and amusements
- Office Applications
- Tools for software development
- Audio and Video
- Personal preferences

The amount of applications which will be available depends on the applications currently installed. Any new software which is installed and which provides dot desktop file will be recognized immediately.

[8]Categories Core, Qt, KDE, GTK and GNOME are not present in the popup submenu because these categories are too large. Nonetheless, they are always available in the icon navigation mode.

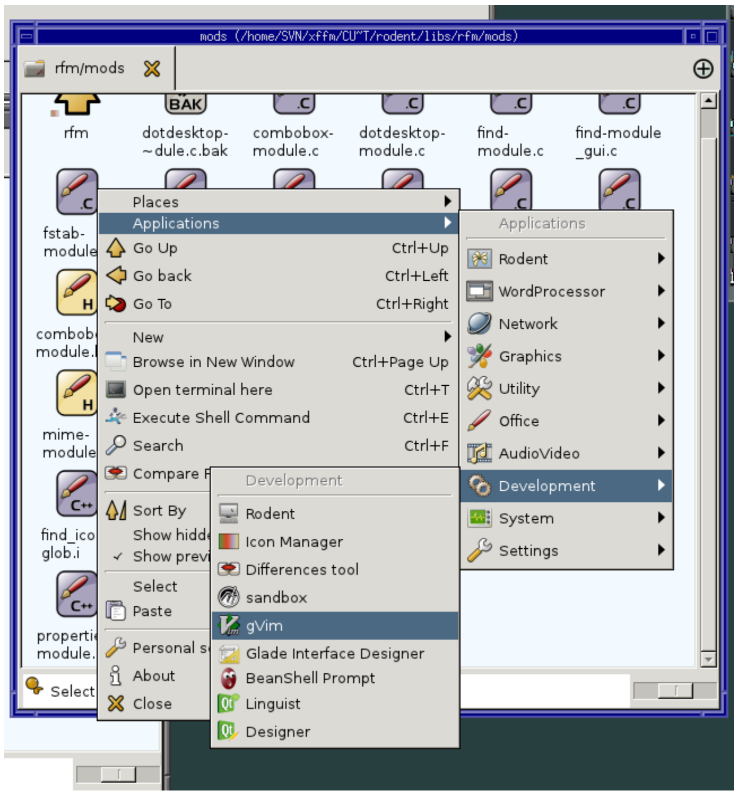
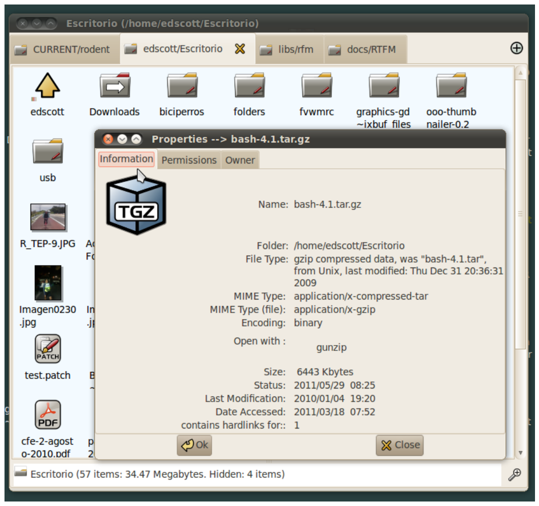


Figure 14: Rodent-dotdesktop application popup submenu



```
                 Name:  bash-4.1.tar.gz

               Folder:  /home/edscott/Escritorio
            File Type:  gzip compressed data, was "bash-4.1.tar",
                        from Unix, last modified: Thu Dec 31 20:36:31
                        2009
            MIME Type:  application/x-compressed-tar
     MIME Type (file):  application/x-gzip
             Encoding:  binary
            Open with :
                           gunzip

                 Size:   6443 Kbytes
               Status:  2011/05/29  08:25
    Last Modification:  2010/01/04  19:20
        Date Accessed:  2011/03/18  07:52
contains hardlinks for::  1
```

Escritorio (57 items: 34.47 Megabytes. Hidden: 4 items)

Figure 15: Properties plugin dialog

4.7 Non root plugins

Non root plugins add extra functionality to Rodent filemanager, and are only loaded on a need to use basis. Furthermore, the operating system does not reserve memory for modules (in contrast to dynamic load libraries) until load time. This helps keep Rodent small. These plugins may be used by other applications. The API and programmer's guide will be available with Rodent Delta.

4.7.1 Properties plugin

This plugin provides the properties dialog which is summoned from the popup menu. It is a front end for chown, chmod and chgrp. If the user is not authorized to apply such commands, then it will try to use sudo, allowing the privileged user to issue such commands without leaving the filemanager environment. See figure 15.

4.7.2 Icons plugin

This plugin provides access to the Rodent icon theme and the xdg spec named icons. If this module is not installed, Rodent will fall back to a very basic GTK subset of icons. Use this if you do not care about looks but want a filemanager that is fast as a hurricane. The icons plugin requires the mime plugin (section) to load.

4.7.3 Mime plugin

This plugin is in charge of resolving mimetypes for files and associating applications. Eliminating this plugin will impede the icons plugin from loading. If the dotdesktop plugin is installed, this plugin will be enriched by the associations between mimetypes and applications harvested from installed applications.

The plugin uses both the freedesktop mimetype definition as well as those defined by the libmagic library (used by the `file` application). Other mimetypes defined by `googlecode` and `/etc/mime.types` is also used. The configuration file which installs the default applications on a system wide basis is located at `rodent/libs/rfm/mime/mime-module.xml` and is installed to `$PREFIX/share/rfm-Gamma/`.

Applications with are associated by the user by means of the "`Open with`" dialog are saved in a simple text format at `$HOME/.config/rfm-Gamma/user-applications`.

4.7.4 Combobox module

This module creates an extended combo box with historic and bash completion. If the module is not installed, dialogs which Rodent presents with input boxes will consist of simple entry boxes without historic or bash completion.

4.7.5 Find module

This module takes care of building and presenting the graphic front end to the `fgr` command. The module is used by Rodent filemanager as well as the standalone front end `rodent-fgr.` See figure 22.

4.7.6 Settings module

This plugin takes care of managing the shared memory and comunicating configuration changes between applications. It also handles the settings dialog window.

5 Popup menus

The popup menu is context sensitive. There is a popup menu over void space and another over an icon space.

5.1 Popup over void space

Figure 16 shows what the popup menu over void space looks like. The increase and decrease menu items are only shown in the rodent-desk version because the iconview has a dedicated scale widget to provide this functionality.

The structure of the menu is as follows:

- Places (Bookmarks) *Submenu* [This is a submenu which allows the user to quickly navigate to predefined places or plugins. It also includes user

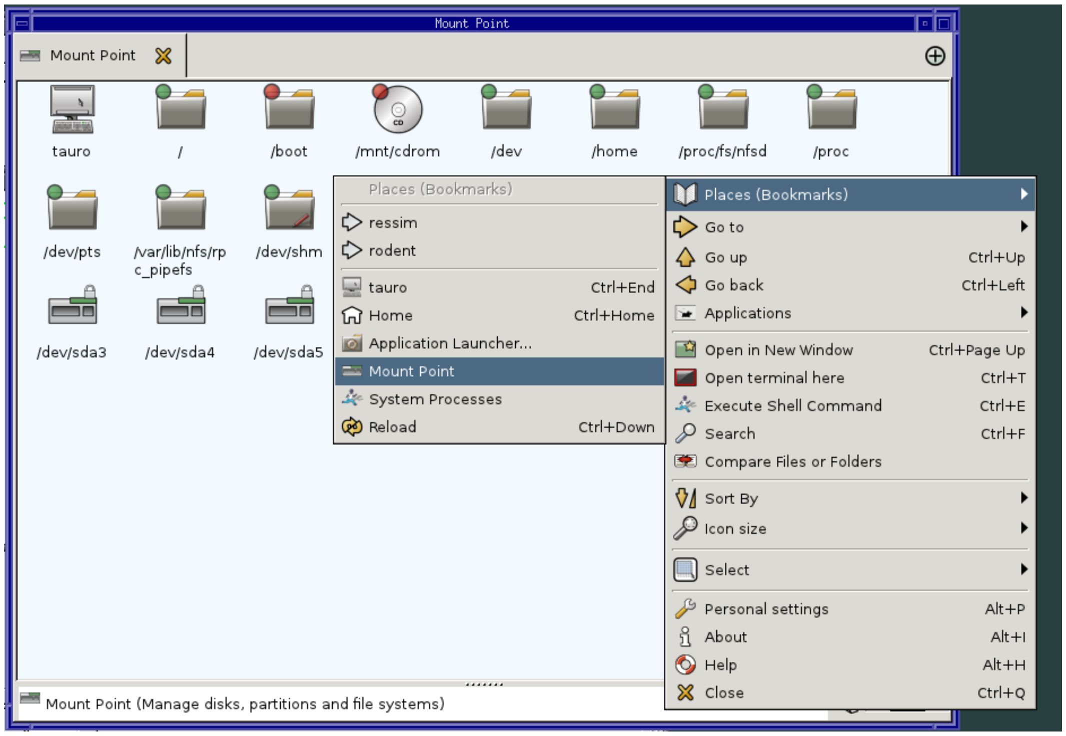


Figure 16: Popup over void space

defined bookmarks. These are the normal bookmarks used by GTK by default.]

- *User bookmark 1* [Go to this bookmark.]
- · · ·
- *User bookmark n*
- Add/remove bookmark (*current directory*) [Add or remove bookmark at current location: only add or remove is shown, depending if location is bookmarked or not.]
- Localhost [This will be the name of the host and will navigate the window to the Rodent root view, where the system root file, user home directory and installed plugins are listed.]
- Home [This will navigate the window to the user's home directory, typically ~/.]

[What follows is a list of menu items to navigate to the installed plugins.]

- Application launcher [This will take the view to the dot desktop, or launchers, plugin.]
- Mount Point [This will take the view to the mount point plugin, showing mount points and partitions and allowing mount/unmount operations.]
- System Processes [This will navigate to the system processes plugin, user or system processes may be viewed and controlled.]
- · · · [Any other root plugins installed.]
- *plugin n*
- Reload [This item will perform a *hard* reload, re-reading directory and recreating thumbnails and previews.]

• Go to *Submenu*

- Go to [This will open a dialog with history driven autocompletion to navigate to a particular point in the filesystem. The combo entry dropdown in the dialog is populated with history values sorted by frequency and bash file completion is enabled on keyboard input. And there is a conventional GTK directory selector button to choose a destination.]
- *Quick chdir 1* [These are the equivalent of the quick change directory lines from CDE's dtfile program and which is implemented as buttons in the Nautilus program: in Rodent the items appear in the popup, as everything else.]
- ⋮

  - *Quick chdir n*

- Go up [This causes the window to navigate to the parent directory. The deepest level shown in the quick navigation items is not affected by this action.]

- Go back [This will go backwards in the navigation history. If there is no history yet, this button will not appear.]

- Applications [This is the application launcher menu, provided by the rodent-dotdesktop plugin. Each category will have launchers for the programs which have self classified themselves into these categories.]

  - Accessories
  - Graphics applications
  - System Tools
  - Internet and Network
  - Games and amusements
  - Office Applications
  - Tools for software development
  - Audio and Video
  - Personal preferences

- New *Submenu*

  - Create a new file in the given directory
  - Create a new empty folder inside this folder

- Browse in New Window [This opens a new Rodent window in the current directory.]

- Open terminal here [This will open a terminal emulator in the currently viewed directory. If the current view is not a directory, then the terminal will open at the user's home directory.]

- Execute Shell Command [Executes a shell command and pipes standard output (in black) and standard error output (in red) to lp terminal area.]

- Search [Opens the rodent-fgr search tool dialog. Output may be directed either to the lp-terminal area or contained within the dialog itself.]

- Compare Files or Folders [This opens the rodent-diff file comparison test. This application can create patch files in a variety of formats and view differences side by side.]

- Sort by [The icon shown in this menu element indicates the current sort order, either ascending or descending. The default order is ascending. When the user sets a sort order or method, that order or method is remembered for future views of the directory until the user changes it. These radio menu items select the sort method. Only one is active at a time and the setting is remembered per path.]
  - Ascending/Descending [Toggles sort order]
  - Type [Sort by file type. This is the default.]
  - By Name [Sort by file name. Sort is case sensitive.]
  - By Size [Sort by file size.]
  - By Date [Sort by file modification date.]
  - By Owner [Sort by owner id]
  - By Group [Sort by group id]
  - By Permissions [Sort by permissions]
- Show hidden files [Toggle whether hidden files should be shown. Default: off. This setting is remembered per path.]
- Show preview [Toggle whether previews of images, pdf files and text files should be shown in the tip popup window. Default: on. This setting is remembered per path.]
- Select [This menu item brings up a submenu with the options:]
  - Select All [Selects all items]
  - Select Items Matching... [Opens a dialog which allows selection based on filter.]
  - Unselect [Unselects all previously selected items.]
- Paste [This item allow you to paste the contents of the pasteboard. If the pasteboard is empty, the item is not shown.]
- Personal settings [This opens the rodent settings dialog. Changes made here will affect all instances of rodent filemanager.]
- About [This opens the dialog that displays information about the program: logo, name, copyright, website and license. Credits are given to the authors, documenters, translators and artists who have contributed to the development of the program.]
- Help [This opens the user guide.]
- Close [Exits the application.]

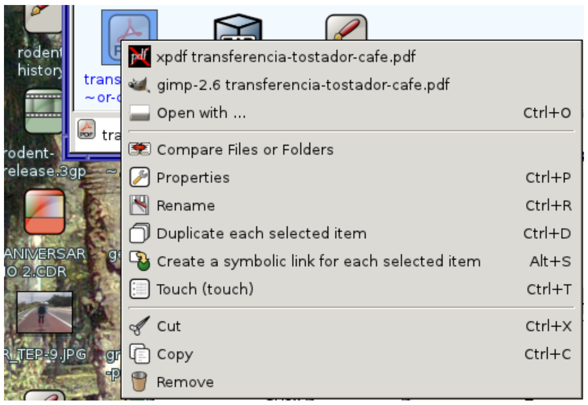

xpdf transferencia-tostador-cafe.pdf
gimp-2.6 transferencia-tostador-cafe.pdf
Open with ... Ctrl+O
Compare Files or Folders
Properties Ctrl+P
Rename Ctrl+R
Duplicate each selected item Ctrl+D
Create a symbolic link for each selected item Alt+S
Touch (touch) Ctrl+T
Cut Ctrl+X
Copy Ctrl+C
Remove

Figure 17: Popup over icon space

5.2 Popup over icon space

Figure 17 shows what the popup menu over void space looks like.

The popup menu is context sensitive. The contents of the menu depend on what is located beneath the pointer when the menu is summoned. If nothing is beneath the pointer, the general popup appears. The layout of the icon specific popup is as follows.

- Execute file [If the file is executable, then this element will appear.]
- Unpack file [If the file is a zipped tarball, then this element will appear.
- Mimetype application 1[According to the mimetype of the file, different applications to open or process the file may appear. The exact contents of the menu items depends on the installed programs and the availability of the dotdesktop and mime plugins.]
- $\vdots$
- Mimetype application n [The amount of mimetype applications is limited to n menu items as a maximum.]
- Open with [Opens a dialog to choose the application to use. Multiple files may be opened with the same application with this option.]
- Compare files or folders [Only appears if selected items are two or less.]
- Properties [This will open the properties dialog where characteristics of the selected file may be viewed or changed. Multiple files may be selected.]
- Mount [If the item is a an x-cd-image, then this item will allow mounting as a loop device. If the item is listed in /etc/fstab, this item will allow mounting the device. If condition does not apply, menu item will not be shown.]

- Unmount [If the item is currently mounted, this item will allow dismounting. If condition does not apply, menu item will not be shown.]
- Rename [This will allow inline renaming of the item.]
- Duplicate [This will allow inline duplicating of the item. Files or entire directories may be duplicated.]
- Symlink [This will allow inline creation of a symbolic link to the item.]
- Touch [This open a front end dialog to the *touch* command.]
- Cut [This will cut the item or items to the paste board. Cut items are tagged with a scissors emblem on the upper left corner. Cut items will be moved to the paste destination.]
- Copy [This will copy the item or items to the paste board. Copied items are tagged with a copy emblem on the upper left corner. Copied items will be copied to the paste destination.]
- Remove [This will open the remove dialog for the item or items. Operation may be cancelled, unlinked or shred. Unlinked items are not recoverable by the ordinary user but may be retrieved by specialized teams. Shreded items may or may not be unlinked, but recovery is almost impossible even for the most capable information services. The schred option is only available is GNU shred is installed in the system.]
- Move to trash [This option is not available in Rodent Gamma. Type "cd ~/.Trash" in lpterminal to access Xdg spec trash folder.]

6 Configuration

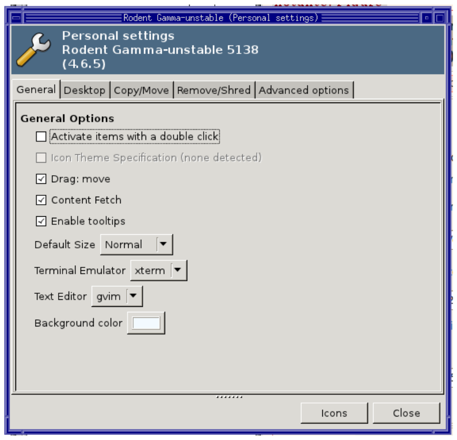
All configurable options are set through environment variables. The settings dialog allows for easy setting of these variables, as well as direct in line text setting and reviewing.

The settings dialog has the following options.

6.1 General

See figure 18.

- Activate items with a double click
  - Icon Theme Specification
  - Drag does move or copy by default (Drag: move)
  - Folders show content emblem (Content Fetch)
  - Popup previews and file information is shown (Enable tooltips)



Rodent Gamma-unstable (Personal settings)

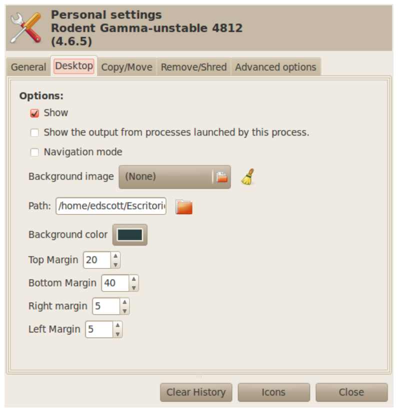
Personal settings
Rodent Gamma-unstable 5138
(4.6.5)

General Options

☐ Activate items with a double click
☐ Icon Theme Specification (none detected)
☑ Drag: move
☑ Content Fetch
☑ Enable tooltips

Default Size: Normal
Terminal Emulator: xterm
Text Editor: gvim
Background color

Figure 18: General settings



Personal settings
Rodent Gamma-unstable 4812
(4.6.5)

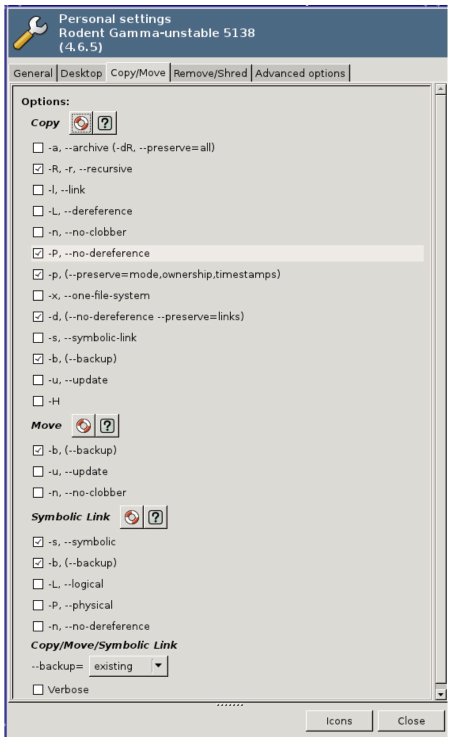
Options:

☑ Show
☐ Show the output from processes launched by this process.
☐ Navigation mode

Background image: (None)
Path: /home/edscott/Escritori
Background color
Top Margin: 20
Bottom Margin: 40
Right margin: 5
Left Margin: 5

Figure 19: Desktop configuration

- Default icon size
  - Normal
  - Details
  - Compact
  - Big
  - Huge
- Default terminal emulator (options depend on the emulators available).
- Default text editor (options depend on the editors available).
- Background color.



Options:

Copy

- ☐ -a, --archive (-dR, --preserve=all)
- ☑ -R, -r, --recursive
- ☐ -l, --link
- ☐ -L, --dereference
- ☐ -n, --no-clobber
- ☑ -P, --no-dereference
- ☑ -p, (--preserve=mode,ownership,timestamps)
- ☐ -x, --one-file-system
- ☑ -d, (--no-dereference --preserve=links)
- ☐ -s, --symbolic-link
- ☑ -b, (--backup)
- ☐ -u, --update
- ☐ -H

Move

- ☑ -b, (--backup)
- ☐ -u, --update
- ☐ -n, --no-clobber

Symbolic Link

- ☑ -s, --symbolic
- ☑ -b, (--backup)
- ☐ -L, --logical
- ☐ -P, --physical
- ☐ -n, --no-dereference

Copy/Move/Symbolic Link

--backup= existing

- ☐ Verbose

Figure 20: GNU cp/mv options

6.2 Desktop

See figure 19.

- Show icons on the root windows (Show Desktop Grid)
  - Show the output from processes launched from the desktop
  - Allow navigation on the desktop
  - Background image selector
  - Path to desktop directory selector
  - Background color selector
  - Top, bottom, right and left margin settings (in pixels).

6.3 Copy/Move options

See figure 20. Some of these options may not be available under BSD.

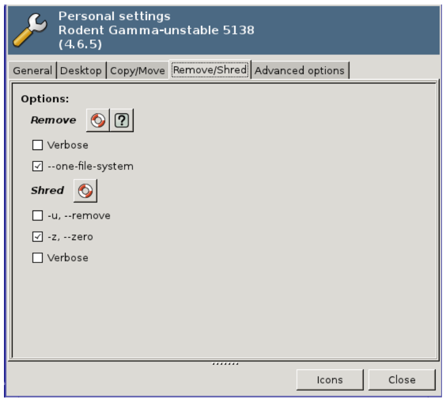
- Copy options

  - -*a* same as -dR –preserve=all
    - -*R* copy directories recursively
    - -*l* link files instead of copying
    - -*L* always follow symbolic links in SOURCE
    - -*n* do not overwrite an existing file
    - -*P* never follow symbolic links in SOURCE
    - -*p* same as –preserve=mode,ownership,timestamps
    - -*x* stay on this file system
    - -*d* no-dereference, preserve links:
    - -*s* make symbolic links instead of copying
    - -*b* make a backup of each existing destination file
    - -*u* copy only when the SOURCE file is newer than the destination file or when the destination file is missing
    - -*H* follow symbolic links in SOURCE
  - Move options
    - -*b* make a backup of each existing destination file
    - -*u* copy only when the SOURCE file is newer than the destination file or when the destination file is missing
    - *n* do not overwrite an existing file
  - Copy/move/link options
    - Backup method selector
      - simple
      - existing
      - numbered
      - none

6.4 Remove/Shred

See figure 21

- Remove
  - -*v* verbose
    - -*x* remain on this file system
  - Shred
    - -*u* truncate and remove file after overwriting
    - -*z* add a final overwrite with zeros to hide shredding
    - -*v* verbose


Personal settings
Rodent Gamma-unstable 5138
(4.6.5)

General | Desktop | Copy/Move | Remove/Shred | Advanced options

Options:

Remove

☐ Verbose
☑ --one-file-system

Shred

☐ -u, --remove
☑ -z, --zero
☐ Verbose

Figure 21: GNU rm/shred options

6.5 Advanced Options

- This item lists and allows editing the configuration enviroment variables.

7 Helper applications

7.1 Internal

Several commands and auxiliary applications or artwork are distributed with the Rodent Filemanager package. These are the internal helper applications, and may be used independently or by other applications.

7.1.1 Rodent-fgr and fgr: File content search command

`rodent-fgr` is a graphic frontend for `fgr`. See figure 22.

SYNOPSYS `fgr [-r] [-v] [-d ddd] [-m mmm] [-f filter] [-s (+/-)size] [-t type] [-p perm] [grep options...]`

-v verbose

-V Print version number information

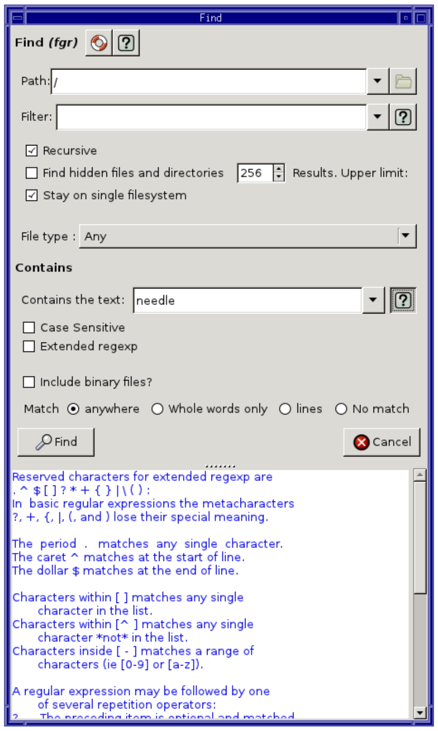
-P print process id

-f filter file filter (enclosed in quotes if regexp `*,?` or `[]` is used)

-r recursive

-s `+kbytes` Size greater than kbytes `KBYTES`

-s `-kbytes` size less than kbytes `KBYTES`



Path: /

Filter:

[✓] Recursive
[ ] Find hidden files and directories
256 Results. Upper limit:
[✓] Stay on single filesystem

File type : Any

Contains

Contains the text: needle

[ ] Case Sensitive
[ ] Extended regexp

[ ] Include binary files?

Match [✓ anywhere] Whole words only lines No match

```
Reserved characters for extended regexp are
. ^ $ [ ] ? * + { } | \ ( ) :
In  basic regular expressions the metacharacters
?, +, {, |, (, and ) lose their special meaning.

The  period  .   matches  any  single  character.
The caret ^ matches at the start of line.
The dollar $ matches at the end of line.

Characters within [ ] matches any single
      character in the list.
Characters within [^ ] matches any single
      character *not* in the list.
Characters inside [ - ] matches a range of
      characters (ie [0-9] or [a-z]).

A regular expression may be followed by one
      of several repetition operators:
?     The preceding item is optional and matched
```

Figure 22: Rodent-fgr

-p perm perm is either `suid | exe`

-t type `reg | dir | sym | sock | blk | chr | fifo` (regular, directory, symlink, socket, blk_dev, chr_dev, fifo)

-d `ddd` created or modified in the previous (int) `ddd` days

-m `mmm` created or modified in the previous (int) mm months

-p `perm` perm is either `suid | exe`

-e `string` containing string (if `*,?` or `[]`, use quotes)

-E `regexp` containing regexp: (use quotes).

-i ignore case (for search string -c)

-y same as -i (obsolete)

7.1.2 Rodent-diff

Graphic front end to the GNU `diff` command. See figure 23.

7.1.3 Rodent-mime

This is the application used to customize the icons used by Rodent. All icons in the Rodent filemanager are customizable. In order to customize, drag and drop the icons from the treeview on the right to the treeview on the left. When you press the save button, a custom will be saved in your configuration directory and the full path will be listed in a dialog after a successful write.

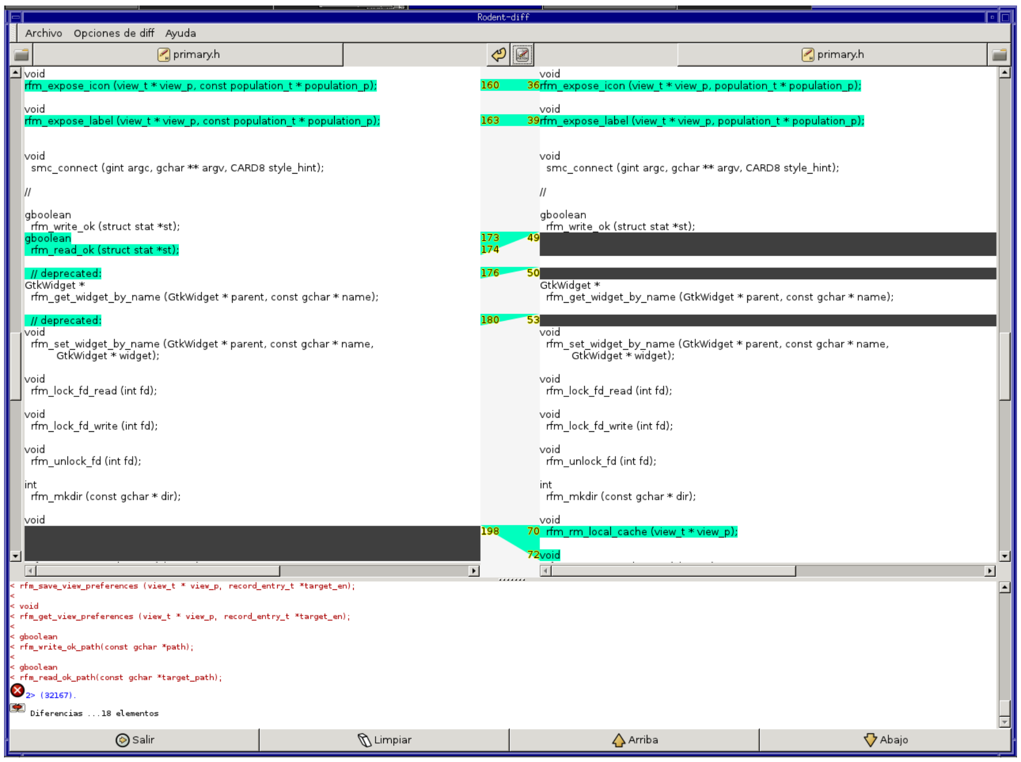
If by any chance you want to make your custom icon layout the system wide default, move it to `$PREFIX/rfm-Gamma/icons.mime.xml`. The default configuration file in the tarball distribution is located at rodent/libs/icons/mime.

If you change icon themes, since the Rodent icon theme follows the xdg naming specification, the respective icons should be found in the new icon theme. If the new icon theme does not have an icon for a standard named icon, then Rodent falls back to the original Rodent icon theme.

The rodent-mime application is shown in figure 24. This application may be executed either from the lp-terminal command line or by pressing the "icons" button in the "Personal preferences" dialog.

7.1.4 Rodent-root

Rodent-root is the application which sets the background image for the desktop. It will set the default X root window to use the image, as well as the background for the rodent-desk application. When the background is changed by means of the "Personal preferences" dialog, this is the application which is called. It may also be called by selecting the popup menu over any supported image type. Once rodent-root selects a background image, this choice becomes the user's default background.

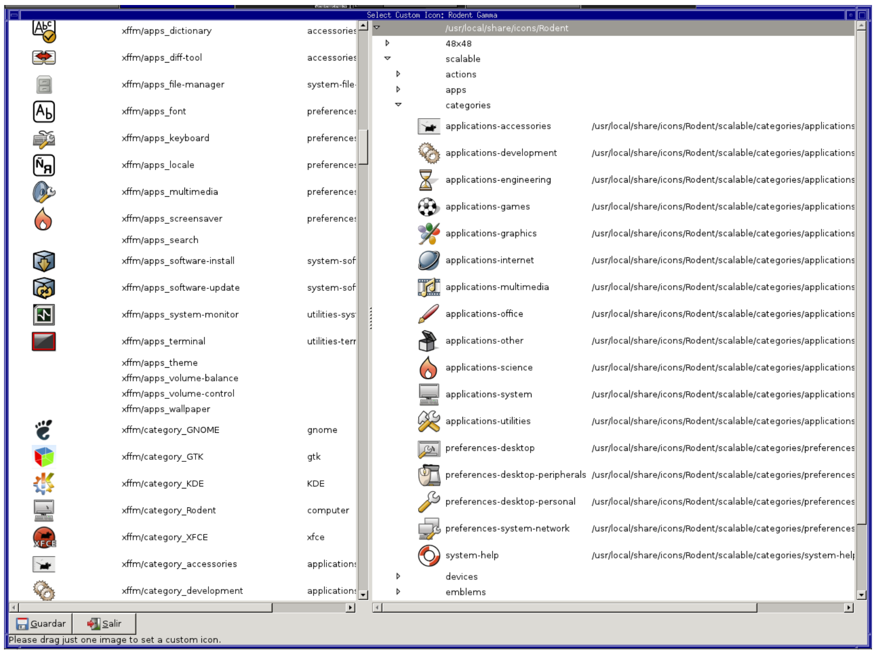


```
void
rfm_expose_icon (view_t * view_p, const population_t * population_p);

void
rfm_expose_label (view_t * view_p, const population_t * population_p);


void
  smc_connect (gint argc, gchar ** argv, CARD8 style_hint);

//

gboolean
  rfm_write_ok (struct stat *st);
gboolean
  rfm_read_ok (struct stat *st);

  // deprecated:
GtkWidget *
  rfm_get_widget_by_name (GtkWidget * parent, const gchar * name);

  // deprecated:
void
  rfm_set_widget_by_name (GtkWidget * parent, const gchar * name,
          GtkWidget * widget);

void
  rfm_lock_fd_read (int fd);

void
  rfm_lock_fd_write (int fd);

void
  rfm_unlock_fd (int fd);

int
  rfm_mkdir (const gchar * dir);

void
```

```
160   36 void
           rfm_expose_icon (view_t * view_p, population_t * population_p);

         void
163   39 rfm_expose_label (view_t * view_p, population_t * population_p);


         void
           smc_connect (gint argc, gchar ** argv, CARD8 style_hint);

         //

         gboolean
           rfm_write_ok (struct stat *st);
173   49
174

176   50
         GtkWidget *
           rfm_get_widget_by_name (GtkWidget * parent, const gchar * name);

180   53
         void
           rfm_set_widget_by_name (GtkWidget * parent, const gchar * name,
                   GtkWidget * widget);

         void
           rfm_lock_fd_read (int fd);

         void
           rfm_lock_fd_write (int fd);

         void
           rfm_unlock_fd (int fd);

         int
           rfm_mkdir (const gchar * dir);

         void
198   70   rfm_rm_local_cache (view_t * view_p);

      72 void
```

```
< rfm_save_view_preferences (view_t * view_p, record_entry_t *target_en);
<
< void
< rfm_get_view_preferences (view_t * view_p, record_entry_t *target_en);
<
< gboolean
< rfm_write_ok_path(const gchar *path);
<
< gboolean
< rfm_read_ok_path(const gchar *target_path);
2> (32167).
 Diferencias ...18 elementos
```

Figure 23: rodent-diff front end



```
xffm/apps_dictionary              accessories
xffm/apps_diff-tool               accessories
xffm/apps_file-manager            system-file-
xffm/apps_font                    preferences
xffm/apps_keyboard                preferences
xffm/apps_locale                  preferences
xffm/apps_multimedia              preferences
xffm/apps_screensaver             preferences
xffm/apps_search
xffm/apps_software-install        system-sof
xffm/apps_software-update         system-sof
xffm/apps_system-monitor          utilities-sys
xffm/apps_terminal                utilities-terr
xffm/apps_theme
xffm/apps_volume-balance
xffm/apps_volume-control
xffm/apps_wallpaper
xffm/category_GNOME               gnome
xffm/category_GTK                 gtk
xffm/category_KDE                 KDE
xffm/category_Rodent              computer
xffm/category_XFCE                xfce
xffm/category_accessories         applications
xffm/category_development         applications
```

```
/usr/local/share/icons/Rodent
  48x48
  scalable
    actions
    apps
    categories
      applications-accessories              /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-development              /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-engineering              /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-games                    /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-graphics                 /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-internet                 /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-multimedia               /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-office                   /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-other                    /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-science                  /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-system                   /usr/local/share/icons/Rodent/scalable/categories/applications
      applications-utilities                /usr/local/share/icons/Rodent/scalable/categories/applications
      preferences-desktop                   /usr/local/share/icons/Rodent/scalable/categories/preferences
      preferences-desktop-peripherals       /usr/local/share/icons/Rodent/scalable/categories/preferences
      preferences-desktop-personal          /usr/local/share/icons/Rodent/scalable/categories/preferences
      preferences-system-network            /usr/local/share/icons/Rodent/scalable/categories/preferences
      system-help                           /usr/local/share/icons/Rodent/scalable/categories/system-hel
    devices
    emblems
```

Please drag just one image to set a custom icon.

Figure 24: Rodent-mime (custom icon layout setup)

7.1.5 Rodent icon theme

The Rodent icon theme is a native Xfce development. It was created by Francois Le Clainche and has been enriched by Lonerocker's contributions at freedesktop.org. The icon theme is now distributed and maintained by the Rodent filemanager project. The name of the Rodent filemanager is derived from this icon theme, although you can choose any desktop icon theme you may fancy.

7.2 External

Reinvent the wheel? No thanks. Bug free code exists for copying, moving, linking, touching and many other fundamental file operations in GNU and BSD versions. Reusability is one of the strengths of free software which should not be ignored.

In the initial years, one of the defining points of Xfce in the early years was to keep applications small. So if time proven versions for the same operations is already available and installed, why duplicate code?

What follows is a description of external programs developed by world class programmers which comprise the core set of operations used by the Rodent filemanager. Currently the full power of these commands is not harnessed by Rodent, but functionality will be extended in future releases.

7.2.1 cp: Copy files and directories [9]

Written by Torbjorn Granlund, David MacKenzie, and Jim Meyering.

'`cp`' copies files (or, optionally, directories). The copy is completely independent of the original. You can either copy one file to another, or copy arbitrarily many files to a destination directory.

Synopses: `cp [OPTION]... [-T] SOURCE DEST cp [OPTION]... SOURCE... DIRECTORY cp [OPTION]... -t DIRECTORY SOURCE...`

- If two file names are given, '`cp`' copies the first file to the second.
- If the '`--target-directory`' ('`-t`') option is given, or failing that if the last file is a directory and the '`--no-target-directory`' ('`-T`') option is not given, '`cp`' copies each `SOURCE` file to the specified directory, using the `SOURCEs`' names.

Generally, files are written just as they are read. For exceptions, see the '`--sparse`' option below.

By default, '`cp`' does not copy directories. However, the '`-R`', '`-a`', and '`-r`' options cause '`cp`' to copy recursively by descending into source directories and copying files to corresponding destination directories.

When copying from a symbolic link, '`cp`' normally follows the link only when not copying recursively. This default can be overridden with the '`--archive`' ('`-a`'), '`-d`', '`--dereference`' ('`-L`'), '`--no-dereference`' ('`-P`'), and '`-H`' options. If more than one of these options is specified, the last one silently overrides the others.

When copying to a symbolic link, '`cp`' follows the link only when it refers to an existing regular file. However, when copying to a dangling symbolic link, '`cp`' refuses by default, and fails with a diagnostic, since the operation is inherently dangerous. This behavior is contrary to historical practice and to `POSIX`. Set '`POSIXLY_CORRECT`' to make '`cp`' attempt to create the target of a dangling destination symlink, in spite of the possible risk. Also, when an option like '`--backup`' or '`--link`' acts to rename or remove the destination before copying, '`cp`' renames or removes the symbolic link rather than the file it points to.

By default, '`cp`' copies the contents of special files only when not copying recursively. This default can be overridden with the '`--copy-contents`' option.

'`cp`' generally refuses to copy a file onto itself, with the following exception: if '`--force --backup`' is specified with `SOURCE` and `DEST` identical, and referring to a regular file, '`cp`' will make a backup file, either regular or numbered, as specified in the usual ways (*note Backup options::). This is useful when you simply want to make a backup of an existing file before changing it.

The program accepts the following options. Also see *note Common options::.

[9] The following section is available with the command '`info cp`'.

'-a' '–archive' Preserve as much as possible of the structure and attributes of the original files in the copy (but do not attempt to preserve internal directory structure; i.e., `'ls -U'` may list the entries in a copied directory in a different order). Try to preserve SELinux security context and extended attributes (`xattr`), but ignore any failure to do that and print no corresponding diagnostic. Equivalent to `'-dR --preserve=all'` with the reduced diagnostics.

'-b' '–backup[=METHOD]' Make a backup of each file that would otherwise be overwritten or removed. As a special case, `'cp'` makes a backup of `SOURCE` when the force and backup options are given and `SOURCE` and `DEST` are the same name for an existing, regular file. More information in section 7.3.1. One useful application of this combination of options is this tiny Bourne shell script:

```
#!/bin/sh
# Usage: backup FILE...
# Create a GNU-style backup of each listed FILE.
for i;
do
  cp --backup --force -- "$i" "$i"
done
```

'–copy-contents' If copying recursively, copy the contents of any special files (e.g., FIFOs and device files) as if they were regular files. This means trying to read the data in each source file and writing it to the destination. It is usually a mistake to use this option, as it normally has undesirable effects on special files like FIFOs and the ones typically found in the `'/dev'` directory. In most cases, `'cp -R --copy-contents'` will hang indefinitely trying to read from FIFOs and special files like `'/dev/console'`, and it will fill up your destination disk if you use it to copy `'/dev/zero'`. This option has no effect unless copying recursively, and it does not affect the copying of symbolic links.

'-d' Copy symbolic links as symbolic links rather than copying the files that they point to, and preserve hard links between source files in the copies. Equivalent to `'--no-dereference --preserve=links'`.

'-f' '–force' When copying without this option and an existing destination file cannot be opened for writing, the copy fails. However, with `'--force'`), when a destination file cannot be opened, `'cp'` then removes it and tries to open it again. Contrast this behavior with that enabled by `'--link'` and `'--symbolic-link'`, whereby the destination file is never opened but rather is removed unconditionally. Also see the description of `'--remove-destination'`.

This option is independent of the `'--interactive'` or `'-i'` option: neither cancels the effect of the other.

This option is redundant if the '--no-clobber' or '-n' option is used.

'-H' If a command line argument specifies a symbolic link, then copy the file it points to rather than the symbolic link itself. However, copy (preserving its nature) any symbolic link that is encountered via recursive traversal.

'-i' '–interactive' When copying a file other than a directory, prompt whether to overwrite an existing destination file. The '-i' option overrides a previous '-n' option.

'-l' '–link' Make hard links instead of copies of non-directories.

'-L' '–dereference' Follow symbolic links when copying from them. With this option, 'cp' cannot create a symbolic link. For example, a symlink (to regular file) in the source tree will be copied to a regular file in the destination tree.

'-n' '–no-clobber' Do not overwrite an existing file. The '-n' option overrides a previous '-i' option. This option is mutually exclusive with '-b' or '--backup' option.

'-P' '–no-dereference' Copy symbolic links as symbolic links rather than copying the files that they point to. This option affects only symbolic links in the source; symbolic links in the destination are always followed if possible.

'-p' '–preserve[=ATTRIBUTE_LIST]' Preserve the specified attributes of the original files. If specified, the `ATTRIBUTE_LIST` must be a comma-separated list of one or more of the following strings:

'mode' Preserve the file mode bits and access control lists.

'ownership' Preserve the owner and group. On most modern systems, only users with appropriate privileges may change the owner of a file, and ordinary users may preserve the group ownership of a file only if they happen to be a member of the desired group.

'timestamps' Preserve the times of last access and last modification, when possible. On older systems, it is not possible to preserve these attributes when the affected file is a symbolic link. However, many systems now provide the 'utimensat' function, which makes it possible even for symbolic links.

'links' Preserve in the destination files any links between corresponding source files. Note that with '-L' or '-H', this option can convert symbolic links to hard links. For example,

```
$ mkdir c; : > a; ln -s a b; cp -aH a b c; ls -i1 c
74161745 a
74161745 b
```

Note the inputs: '`b`' is a symlink to regular file '`a`', yet the files in destination directory, '`c/`', are hard-linked. Since '`-a`' implies '`--preserve=links`', and since '`-H`' tells '`cp`' to dereference command line arguments, it sees two files with the same inode number, and preserves the perceived hard link.

Here is a similar example that exercises '`cp`''s '`-L`' option:

```
$ mkdir b c; (cd b; : > a; ln -s a b)
$ cp -aL b c; ls -i1 c/b
74163295 a
74163295 b
```

'context' Preserve SELinux security context of the file, or fail with full diagnostics.

'xattr' Preserve extended attributes of the file, or fail with full diagnostics. If '`cp`' is built without xattr support, ignore this option. If SELinux context, ACLs or Capabilities are implemented using xattrs, they are preserved by this option as well.

'all' Preserve all file attributes. Equivalent to specifying all of the above, but with the difference that failure to preserve SELinux security context or extended attributes does not change 'cp''s exit status. In contrast to '-a', all but 'Operation not supported' warnings are output.

Using '`--preserve`' with no `ATTRIBUTE_LIST` is equivalent to '`--preserve=mode,ownership,timestamps`'.

In the absence of this option, each destination file is created with the mode bits of the corresponding source file, minus the bits set in the umask and minus the set-user-ID and set-group-ID bits (see section 7.3.2).

'–no-preserve=ATTRIBUTE_LIST' Do not preserve the specified attributes. The `ATTRIBUTE_LIST` has the same form as for '`--preserve`'.

'–parents' Form the name of each destination file by appending to the target directory a slash and the specified name of the source file. The last argument given to '`cp`' must be the name of an existing directory. For example, the command:

```
cp --parents a/b/c existing_dir
```

copies the file '`a/b/c`' to '`existing_dir/a/b/c`', creating any missing intermediate directories.

'-R' '-r' '–recursive' Copy directories recursively. By default, do not follow symbolic links in the source; see the '`--archive`' ('`-a`'), '`-d`', '`--dereference`' ('`-L`'), '`--no-dereference`' ('`-P`'), and '`-H`' options. Special files are copied by creating a destination file of the same type as the source; see the '`--copy-contents`' option. It is not portable to use '`-r`' to copy symbolic links or special files. On some non-GNU systems, '`-r`' implies the equivalent of '`-L`' and '`--copy-contents`' for historical reasons. Also, it is not portable to use '`-R`' to copy symbolic links unless you also specify '`-P`', as `POSIX` allows implementations that dereference symbolic links by default.

'–reflink[=WHEN]' Perform a lightweight, copy-on-write (COW) copy. Copying with this option can succeed only on some file systems. Once it has succeeded, beware that the source and destination files share the same disk data blocks as long as they remain unmodified. Thus, if a disk I/O error affects data blocks of one of the files, the other suffers the exact same fate.

The `WHEN` value can be one of the following:

'always' The default behavior: if the copy-on-write operation is not supported then report the failure for each file and exit with a failure status.

'auto' If the copy-on-write operation is not supported then fall back to the standard copy behaviour.

'–remove-destination' Remove each existing destination file before attempting to open it (contrast with '`-f`' above).

'–sparse=WHEN' A "sparse file" contains "holes" —a sequence of zero bytes that does not occupy any physical disk blocks; the '`read`' system call reads these as zeros. This can both save considerable disk space and increase speed, since many binary files contain lots of consecutive zero bytes. By default, '`cp`' detects holes in input source files via a crude heuristic and makes the corresponding output file sparse as well. Only regular files may be sparse.

The `WHEN` value can be one of the following:

'auto' The default behavior: if the input file is sparse, attempt to make the output file sparse, too. However, if an output file exists but refers to a non-regular file, then do not attempt to make it sparse.

'always' For each sufficiently long sequence of zero bytes in the input file, attempt to create a corresponding hole in the output file, even if the input file does not appear to be sparse. This is useful when the input file resides on a file system that does not support sparse files (for example, 'efs' file systems in SGI IRIX 5.3 and earlier), but the output file is on a type of file system that does support them. Holes may be created only in regular files, so if the destination file is of some other type, '`cp`' does not even try to make it sparse.

'never' Never make the output file sparse. This is useful in creating a file for use with the '`mkswap`' command, since such a file must not have any holes.

'–strip-trailing-slashes' Remove any trailing slashes from each `SOURCE` argument (see section 7.3.10).

'-s' '–symbolic-link' Make symbolic links instead of copies of non-directories. All source file names must be absolute (starting with '`/`') unless the destination files are in the current directory. This option merely results in an error message on systems that do not support symbolic links.

'-S SUFFIX' '–suffix=SUFFIX' Append `SUFFIX` to each backup file made with '`-b`'(see section 7.3.1).

'-t DIRECTORY' '–target-directory=DIRECTORY' Specify the destination DIRECTORY (see section 7.3.9).

'-T' '–no-target-directory' Do not treat the last operand specially when it is a directory or a symbolic link to a directory(see section 7.3.9).

'-u' '–update' Do not copy a non-directory that has an existing destination with the same or newer modification time. If time stamps are being preserved, the comparison is to the source time stamp truncated to the resolutions of the destination file system and of the system calls used to update time stamps; this avoids duplicate work if several '`cp -pu`' commands are executed with the same source and destination.

'-v' '–verbose' Print the name of each file before copying it.

'-x' '–one-file-system' Skip subdirectories that are on different file systems from the one that the copy started on. However, mount point directories *are* copied.

An exit status of zero indicates success, and a nonzero value indicates failure.

7.2.2 rm: Remove files or directories

Written by Paul Rubin, David MacKenzie, Richard M. Stallman, and Jim Meyering.

'`rm`' removes each given `FILE`. By default, it does not remove directories.

Synopsis:

```
rm [OPTION]... [FILE]...
```

If the '`-I`' or '`--interactive=once`' option is given, and there are more than three files or the '`-r`', '`-R`', or '`--recursive`' are given, then '`rm`' prompts the user for whether to proceed with the entire operation. If the response is not affirmative, the entire command is aborted.

Otherwise, if a file is unwritable, standard input is a terminal, and the '`-f`' or '`--force`' option is not given, or the '`-i`' or '`--interactive=always`' option *is* given, '`rm`' prompts the user for whether to remove the file. If the response is not affirmative, the file is skipped.

Any attempt to remove a file whose last file name component is '`.`' or '`..`' is rejected without any prompting.

Warning: If you use '`rm`' to remove a file, it is usually possible to recover the contents of that file. If you want more assurance that the contents are truly unrecoverable, consider using '`shred`' (section 7.2.10).

The program accepts the following options. Also see *note Common options::.

'-f' '–force' Ignore nonexistent files and never prompt the user. Ignore any previous '`--interactive`' ('`-i`') option.

'-i' Prompt whether to remove each file. If the response is not affirmative, the file is skipped. Ignore any previous '`--force`' ('`-f`') option. Equivalent to '`--interactive=always`'.

'-I' Prompt once whether to proceed with the command, if more than three files are named or if a recursive removal is requested. Ignore any previous '`--force`' ('`-f`') option. Equivalent to '`--interactive=once`'.

'–interactive [=WHEN]' Specify when to issue an interactive prompt. `WHEN` may be omitted, or one of:

never - Do not prompt at all.

once - Prompt once if more than three files are named or if a recursive removal is requested. Equivalent to '`-I`'.

always - Prompt for every file being removed. Equivalent to '`-i`'. '`--interactive`' with no `WHEN` is equivalent to '`--interactive=always`'.

'–one-file-system' When removing a hierarchy recursively, skip any directory that is on a file system different from that of the corresponding command line argument.

This option is useful when removing a build "`chroot`" hierarchy, which normally contains no valuable data. However, it is not uncommon to bind-mount '`/home`' into such a hierarchy, to make it easier to use one's start-up file. The catch is that it's easy to forget to unmount '`/home`'. Then, when you use '`rm -rf`' to remove your normally throw-away `chroot`, that command will remove everything under '`/home`', too. Use the '`--one-file-system`' option, and it will warn about and skip directories on other file systems. Of course, this will not save your '`/home`' if it and your `chroot` happen to be on the same file system.

'–preserve-root' Fail upon any attempt to remove the root directory, '`/`', when used with the '`--recursive`' option. This is the default behavior (see section 7.3.11).

'–no-preserve-root' Do not treat '`/`' specially when removing recursively. This option is not recommended unless you really want to remove all the files on your computer (see section 7.3.11).

'-r' '-R' '–recursive' Remove the listed directories and their contents recursively.

'-v' '–verbose' Print the name of each file before removing it.

One common question is how to remove files whose names begin with a '`-`'. GNU '`rm`', like every program that uses the '`getopt`' function to parse its arguments, lets you use the '`--`' option to indicate that all following arguments are non-options. To remove a file called '`-f`' in the current directory, you could type either:

```
rm -- -f
```

or:

```
rm ./-f
```

The Unix '`rm`' program's use of a single '`-`' for this purpose predates the development of the `getopt` standard syntax.

An exit status of zero indicates success, and a nonzero value indicates failure.

7.2.3 ln: Make links between files

Written by Mike Parker and David MacKenzie.

'`ln`' makes links between files. By default, it makes hard links; with the '`-s`' option, it makes symbolic (or *soft*) links.

Synopses:

```
ln [OPTION]...  [-T] TARGET LINKNAME ln [OPTION]...  TARGET
ln [OPTION]...  TARGET...  DIRECTORY ln [OPTION]...  -t DIRECTORY
TARGET...
```

- If two file names are given, '`ln`' creates a link to the first file from the second.
- If one TARGET is given, '`ln`' creates a link to that file in the current directory.
- If the '`--target-directory`' ('`-t`') option is given, or failing that if the last file is a directory and the '`--no-target-directory`' ('`-T`') option is not given, '`ln`' creates a link to each `TARGET` file in the specified directory, using the `TARGETs'` names.

Normally '`ln`' does not remove existing files. Use the '`--force`' ('`-f`') option to remove them unconditionally, the '`--interactive`' ('`-i`') option to remove them conditionally, and the '`--backup`' ('`-b`') option to rename them.

A *hard link* is another name for an existing file; the link and the original are indistinguishable. Technically speaking, they share the same inode, and the inode contains all the information about a file —indeed, it is not incorrect to say that the inode *is* the file. Most systems prohibit making a hard link to a directory; on those where it is allowed, only the super-user can do so (and with caution, since creating a cycle will cause problems to many other utilities). Hard links cannot cross file system boundaries. (These restrictions are not mandated by POSIX, however.)

Symbolic links (*symlinks* for short), on the other hand, are a special file type (which not all kernels support: System V release 3 (and older) systems lack symlinks) in which the link file actually refers to a different file, by name. When most operations (opening, reading, writing, and so on) are passed the symbolic link file, the kernel automatically *dereferences* the link and operates on the target of the link. But some operations (e.g., removing) work on the link file itself, rather than on its target. The owner and group of a symlink are not significant to file access performed through the link, but do have implications on deleting a symbolic link from a directory with the restricted deletion bit set. On the GNU system, the mode of a symlink has no significance and cannot be changed, but on some BSD systems, the mode can be changed and will affect whether the symlink will be traversed in file name resolution. More on symbolic links in section 7.3.7.

Symbolic links can contain arbitrary strings; a *dangling symlink* occurs when the string in the symlink does not resolve to a file. There are no restrictions against creating dangling symbolic links. There are trade-offs to using absolute or relative symlinks. An absolute symlink always points to the same file, even if the directory containing the link is moved. However, if the symlink is visible from more than one machine (such as on a networked file system), the file pointed to might not always be the same. A relative symbolic link is resolved in relation to the directory that contains the link, and is often useful in referring to files on the same device without regards to what name that device is mounted on when accessed via networked machines.

When creating a relative symlink in a different location than the current directory, the resolution of the symlink will be different than the resolution of the same string from the current directory. Therefore, many users prefer to first change directories to the location where the relative symlink will be created, so that tab-completion or other file resolution will find the same target as what will be placed in the symlink.

The program accepts the following options. Also see section 7.3.8.

'-b' '–backup[=METHOD]' Make a backup of each file that would otherwise be overwritten or removed (see section 7.3.1).

'-d' '-F' '–directory' Allow users with appropriate privileges to attempt to make hard links to directories. However, note that this will probably fail due to system restrictions, even for the super-user.

'-f' '–force' Remove existing destination files.

'-i' '–interactive' Prompt whether to remove existing destination files.

'-L' '–logical' If '`-s`' is not in effect, and the source file is a symbolic link, create the hard link to the file referred to by the symbolic link, rather than the symbolic link itself.

'-n' '–no-dereference' Do not treat the last operand specially when it is a symbolic link to a directory. Instead, treat it as if it were a normal file.

When the destination is an actual directory (not a symlink to one), there is no ambiguity. The link is created in that directory. But when the specified destination is a symlink to a directory, there are two ways to treat the user's request. `'ln'` can treat the destination just as it would a normal directory and create the link in it. On the other hand, the destination can be viewed as a non-directory —as the symlink itself. In that case, `'ln'` must delete or backup that symlink before creating the new link. The default is to treat a destination that is a symlink to a directory just like a directory.

This option is weaker than the `'--no-target-directory'` (`'-T'`) option, so it has no effect if both options are given.

'-P' '–physical' If `'-s'` is not in effect, and the source file is a symbolic link, create the hard link to the symbolic link itself. On platforms where this is not supported by the kernel, this option creates a symbolic link with identical contents; since symbolic link contents cannot be edited, any file name resolution performed through either link will be the same as if a hard link had been created.

'-s' '–symbolic' Make symbolic links instead of hard links. This option merely produces an error message on systems that do not support symbolic links.

'-S SUFFIX' '–suffix=SUFFIX' Append `SUFFIX` to each backup file made with `'-b'` (see section 7.3.1).

'-t DIRECTORY' '–target-directory=DIRECTORY' Specify the destination `DIRECTORY` (see section 7.3.9).

'-T' '–no-target-directory' Do not treat the last operand specially when it is a directory or a symbolic link to a directory (see section 7.3.9).

'-v' '–verbose' Print the name of each file after linking it successfully.

If `'-L'` and `'-P'` are both given, the last one takes precedence. If `'-s'` is also given, `'-L'` and `'-P'` are silently ignored. If neither option is given, then this implementation defaults to `'-P'` if the system `'link'` supports hard links to symbolic links (such as the GNU system), and `'-L'` if `'link'` follows symbolic links (such as on BSD).

An exit status of zero indicates success, and a nonzero value indicates failure.

- Examples:

  Bad Example:

  ```
  # Create link ../a pointing to a in that directory.

  # Not really useful because it points to itself.

  ln -s a ..
  ```

Better Example:

```
# Change to the target before creating symlinks to avoid being
confused.  cd ..  ln -s adir/a .
```

Bad Example:

```
# Hard coded file names don't move well.

ln -s $(pwd)/a /some/dir/
```

Better Example:

```
# Relative file names survive directory moves and also

# work across networked file systems.

ln -s afile anotherfile

ln -s ../adir/afile yetanotherfile
```

7.2.4 touch: Change file timestamps

Written by Paul Rubin, Arnold Robbins, Jim Kingdon, David MacKenzie, and Randy Smith.

'touch' changes the access and/or modification times of the specified files. Synopsis:

```
touch [OPTION]... FILE...
```

Any FILE argument that does not exist is created empty, unless option '--no-create' ('-c') or '--no-dereference' ('-h') was in effect.

A FILE argument string of '-' is handled specially and causes 'touch' to change the times of the file associated with standard output.

If changing both the access and modification times to the current time, 'touch' can change the timestamps for files that the user running it does not own but has write permission for. Otherwise, the user must own the files.

Although 'touch' provides options for changing two of the times —the times of last access and modification— of a file, there is actually a standard third one as well: the inode change time. This is often referred to as a file's 'ctime'. The inode change time represents the time when the file's meta-information last changed. One common example of this is when the permissions of a file change. Changing the permissions doesn't access the file, so the atime doesn't change, nor does it modify the file, so the mtime doesn't change. Yet, something about the file itself has changed, and this must be noted somewhere. This is the job of the ctime field. This is necessary, so that, for example, a backup program can make a fresh copy of the file, including the new permissions value. Another operation that modifies a file's ctime without affecting the others is renaming. In any case, it is not possible, in normal operations, for a user to change the ctime field to a user-specified value. Some operating systems and file systems support a fourth time: the birth time, when the file was first created; by definition, this timestamp never changes.

Time stamps assume the time zone rules specified by the 'TZ' environment variable, or by the system default rules if 'TZ' is not set (see section 7.3.12).

You can avoid ambiguities during daylight saving transitions by using UTC time stamps.

The program accepts the following options. Also see section_7.3.8.

'-a' '–time=atime' '–time=access' '–time=use' Change the access time only.

'-c' '–no-create' Do not warn about or create files that do not exist.

'-d' '–date=TIME' Use `TIME` instead of the current time. It can contain month names, time zones, `'am'` and `'pm'`, `'yesterday'`, etc. For example, `'--date="2004-02-27 14:19:13.489392193 +0530"'` specifies the instant of time that is 489,392,193 nanoseconds after February 27, 2004 at 2:19:13 PM in a time zone that is 5 hours and 30 minutes east of UTC. File systems that do not support high-resolution time stamps silently ignore any excess precision here.

'-f' Ignored; for compatibility with BSD versions of 'touch'.

'-h' '–no-dereference' Attempt to change the timestamps of a symbolic link, rather than what the link refers to. When using this option, empty files are not created, but option `'-c'` must also be used to avoid warning about files that do not exist. Not all systems support changing the timestamps of symlinks, since underlying system support for this action was not required until `POSIX 2008`. Also, on some systems, the mere act of examining a symbolic link changes the access time, such that only changes to the modification time will persist long enough to be observable. When coupled with option `'-r'`, a reference timestamp is taken from a symbolic link rather than the file it refers to.

'-m' '–time=mtime' '–time=modify' Change the modification time only.

'-r FILE' '–reference=FILE' Use the times of the reference `FILE` instead of the current time. If this option is combined with the `'--date=TIME' ('-d TIME')` option, the reference `FILE`'s time is the origin for any relative `TIME`s given, but is otherwise ignored. For example, `'-r foo -d '-5 seconds"` specifies a time stamp equal to five seconds before the corresponding time stamp for `'foo'`. If `FILE` is a symbolic link, the reference timestamp is taken from the target of the symlink, unless `'-h'` was also in effect.

'-t [[CC]YY]MMDDHHMM[.SS]' Use the argument (optional four-digit or two-digit years, months, days, hours, minutes, optional seconds) instead of the current time. If the year is specified with only two digits, then `CC` is 20 for years in the range 0 ... 68, and 19 for years in 69 ... 99. If no digits of the year are specified, the argument is interpreted as a date in the current year. Note that `SS` may be `'60'`, to accommodate leap seconds.

On older systems, `'touch'` supports an obsolete syntax, as follows. If no timestamp is given with any of the `'-d', '-r'`, or `'-t'` options, and if there are two or more `FILE`s and the first `FILE` is of the form '`MMDDHHMM[YY]`' and this would be a valid argument to the '`-t`' option (if the `YY`, if any, were moved to the front), and if the represented year is in the range 1969-1999, that argument is interpreted as the time for the other files instead of as a file name. This obsolete behavior can be enabled or disabled with the '`_POSIX2_VERSION`' environment variable, but portable scripts should avoid commands whose behavior depends on this variable. For example, use '`touch ./12312359 main.c`' or '`touch -t 12312359 main.c`' rather than the ambiguous '`touch 12312359 main.c`'.

An exit status of zero indicates success, and a nonzero value indicates failure.

7.2.5 sudo: execute a command as another user

Many people have worked on sudo over the years; this version consists of code written primarily by Todd C. Miller. See the HISTORY file in the sudo distribution or visit http://www.sudo.ws/sudo/history.html for a short history of sudo.

`sudo` allows a permitted user to execute a command as the superuser or another user, as specified in the sudoers file. The real and effective uid and gid are set to match those of the target user as specified in the passwd file and the group vector is initialized based on the group file (unless the `-P` option was specified). If the invoking user is root or if the target user is the same as the invoking user, no password is required. Otherwise, sudo requires that users authenticate themselves with a password by default (NOTE: in the default configuration this is the user's password, not the root password). Once a user has been authenticated, a time stamp is updated and the user may then use sudo without a password for a short period of time (5 minutes unless overridden in sudoers).

When invoked as `sudoedit`, the `-e` option (described below), is implied.

`sudo` determines who is an authorized user by consulting the file `/etc/sudoers`. By running `sudo` with the `-v` option, a user can update the time stamp without running a command. If a password is required, `sudo` will exit if the user's password is not entered within a configurable time limit. The default password prompt timeout is 5 minutes.

If a user who is not listed in the sudoers file tries to run a command via `sudo`, mail is sent to the proper authorities, as defined at configure time or in the sudoers file (defaults to root). Note that the mail will not be sent if an unauthorized user tries to run `sudo` with the `-l` or `-v` option. This allows users to determine for themselves whether or not they are allowed to use `sudo`.

If `sudo` is run by root and the `SUDO_USER` environment variable is set, `sudo` will use this value to determine who the actual user is. This can be used by a user to log commands through sudo even when a root shell has been invoked. It also allows the `-e` option to remain useful even when being run via a `sudo-run` script or program. Note however, that the sudoers lookup is still done for root, not the user specified by `SUDO_USER`.

`sudo` can log both successful and unsuccessful attempts (as well as errors) to `syslog(3)`, a log file, or both. By default `sudo` will log via `syslog(3)` but this is changeable at configure time or via the sudoers file.

`sudo` accepts the following command line options:

-A Normally, if sudo requires a password, it will read it from the current terminal. If the `-A` (askpass) option is specified, a (possibly graphical) helper program is executed to read the user's password and output the password to the standard output. If the `SUDO_ASKPASS` environment variable is set, it specifies the path to the helper program. Otherwise, the value specified by the askpass option in sudoers(5) is used.

`-s` [command] The `-s` (shell) option runs the shell specified by the `SHELL` environment variable if it is set or the shell as specified in `passwd(5)`. If a command is specified, it is passed to the shell for execution. Otherwise, an interactive shell is executed.

– The `--` option indicates that sudo should stop processing command line arguments.

SECURITY NOTES

`sudo` tries to be safe when executing external commands.

There are two distinct ways to deal with environment variables. By default, the env_reset sudoers option is enabled. This causes commands to be executed with a minimal environment containing `TERM, PATH, HOME, SHELL, LOGNAME, USER` and `USERNAME` in addition to variables from the invoking process permitted by the `env_check` and `env_keep` sudoers options. There is effectively a whitelist for environment variables.

If, however, the `env_reset` option is disabled in sudoers, any variables not explicitly denied by the `env_check` and `env_delete` options are inherited from the invoking process. In this case, `env_check` and `env_delete` behave like a blacklist. Since it is not possible to blacklist all potentially dangerous environment variables, use of the default `env_reset` behavior is encouraged.

In all cases, environment variables with a value beginning with `()` are removed as they could be interpreted as bash functions. The list of environment variables that sudo allows or denies is contained in the output of `sudo -V` when run as root.

Please note that `sudo` will normally only log the command it explicitly runs. If a user runs a command such as `sudo su` or `sudo sh`, subsequent commands run from that shell will not be logged, nor will `sudo`'s access control affect them. The same is true for commands that offer shell escapes (including most editors). Because of this, care must be taken when giving users access to commands via `sudo` to verify that the command does not inadvertently give the user an effective root shell.

7.2.6 libmagic

Written by Måns Rullgård Initial libmagic implementation, and configuration. Christos Zoulas API cleanup, error code and allocation

handling.

`file` is a standard Unix program for recognizing the type of data contained in a computer file using magic number. Libmagic is the library which file uses, made available for Rodent to use directly.

The original version of file originated in Unix Research Version 4[1] in 1973. System V saw a major update with several important changes, most notably moving the file type information into an external text file rather than compiling it into the binary itself.

All major BSD and Linux distributions use a free, open-source reimplementation which was written in 1986-87 by Ian Darwin[2] from scratch. It was expanded by Geoff Collyer in 1989 and since then has had input from many others, including Guy Harris, Chris Lowth and Eric Fischer; from late 1993 onward its maintenance has been organized by Christos Zoulas.

7.2.7 ps: report a snapshot of the current processes

Written by by Branko Lankester <lankeste@fwi.uva.nl>. Michael K. Johnson <johnsonm@redhat.com> re-wrote it significantly to use the proc filesystem, changing a few things in the process. Michael Shields <mjshield@nyx.cs.du.edu> added the pid-list feature. Charles Blake <cblake@bbn.com> added multi-level sorting, the dirent-style library, the device name-to-number mmaped database, the approximate binary search directly on System.map, and many code and documentation cleanups. David Mossberger-Tang wrote the generic BFD support for psupdate. Albert Cahalan <albert@users.sf.net> rewrote ps for full Unix98 and BSD support, along with some ugly hacks for obsolete and foreign syntax.

SYNOPSIS

`ps [options]`

`ps` displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use `top(1)` instead.

This version of ps accepts several kinds of options:

1. UNIX options, which may be grouped and must be preceded by a dash.
2. BSD options, which may be grouped and must not be used with a dash.
3. GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and ps implementations that this ps is compatible with.

Note that `"ps -aux"` is distinct from `"ps aux"`. The `POSIX` and `UNIX` standards require that `"ps -aux"` print all processes owned by a user named `"x"`, as well as printing all processes that would be selected by the `-a` option. If the user named `"x"` does not exist, this `ps` may interpret the command as `"ps aux"` instead and print a warning. This behavior is intended to aid in transitioning old scripts and habits. It is fragile, subject to change, and thus should not be relied upon.

By default, `ps` selects all processes with the same effective user ID (`euid=EUID`) as the current user and associated with the same terminal as the invoker. It displays the process ID (`pid=PID`), the terminal associated with the process (`tname=TTY`), the cumulated CPU time in `[dd-]hh:mm:ss` format (`time=TIME`), and the executable name (`ucmd=CMD`). Output is unsorted by default.

The use of BSD-style options will add process state (`stat=STAT`) to the default display and show the command args (`args=COMMAND`) instead of the executable name. You can override this with the `PS_FORMAT` environment variable. The use of BSD-style options will also change the process selection to include processes on other terminals (`TTY`s) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so `-M` will be considered identical to `Z` and so on.

Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.

7.2.8 diff

Written by Paul Eggert, Mike Haertel, David Hayes, Richard Stallman, and Len Tower.

```
diff OPTIONS...  FILES...
```

In the simplest case, two file names `FROM-FILE` and `TO-FILE` are given, and '`diff`' compares the contents of `FROM-FILE` and `TO-FILE`. A file name of '`-`' stands for text read from the standard input. As a special case, '`diff - -`' compares a copy of standard input to itself.

If one file is a directory and the other is not, '`diff`' compares the file in the directory whose name is that of the non-directory. The non-directory file must not be '`-`'.

If two file names are given and both are directories, '`diff`' compares corresponding files in both directories, in alphabetical order; this comparison is not recursive unless the '`-r`' or '`--recursive`' option is given. '`diff`' never compares the actual contents of a directory as if it were a file. The file that is fully specified may not be standard input, because standard input is nameless and the notion of *file with the same name* does not apply.

If the '`--from-file=FILE`' option is given, the number of file names is arbitrary, and `FILE` is compared to each named file. Similarly, if the '`--to-file=FILE`' option is given, each named file is compared to `FILE`.

'`diff`' options begin with '`-`', so normally file names may not begin '`diff`' options begin with '`-`', so normally file names may not begin with '`-`'. However, '`--`' as an argument by itself treats the remaining arguments as file names even if they begin with '`-`'.

An exit status of 0 means no differences were found, 1 means some differences were found, and 2 means trouble. Normally, differing binary files count as trouble, but this can be altered by using the '`-a`' or '`--text`' option, or the '`-q`' or '`--brief`' option.

7.2.9 bash: GNU Bourne-Again SHell

Written by Brian Fox, Free Software Foundation bfox@gnu.org, Chet Ramey, Case Western Reserve University chet.ramey@case.edu

SYNOPSIS:

`bash [options] [file]`

Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash also incorporates useful features from the Korn and C shells (ksh and csh).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). Bash can be configured to be POSIX-conformant by default.

7.2.10 shred: Remove files more securely

Written by Colin Plumb.

'`shred`' overwrites devices or files, to help prevent even very expensive hardware from recovering the data.

Ordinarily when you remove a file (see section 7.2.2, the data is not actually destroyed. Only the index listing where the file is stored is destroyed, and the storage is made available for reuse. There are undelete utilities that will attempt to reconstruct the index and can bring the file back if the parts were not reused.

On a busy system with a nearly-full drive, space can get reused in a few seconds. But there is no way to know for sure. If you have sensitive data, you may want to be sure that recovery is not possible by actually overwriting the file with non-sensitive data.

However, even after doing that, it is possible to take the disk back to a laboratory and use a lot of sensitive (and expensive) equipment to look for the faint "echoes" of the original data underneath the overwritten data. If the data has only been overwritten once, it's not even that hard.

The best way to remove something irretrievably is to destroy the media it's on with acid, melt it down, or the like. For cheap removable media like floppy disks, this is the preferred method. However, hard drives are expensive and hard to melt, so the '`shred`' utility tries to achieve a similar effect non-destructively.

This uses many overwrite passes, with the data patterns chosen to maximize the damage they do to the old data. While this will work on floppies, the patterns are designed for best effect on hard drives. For more details, see the source code and Peter Gutmann's paper *Secure Deletion of Data from Magnetic and Solid-State Memory* (`http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html`), from the proceedings of the Sixth USENIX Security Symposium (San Jose, California, July 22-25, 1996).

Please note that '`shred`' relies on a very important assumption: that the file system overwrites data in place. This is the traditional way to do things, but many modern file system designs do not satisfy this assumption. Exceptions include:

- Log-structured or journaled file systems, such as those supplied with AIX and Solaris, and JFS, ReiserFS, XFS, Ext3 (in '`data=journal`' mode), BFS, NTFS, etc. when they are configured to journal *data*.
- File systems that write redundant data and carry on even if some writes fail, such as RAID-based file systems.
- File systems that make snapshots, such as Network Appliance's NFS server.
- File systems that cache in temporary locations, such as NFS version 3 clients.
- Compressed file systems.

In the particular case of ext3 file systems, the above disclaimer applies (and '`shred`' is thus of limited effectiveness) only in '`data=journal`' mode, which journals file data in addition to just metadata. In both the '`data=ordered`' (default) and '`data=writeback`' modes, '`shred`' works as usual. Ext3 journaling modes can be changed by adding the '`data=something`' option to the mount options for a particular file system in the '`/etc/fstab`' file, as documented in the `mount` man page (`man mount`).

If you are not sure how your file system operates, then you should assume that it does not overwrite data in place, which means that `shred` cannot reliably operate on regular files in your file system.

Generally speaking, it is more reliable to shred a device than a file, since this bypasses the problem of file system design mentioned above. However, even shredding devices is not always completely reliable. For example, most disks map out bad sectors invisibly to the application; if the bad sectors contain sensitive data, '`shred`' won't be able to destroy it.

'`shred`' makes no attempt to detect or report this problem, just as it makes no attempt to do anything about backups. However, since it is more reliable to shred devices than files, '`shred`' by default does not truncate or remove the output file. This default is more suitable for devices, which typically cannot be truncated and should not be removed.

Finally, consider the risk of backups and mirrors. File system backups and remote mirrors may contain copies of the file that cannot be removed, and that will allow a shredded file to be recovered later. So if you keep any data you may later want to destroy using '`shred`', be sure that it is not backed up or mirrored.

```
shred [OPTION]...  FILE[...]
```

`shred [OPTION]... FILE[...]`

The program accepts the following options. Also see section 7.3.8.

'-f' '–force' Override file permissions if necessary to allow overwriting.

'-NUMBER' '-n NUMBER' '–iterations=NUMBER' By default, `'shred'` uses 3 passes of overwrite. You can reduce this to save time, or increase it if you think it's appropriate. After 25 passes all of the internal overwrite patterns will have been used at least once.

'–random-source=FILE' Use `FILE` as a source of random data used to overwrite and to choose pass ordering.

'-s BYTES' '–size=BYTES' Shred the first `BYTES` bytes of the file. The default is to shred the whole file. `BYTES` can be followed by a size specification like `'K'`, `'M'`, or `'G'` to specify a multiple.

'-u' '–remove' After shredding a file, truncate it (if possible) and then remove it. If a file has multiple links, only the named links will be removed.

'-v' '–verbose' Display to standard error all status updates as sterilization proceeds.

'-x' '–exact' By default, `'shred'` rounds the size of a regular file up to the next multiple of the file system block size to fully erase the last block of the file. Use `'--exact'` to suppress that behavior. Thus, by default if you shred a 10-byte regular file on a system with 512-byte blocks, the resulting file will be 512 bytes long. With this option, shred does not increase the apparent size of the file.

'-z' '–zero' Normally, the last pass that `'shred'` writes is made up of random data. If this would be conspicuous on your hard drive (for example, because it looks like encrypted data), or you just think it's tidier, the `'--zero'` option adds an additional overwrite pass with all zero bits. This is in addition to the number of passes specified by the `'--iterations'` option.

You might use the following command to erase all trace of the file system you'd created on the floppy disk in your first drive. That command takes about 20 minutes to erase a "1.44MB" (actually 1440 KiB) floppy.

```
shred --verbose /dev/fd0
```

Similarly, to erase all data on a selected partition of your hard disk, you could give a command like this:

```
shred --verbose /dev/sda5
```

A `FILE` of `'-'` denotes standard output. The intended use of this is to shred a removed temporary file. For example:

```
i=‘mktemp‘ exec 3<>"$i" rm -- "$i" echo "Hello, world" >&3 shred
- >&3 exec 3>-
```

However, the command `'shred - >file'` does not shred the contents of `FILE`, since the shell truncates `FILE` before invoking `'shred'`. Use the command '`shred file`' or (if using a Bourne-compatible shell) the command '`shred - 1<>file`' instead.

An exit status of zero indicates success, and a nonzero value indicates failure.

7.2.11 ghostscript: PostScript and PDF language interpreter and pre- viewer

Artifex Software, Inc. are the primary maintainers of Ghostscript. Russell J. Lang, gsview at ghostgum.com.au, is the author of most of the MS Windows code in Ghostscript.

The `gs` command invokes `Ghostscript`, an interpreter of Adobe Systems' PostScript(tm) and Portable Document Format (PDF) languages. `gs` reads "files" in sequence and executes them as Ghostscript programs. After doing this, it reads further input from the standard input stream (normally the keyboard), interpreting each line separately. The interpreter exits gracefully when it encounters the "quit" command (either in a file or from the keyboard), at end-of-file, or at an interrupt signal (such as Control-C at the keyboard).

The interpreter recognizes many option switches, some of which are described below. Please see the usage documentation for complete information. Switches may appear anywhere in the command line and apply to all files thereafter. Invoking `Ghostscript` with the `-h` or `-?` switch produces a message which shows several useful switches, all the devices known to that executable, and the search path for fonts; on Unix it also shows the location of detailed documentation.

`Ghostscript` may be built to use many different output devices. To see which devices your executable includes, run "`gs -h`". Unless you specify a particular device, `Ghostscript` normally opens the first one of those and directs output to it, so if the first one in the list is the one you want to use, just issue the command

```
gs myfile.ps
```

On Unix and MS Windows systems you can also send output to a pipe. For example, to pipe output to the "`lpr`" command (which, on many Unix systems, directs it to a printer), use the option

```
-sOutputFile=%pipe%lpr
```

7.3 Apendix

7.3.1 Backup options

Some GNU programs (at least '`cp`', '`install`', '`ln`', and '`mv`') optionally make backups of files before writing new versions. These options control the details of these backups. The options are also briefly mentioned in the descriptions of the particular programs.

'-b' '–backup[=METHOD]' Make a backup of each file that would otherwise be overwritten or removed. Without this option, the original versions are destroyed. Use `METHOD` to determine the type of backups to make. When this option is used but `METHOD` is not specified, then the value of the `'VERSION_CONTROL'` environment variable is used. And if `'VERSION_CONTROL'` is not set, the default backup type is `'existing'`.

Note that the short form of this option, `'-b'` does not accept any argument. Using `'-b'` is equivalent to using `'--backup=existing'`.

This option corresponds to the Emacs variable `'version-control'`; the values for `METHOD` are the same as those used in Emacs. This option also accepts more descriptive names. The valid `METHOD`s are (unique abbreviations are accepted):

'none' 'off' Never make backups.

'numbered' 't' Always make numbered backups.

'existing' 'nil' Make numbered backups of files that already have them, simple backups of the others.

'simple' 'never' Always make simple backups. Please note `'never'` is not to be confused with `'none'`.

7.3.2 File permissions

Each file has a set of *file mode bits* that control the kinds of access that users have to that file. They can be represented either in symbolic form or as an octal number.

- Mode Structure:: Structure of file mode bits.

  The file mode bits have two parts: the *file permission bits*, which control ordinary access to the file, and *special mode bits*, which affect only some files.

  There are three kinds of permissions that a user can have for a file:

  1. permission to read the file. For directories, this means permission to list the contents of the directory.
  2. permission to write to (change) the file. For directories, this means permission to create and remove files in the directory.
  3. permission to execute the file (run it as a program). For directories, this means permission to access files in the directory.

  There are three categories of users who may have different permissions to perform any of the above operations on a file:

  1. the file's owner;
  2. other users who are in the file's group;
  3. everyone else.

Files are given an owner and group when they are created. Usually the owner is the current user and the group is the group of the directory the file is in, but this varies with the operating system, the file system the file is created on, and the way the file is created. You can change the owner and group of a file by using the '`chown`' and '`chgrp`' commands.

In addition to the three sets of three permissions listed above, the file mode bits have three special components, which affect only executable files (programs) and, on most systems, directories:

1. Set the process's effective user ID to that of the file upon execution (called the *set-user-ID bit*, or sometimes the *setuid bit*). For directories on a few systems, give files created in the directory the same owner as the directory, no matter who creates them, and set the set-user-ID bit of newly-created subdirectories.
2. Set the process's effective group ID to that of the file upon execution (called the *set-group-ID bit*, or sometimes the *setgid bit*). For directories on most systems, give files created in the directory the same group as the directory, no matter what group the user who creates them is in, and set the set-group-ID bit of newly-created subdirectories.
3. Prevent unprivileged users from removing or renaming a file in a directory unless they own the file or the directory; this is called the *restricted deletion flag* for the directory, and is commonly found on world-writable directories like '`/tmp`'.

For regular files on some older systems, save the program's text image on the swap device so it will load more quickly when run; this is called the *sticky bit*.

In addition to the file mode bits listed above, there may be file attributes specific to the file system, e.g., access control lists (ACLs), whether a file is compressed, whether a file can be modified (immutability), and whether a file can be dumped. These are usually set using programs specific to the file system. For example:

ext2 On GNU and GNU/Linux the file attributes specific to the ext2 file system are set using '`chattr`'.

FFS On FreeBSD the file flags specific to the FFS file system are set using '`chflags`'.

Even if a file's mode bits allow an operation on that file, that operation may still fail, because:

- the file-system-specific attributes or flags do not permit it; or
- the file system is mounted as read-only.

For example, if the immutable attribute is set on a file, it cannot be modified, regardless of the fact that you may have just run `‘chmod a+w FILE’`.

- Symbolic Modes:: Mnemonic representation of file mode bits.

  Symbolic modes represent changes to files' mode bits as operations on single-character symbols. They allow you to modify either all or selected parts of files' mode bits, optionally based on their previous values, and perhaps on the current `‘umask’` as well (see section 7.3.3).

  The format of symbolic modes is:

  `[ugoa...][+-=]PERMS...[,...]`

  where `PERMS` is either zero or more letters from the set `‘rwxXst’`, or a single letter from the set `‘ugo’`.

  The following sections describe the operators and other details of symbolic modes.

  - Setting Permissions (see section).
  - Copying Permissions (see section).
  - Changing Special Mode Bits(see section).
  - Conditional Executability (see section 7.3.4).
  - Multiple Changes (see section 7.3.5).
  - Umask and Protection (see section 7.3.3).

- Numeric Modes:: File mode bits as octal numbers.

  As an alternative to giving a symbolic mode, you can give an octal (base 8) number that represents the mode. This number is always interpreted in octal; you do not have to add a leading `‘0’`, as you do in *C*. Mode `‘0055’` is the same as mode `‘55’`.

  A numeric mode is usually shorter than the corresponding symbolic mode, but it is limited in that normally it cannot take into account the previous file mode bits; it can only set them absolutely. (As discussed in the next section, the set-user-ID and set-group-ID bits of directories are an exception to this general limitation.)

  The permissions granted to the user, to other users in the file's group, and to other users not in the file's group each require three bits, which are represented as one octal digit. The three special mode bits also require one bit each, and they are as a group represented as another octal digit. Here is how the bits are arranged, starting with the lowest valued bit:

Value Mode	Corresponding Mode Bit
	Other users not in the file's group:
1	Execute/search
2	Write
4	Read
	Other users in the file's group
10	Execute/search
20	Write
40	Read
	The file's owner:
100	Execute/search
200	Write
400	Read
	Special mode bits:
1000	Restricted deletion flag or sticky bit
2000	Set group ID on execution
4000	Set user ID on execution

For example, numeric mode '`4755`' corresponds to symbolic mode '`u=rwxs,go=rx`', and numeric mode '`664`' corresponds to symbolic mode '`ug=rw,o=r`'. Numeric mode '`0`' corresponds to symbolic mode '`a=`'.

- Directory Setuid and Setgid:: Set-user-ID and set-group-ID on directories.

  On most systems, if a directory's set-group-ID bit is set, newly created subfiles inherit the same group as the directory, and newly created subdirectories inherit the set-group-ID bit of the parent directory. On a few systems, a directory's set-user-ID bit has a similar effect on the ownership of new subfiles and the set-user-ID bits of new subdirectories. These mechanisms let users share files more easily, by lessening the need to use '`chmod`' or '`chown`' to share new files.

  These convenience mechanisms rely on the set-user-ID and set-group-ID bits of directories. If commands like '`chmod`' and '`mkdir`' routinely cleared these bits on directories, the mechanisms would be less convenient and it would be harder to share files. Therefore, a command like '`chmod`' does not affect the set-user-ID or set-group-ID bits of a directory unless the user specifically mentions them in a symbolic mode, or sets them in a numeric mode. For example, on systems that support set-group-ID inheritance:

  ```
  # These commands leave the set-user-ID and

  # set-group-ID bits of the subdirectories alone,

  # so that they retain their default values.
  ```

```
mkdir A B C
chmod 755 A
chmod 0755 B
chmod u=rwx,go=rx C
mkdir -m 755 D
mkdir -m 0755 E
mkdir -m u=rwx,go=rx F
```

If you want to try to set these bits, you must mention them explicitly in the symbolic or numeric modes, e.g.:

```
# These commands try to set the set-user-ID
# and set-group-ID bits of the subdirectories.
mkdir G H
chmod 6755 G
chmod u=rwx,go=rx,a+s H
mkdir -m 6755 I
mkdir -m u=rwx,go=rx,a+s J
```

If you want to try to clear these bits, you must mention them explicitly in a symbolic mode, e.g.:

```
# This command tries to clear the set-user-ID
# and set-group-ID bits of the directory D.
chmod a-s D
```

This behavior is a GNU extension. Portable scripts should not rely on requests to set or clear these bits on directories, as `POSIX` allows implementations to ignore these requests.

7.3.3 Umask and Protection

If the `USERS` part of a symbolic mode is omitted, it defaults to '`a`' (affect all users), except that any permissions that are *set* in the system variable '`umask`' are *not affected*. The value of '`umask`' can be set using the '`umask`' command. Its default value varies from system to system.

Omitting the `USERS` part of a symbolic mode is generally not useful with operations other than '`+`'. It is useful with '`+`' because it allows you to use '`umask`' as an easily customizable protection against giving away more permission to files than you intended to.

As an example, if '`umask`' has the value 2, which removes write permission for users who are not in the file's group, then the mode:

```
+w
```

adds permission to write to the file to its owner and to other users who are in the file's group, but *not* to other users. In contrast, the mode:

a+w
ignores 'umask', and *does* give write permission for the file to all users.

7.3.4 Conditional Executability

There is one more special type of symbolic permission: if you use 'X' instead of 'x', execute/search permission is affected only if the file is a directory or already had execute permission.

For example, this mode:

`a+X`

gives all users permission to search directories, or to execute files if anyone could execute them before.

7.3.5 Making Multiple Changes

The format of symbolic modes is actually more complex than described above (*note Setting Permissions::). It provides two ways to make multiple changes to files' mode bits.

The first way is to specify multiple OPERATION and PERMISSIONS parts after a USERS part in the symbolic mode.

For example, the mode:

`og+rX-w`

gives users other than the owner of the file read permission and, if it is a directory or if someone already had execute permission to it, gives them execute/search permission; and it also denies them write permission to the file. It does not affect the permission that the owner of the file has for it. The above mode is equivalent to the two modes:

`og+rX og-w`

The second way to make multiple changes is to specify more than one simple symbolic mode, separated by commas. For example, the mode:

`a+r,go-w`

gives everyone permission to read the file and removes write permission on it for all users except its owner. Another example:

`u=rwx,g=rx,o=`

sets all of the permission bits for the file explicitly. (It gives users who are not in the file's group no permission at all for it.)

The two methods can be combined. The mode:

`a+r,g+x-w`

gives all users permission to read the file, and gives users who are in the file's group permission to execute/search it as well, but not permission to write to it. The above mode could be written in several different ways; another is:

`u+r,g+rx,o+r,g-w`

7.3.6 Setting Permissions

The basic symbolic operations on a file's permissions are adding, removing, and setting the permission that certain users have to read, write, and execute or search the file. These operations have the following format:

```
USERS OPERATION PERMISSIONS
```

The spaces between the three parts above are shown for readability only; symbolic modes cannot contain spaces.

The `USERS` part tells which users' access to the file is changed. It consists of one or more of the following letters (or it can be empty; see section 7.3.3 for a description of what happens then). When more than one of these letters is given, the order that they are in does not matter.

'`u`' the user who owns the file;

'`g`' other users who are in the file's group;

'`o`' all other users;

'`a`' all users; the same as 'ugo'.

The `OPERATION` part tells how to change the affected users' access to the file, and is one of the following symbols:

'`+`' to add the `PERMISSIONS` to whatever permissions the `USERS` already have for the file;

'`-`' to remove the `PERMISSIONS` from whatever permissions the `USERS` already have for the file;

'`=`' to make the `PERMISSIONS` the only permissions that the `USERS` have for the file.

The `PERMISSIONS` part tells what kind of access to the file should be changed; it is normally zero or more of the following letters. As with the `USERS` part, the order does not matter when more than one letter is given. Omitting the `PERMISSIONS` part is useful only with the '`=`' operation, where it gives the specified `USERS` no access at all to the file.

'`r`' the permission the `USERS` have to read the file;

'`w`' the permission the `USERS` have to write to the file;

'`x`' the permission the `USERS` have to execute the file, or search it if it is a directory.

For example, to give everyone permission to read and write a regular file, but not to execute it, use:

```
a=rw
```

To remove write permission for all users other than the file's owner, use:

```
go-w
```

The above command does not affect the access that the owner of the file has to it, nor does it affect whether other users can read or execute the file.

To give everyone except a file's owner no permission to do anything with that file, use the mode below. Other users could still remove the file, if they have write permission on the directory it is in.

```
go=
```

Another way to specify the same thing is:

```
og-rwx
```

7.3.7 Symbolic links

The GNU system supports *soft links* or *symbolic links*. This is a kind of *file* that is essentially a pointer to another file name. Unlike hard links, symbolic links can be made to directories or across file systems with no restrictions. You can also make a symbolic link to a name which is not the name of any file. (Opening this link will fail until a file by that name is created.) Likewise, if the symbolic link points to an existing file which is later deleted, the symbolic link continues to point to the same file name even though the name no longer names any file.

The reason symbolic links work the way they do is that special things happen when you try to open the link. The '`open`' function realizes you have specified the name of a link, reads the file name contained in the link, and opens that file name instead. The '`stat`' function likewise operates on the file that the symbolic link points to, instead of on the link itself.

By contrast, other operations such as deleting or renaming the file operate on the link itself. The functions '`readlink`' and '`lstat`' also refrain from following symbolic links, because their purpose is to obtain information about the link. '`link`', the function that makes a hard link, does too. It makes a hard link to the symbolic link, which one rarely wants.

Some systems have for some functions operating on files have a limit on how many symbolic links are followed when resolving a path name. The limit if it exists is published in the '`sys/param.h`' header file.

7.3.8 Common options

Certain options are available in all of these programs. Rather than writing identical descriptions for each of the programs, they are described here. (In fact, every GNU program accepts (or should accept) these options.)

Normally options and operands can appear in any order, and programs act as if all the options appear before any operands. For example, '`sort -r passwd -t :`' acts like '`sort -r -t : passwd`', since '`:`' is an option-argument of '`-t`'. However, if the '`POSIXLY_CORRECT`' environment variable is set, options must appear before operands, unless otherwise specified for a particular command.

A few programs can usefully have trailing operands with leading '`-`'. With such a program, options must precede operands even if '`POSIXLY_CORRECT`' is not set, and this fact is noted in the program description. For example, the '`env`' command's options must appear before its operands, since in some cases the operands specify a command that itself contains options.

Most programs that accept long options recognize unambiguous abbreviations of those options. For example, '`rmdir --ignore-fail-on-non-empty`' can be invoked as '`rmdir --ignore-fail`' or even '`rmdir --i`'. Ambiguous options, such as '`ls --h`', are identified as such.

Some of these programs recognize the '`--help`' and '`--version`' options only when one of them is the sole command line argument. For these programs, abbreviations of the long options are not always recognized.

'–help' Print a usage message listing all available options, then exit successfully.

'–version' Print the version number, then exit successfully.

'–' Delimit the option list. Later arguments, if any, are treated as operands even if they begin with '`-`'. For example, '`sort -- -r`' reads from the file named '`-r`'.

A single '`-`' operand is not really an option, though it looks like one. It stands for standard input, or for standard output if that is clear from the context. For example, '`sort -`' reads from standard input, and is equivalent to plain '`sort`', and '`tee -`' writes an extra copy of its input to standard output. Unless otherwise specified, '`-`' can appear as any operand that requires a file name.

7.3.9 Target directory

The '`cp`', '`install`', '`ln`', and '`mv`' commands normally treat the last operand specially when it is a directory or a symbolic link to a directory. For example, '`cp source dest`' is equivalent to '`cp source dest/source`' if '`dest`' is a directory. Sometimes this behavior is not exactly what is wanted, so these commands support the following options to allow more fine-grained control:

'-T' '–no-target-directory' Do not treat the last operand specially when it is a directory or a symbolic link to a directory. This can help avoid race conditions in programs that operate in a shared area. For example, when the command '`mv /tmp/source /tmp/dest`' succeeds, there is no guarantee that '`/tmp/source`' was renamed to '`/tmp/dest`': it could have been renamed to '`/tmp/dest/source`' instead, if some other process created '`/tmp/dest`' as a directory. However, if '`mv -T /tmp/source /tmp/dest`' succeeds, there is no question that '`/tmp/source`' was renamed to '`/tmp/dest`'.

In the opposite situation, where you want the last operand to be treated as a directory and want a diagnostic otherwise, you can use the '`--target-directory`' ('`-t`') option.

'-t DIRECTORY' '–target-directory=DIRECTORY' Use `DIRECTORY` as the directory component of each destination file name.

The interface for most programs is that after processing options and a finite (possibly zero) number of fixed-position arguments, the remaining argument list is either expected to be empty, or is a list of items (usually files) that will all be handled identically. The '`xargs`' program is designed to work well with this convention.

The commands in the '`mv`'-family are unusual in that they take a variable number of arguments with a special case at the *end* (namely, the target directory). This makes it nontrivial to perform some operations, e.g., "move all files from here to `../d/`", because '`mv * ../d/`' might exhaust the argument space, and '`ls | xargs ...`' doesn't have a clean way to specify an extra final argument for each invocation of the subject command. (It can be done by going through a shell command, but that requires more human labor and brain power than it should.)

The '`--target-directory`' ('`-t`') option allows the '`cp`', '`install`', '`ln`', and '`mv`' programs to be used conveniently with '`xargs`'. For example, you can move the files from the current directory to a sibling directory, '`d`' like this:

`ls | xargs mv -t ../d --`

However, this doesn't move files whose names begin with '`.`'. If you use the GNU '`find`' program, you can move those files too, with this command:

`find .  -mindepth 1 -maxdepth 1 \ | xargs mv -t ../d`

But both of the above approaches fail if there are no files in the current directory, or if any file has a name containing a blank or some other special characters. The following example removes those limitations and requires both GNU '`find`' and GNU '`xargs`':

`find .  -mindepth 1 -maxdepth 1 -print0 \ | xargs --null --no-run-if-empty \ mv -t ../d`[10]

The '`--target-directory`' ('`-t`') and '`--no-target-directory`' ('`-T`') options cannot be combined.

7.3.10 Trailing slashes

Some GNU programs (at least '`cp`' and '`mv`') allow you to remove any trailing slashes from each `SOURCE` argument before operating on it. The '–strip-trailing-slashes' option enables this behavior.

This is useful when a `SOURCE` argument may have a trailing slash and specify a symbolic link to a directory. This scenario is in fact rather common because some shells can automatically append a trailing slash when performing file name completion on such symbolic links. Without this option, '`mv`', for example, (via the system's rename function) must interpret a trailing slash as a request to dereference the symbolic link and so must rename the indirectly referenced *directory* and not the symbolic link. Although it may seem surprising that such behavior be the default, it is required by `POSIX` and is consistent with other parts of that standard.

7.3.11 Treating '/' specially

Certain commands can operate destructively on entire hierarchies. For example, if a user with appropriate privileges mistakenly runs '`rm -rf / tmp/junk`', that may remove all files on the entire system. Since there are so few legitimate uses for such a command, GNU '`rm`' normally declines to operate on any directory that resolves to '`/`'. If you really want to try to remove all the files on your system, you can use the '`--no-preserve-root`' option, but the default behavior, specified by the '`--preserve-option`', is safer for most purposes.

[10]Now isn't that nerdy?

The commands 'chgrp', 'chmod' and 'chown' can also operate destructively on entire hierarchies, so they too support these options. Although, unlike 'rm', they don't actually unlink files, these commands are arguably more dangerous when operating recursively on '/', since they often work much more quickly, and hence damage more files before an alert user can interrupt them. Tradition and POSIX require these commands to operate recursively on '/', so they default to '--no-preserve-root', but using the '--preserve-root' option makes them safer for most purposes. For convenience you can specify '--preserve-root' in an alias or in a shell function.

Note that the '--preserve-root' option also ensures that 'chgrp' and 'chown' do not modify '/' even when dereferencing a symlink pointing to '/'.

7.3.12 Specifying the Time Zone with 'TZ'

In POSIX systems, a user can specify the time zone by means of the 'TZ' environment variable. For information about how to set environment variables, see section 7.3.13. The functions for accessing the time zone are declared in 'time.h'.

You should not normally need to set 'TZ'. If the system is configured properly, the default time zone will be correct. You might set 'TZ' if you are using a computer over a network from a different time zone, and would like times reported to you in the time zone local to you, rather than what is local to the computer.

In POSIX.1 systems the value of the 'TZ' variable can be in one of three formats. With the GNU C library, the most common format is the last one, which can specify a selection from a large database of time zone information for many regions of the world. The first two formats are used to describe the time zone information directly, which is both more cumbersome and less precise. But the POSIX.1 standard only specifies the details of the first two formats, so it is good to be familiar with them in case you come across a POSIX.1 system that doesn't support a time zone information database.

The first format is used when there is no Daylight Saving Time (or summer time) in the local time zone:

```
STD OFFSET
```

The STD string specifies the name of the time zone. It must be three or more characters long and must not contain a leading colon, embedded digits, commas, nor plus and minus signs. There is no space character separating the time zone name from the OFFSET, so these restrictions are necessary to parse the specification correctly.

The OFFSET specifies the time value you must add to the local time to get a Coordinated Universal Time value. It has syntax like ['+'|'-']HH[':'MM[':'SS]]. This is positive if the local time zone is west of the Prime Meridian and negative if it is east. The hour must be between '0' and '23', and the minute and seconds between '0' and '59'.

For example, here is how we would specify Eastern Standard Time, but without any Daylight Saving Time alternative:

EST+5

The second format is used when there is Daylight Saving Time:

STD OFFSET DST [OFFSET]‘,’START[‘/’TIME]‘,’END[‘/’TIME]

The initial STD and OFFSET specify the standard time zone, as described above. The DST string and OFFSET specify the name and offset for the corresponding Daylight Saving Time zone; if the OFFSET is omitted, it defaults to one hour ahead of standard time.

The remainder of the specification describes when Daylight Saving Time is in effect. The START field is when Daylight Saving Time goes into effect and the END field is when the change is made back to standard time. The following formats are recognized for these fields:

‘JN’ This specifies the Julian day, with N between ‘1’ and ‘365’. February 29 is never counted, even in leap years.

‘N’ This specifies the Julian day, with N between ‘0’ and ‘365’. February 29 is counted in leap years.

‘MM.W.D’ This specifies day D of week W of month M. The day D must be between ‘0’ (Sunday) and ‘6’. The week W must be between ‘1’ and ‘5’; week ‘1’ is the first week in which day D occurs, and week ‘5’ specifies the *last* D day in the month. The month M should be between ‘1’ and ‘12’.

The TIME fields specify when, in the local time currently in effect, the change to the other time occurs. If omitted, the default is ‘02:00:00’.

For example, here is how you would specify the Eastern time zone in the United States, including the appropriate Daylight Saving Time and its dates of applicability. The normal offset from UTC is 5 hours; since this is west of the prime meridian, the sign is positive. Summer time begins on the first Sunday in April at 2:00am, and ends on the last Sunday in October at 2:00am.

EST+5EDT,M4.1.0/2,M10.5.0/2

The schedule of Daylight Saving Time in any particular jurisdiction has changed over the years. To be strictly correct, the conversion of dates and times in the past should be based on the schedule that was in effect then. However, this format has no facilities to let you specify how the schedule has changed from year to year. The most you can do is specify one particular schedule —usually the present day schedule— and this is used to convert any date, no matter when. For precise time zone specifications, it is best to use the time zone information database (see below).

The third format looks like this:

:CHARACTERS

Each operating system interprets this format differently; in the GNU C library, CHARACTERS is the name of a file which describes the time zone.

If the ‘TZ’ environment variable does not have a value, the operation chooses a time zone by default. In the GNU C library, the default time zone is like the specification ‘TZ=:/etc/localtime’ (or ‘TZ=:/usr/local/etc/localtime’, depending on how GNU C library was configured. Other C libraries use their own rule for choosing the default time zone, so there is little we can say about them.

If `CHARACTERS` begins with a slash, it is an absolute file name; otherwise the library looks for the file '`/share/lib/zoneinfo/CHARACTERS`'. The '`zoneinfo`' directory contains data files describing local time zones in many different parts of the world. The names represent major cities, with subdirectories for geographical areas; for example, '`America/New_York`', '`Europe/London`', '`Asia/Hong_Kong`'. These data files are installed by the system administrator, who also sets '`/etc/localtime`' to point to the data file for the local time zone. The GNU C library comes with a large database of time zone information for most regions of the world, which is maintained by a community of volunteers and put in the public domain.

7.3.13 Environment Variables

When a program is executed, it receives information about the context in which it was invoked in two ways. The first mechanism uses the `ARGV` and `ARGC` arguments to its '`main`' function. The second mechanism uses "environment variables" and is discussed in this section.

The `ARGV` mechanism is typically used to pass command-line arguments specific to the particular program being invoked. The environment, on the other hand, keeps track of information that is shared by many programs, changes infrequently, and that is less frequently used.

The environment variables discussed in this section are the same environment variables that you set using assignments and the '`export`' command in the shell. Programs executed from the shell inherit all of the environment variables from the shell.

Standard environment variables are used for information about the user's home directory, terminal type, current locale, and so on; you can define additional variables for other purposes. The set of all environment variables that have values is collectively known as the "environment".

Names of environment variables are case-sensitive and must not contain the character '`=`'. System-defined environment variables are invariably uppercase.

The values of environment variables can be anything that can be represented as a string. A value must not contain an embedded null character, since this is assumed to terminate the string.

8 Modules

This section contains information about structural module design of interest only to programmers, so you may skip this section if you are not interested in looking under the hood and examining how the cables are wired up.

Modules are divided into two types, root modules and functional modules. The functionality of both type of modules is defined in terms of four basic types of functions which generate a functional space isomorphic to the set of complex numbers with rational real component and integer imaginary component. That might sound like a bunch of hogwash to someone without a graduate mathematics degree, but it works and is very flexible (both in practice and in theory).

Plugin functions may be specified as one of the following types.

void: These functions all return a void pointer and take no argument

natural: These functions all return a void pointer and take a single void pointer argument. The space of natural functions is isomorphic to the set of natural numbers.

rational: These functions all return a void pointer and take a two void pointer arguments. The space of rational functions is isomorphic to the set of rational numbers.

complex: These functions all return a void pointer and take a three void pointer arguments. The space of rational functions is isomorphic to the set of complex numbers with integer imaginary component.

The api for programming modules is scheduled to be released with Rodent Delta, along with the Programmer's Guide. This will allow to use the basic Rodent framework with any applicacion the programmer may deem fit, just as the plugins distributed with the Rodent package do.