Lemma 1 Ltd.
2nd Floor
31A Chain St.
Reading
Berks
RG1 2HX

ProofPower

Xpp User Guide

Abstract

This document is the user guide for xpp — the X Windows interface for the ProofPower specification and proof tools.

Version: 1.38
Date: 12 August 2009
Reference: LEMMA1/XPP/USR031
Pages: 21

Prepared by: R.D. Arthan
Tel: +44 118 958 4409
E-Mail: rda@lemma-one.com

0.1 Contents

0.2 List of Figures

0.3 References

[1] DS/FMU/IED/USR001. *ProofPower Document Preparation.* Lemma 1 Ltd.

1 INTRODUCTION

This document is the User Guide for xpp. Xpp provides a convenient way to prepare, check and execute ProofPower scripts. It combines a general purpose text editor with a command interface for operating the ProofPower tools such as the ProofPower-HOL and ProofPower-Z systems.

Xpp includes a help system that explains each of its user interface features. This document supplements the information in the help system.

2 OVERVIEW OF xpp

Xpp supports interactive development of ProofPower documents. As you develop a document, you can process it with the appropriate ProofPower tool to type-check your specifications, develop proofs, browse the theory hierarchy, or indeed carry out any other function supported by the ProofPower-ML interface to the tool in question.

By using an extended character set supported by custom X Windows fonts, xpp displays the most common mathematical symbols supported by ProofPower on the screen much as they appear on the printed page. The fonts are described in section 6 below. See ProofPower *Document Preparation* [1] for information on the facilities for using LaTeX to typeset your document.

Xpp sessions come in two guises: edit-only sessions and command interface sessions. An edit-only session with xpp is shown in figure 1 on page 5; Examples of command interface session are shown in figure 2 on page 6 and figure 3 on page 9.

In an edit-only session, you are just editing a ProofPower document. There are several features to help you work with the document as a document. For example, the more common mathematical symbols can easily be entered in your script using a Palette Tool, or via the keyboard using special shift characters. For some languages, such as, ProofPower-Z, the Template Tool gives you a quick way of entering common constructs, such as schema boxes.

In a command interface session you are both editing a ProofPower document and running an interactive session typically with one of the ProofPower-ML-based tools, such as ProofPower-HOL or ProofPower-Z. The main window of a command interface session contains two main text areas (see figure 2 on page 6): the *script window*, where you edit your ProofPower document and the *journal window* which displays a record of your transactions with the other tool (ProofPower-HOL in the example in figure 2 on page 6).

Figure 1: An Example `xpp` Edit-Only Session

Figure 2: An Example `xpp` Command Interface Session (i)

3 USING xpp

3.1 Starting xpp

Xpp is started from a Unix or Linux shell (command line) using the command xpp. The command line syntax for xpp is:

```
xpp [Standard X Toolkit options] [xpp option] ...
```

The Standard X Toolkit options allow you to control many aspects of the behaviour and appearance of xpp, e.g., if you are logged in to a remote system, you can tell xpp to use your local keyboard and screen (your "display", to use the X Windows Systems term) by using the -display option. Please consult the main man page for the X Windows System for detailed information about these options (which are called the "X Toolkit Intrinsics options" in that man page). On most systems, one of the commands man X or man X11 should display this man page.

The xpp options are described in the following table:

-b	"blocking": run in the foreground not in the background;
-c command [arg ...]	"command line": run command with the specified arguments, if any;
-d database[#theoryname]	"database": run ProofPower on the specified database making the specified theory (if supplied) the current theory at the start of the run;
[-f] name	"file": open file name for editing;
-h	"have fonts": do not attempt to load any fonts;
-i files	"include files": include the specified comma-separated list of files into the ProofPower run by executing them in turn before executing commands interactively;
-r	"read only": start with the read-only option turned on.
-- ml_flags	"ML flags": pass everything after "--" as flags to the Standard ML compiler in a ProofPower run.

In the form with -c command everything after command is taken as comprising the argument list [arg ...]. -f may be omitted if name is not itself an option or part of an option.

If neither -d database nor -c command [args ...] is specified, then xpp will run an editor session. If -d database is specified, xpp will run the program pp in a command session passing it any relevant options. If -c command is specified, xpp will run the command specified in a command session. It is an error to specify -d database and then -c command (if -d database appears after -c command it will be treated as part of the argument list for the command).

For example, the command line xpp myfile.doc -d hol, xpp will start up editing the file

`myfile.doc` and running a ProofPower session with the `hol` database.

If the file specified on the command line does not exist, `xpp` will put up a dialogue box with buttons labelled "New" and "Quit". If you click New, `xpp` will start up with the script window empty and with the file name set to what you specified on the file name. `xpp` will then try to create the file when you first save your work. If you click Quit, then `xpp` will quit immediately.

If you specify an empty string either for the file name or for command and args, `xpp` will start up displaying a dialogue box inviting you to enter the missing information interactively.

By default, `xpp` runs in the background, i.e., when you call `xpp` from the shell, the `xpp` user interface starts asynchronously and control returns immediately to the shell. If you want it to run in the foreground, specify `-b` on the command line.

The command line interface to `xpp` is configurable. The above description applies to the usual configuration to run ProofPower sessions. The command options are recognised using an option description string configured into `xpp`. See section 4.2.2 below for more information on this. If you are customising the configuration of `xpp` to run other commands, we recommend that you provide a separate script to run it with your variant configuration.

3.2 Working with the xpp Interface

By default, `xpp` will attempt to add the ProofPower font directory to the font search path if that has not been done. This is not done if the `-havefonts` option is specified.

Figure 2 on page 6 shows `xpp` being used to work with a ProofPower-HOL specification. The user is incrementally type-checking a specification. An HOL constant *concl* has just been entered and type-checked and then the user has used the Command Line tool, to enter a fragment of ProofPower-ML to find out more information about the types that have been inferred. The user has then selected the next paragraph of the specification, which is therefore highlighted in the script window. The next step will be to use Execute from the Command menu to enter the paragraph into ProofPower-HOL to be type-checked.

Figure 3 on page 9 shows `xpp` being used to develop a proof interactively with the subgoal package. The user has used the Command Line tool to load in a specification from a file, has then set up the goal by executing the line beginning *set_goal* and has just executed the first tactic (the line that is selecte in the journal window).

Consult the `xpp` help system for further information about working with the `xpp` interface. The Tutorial or Help items in the Help menu are the best place to start. Many `xpp` functions are provided by tools windows, such as the Command Line tool that can be seen in figure 2 on page 6. In most of these windows, there is a Help button that you can press to get help with using that window.

Figure 3: Example `xpp` Command Interface Session (ii)

4 CUSTOMISING xpp

4.1 Resource Settings

Xpp is customisable via its resource file (application defaults file). This resource file is a text file called Xpp which you can edit with any text editor (e.g., xpp itself!) to change the appearance and behaviour of xpp. The location of the resource files depends on your local X Windows System configuration. Typically, it will be in a subdirectory app-defaults of your $HOME directory. Please consult either the README file for Xpp or your local systems administrator for more information.

An example resource file for xpp is provided in the subdirectory app-defaults of the ProofPower installation directory. The example resource file uses #include directives to include two files XppKeyboard and XppTemplates which define the keyboard layoiut and the behaviour of the Templates Tool. XppKeyboard and XppTemplates are set up as symbolic links to other resource files in the same directory.

If you want to customise the example resource files, then it is best to copy the app-defaults directory preserving the symbolic links. A gzipped, tar archive of the directory is provided in the file app-defaults.tgz in the ProofPower installation directory. Unpacking this tar archive into your $HOME directory will give you a copy of the files and links ready for you to start customising.

A typical entry in the resource file has the form:

```
Xpp[<sep><widget name>].<resource name>:        <value>
```

Here the optional <sep><widget name> may identify the part of xpp whose behaviour is being controlled, <sep> is either an asterisk or a dot, <resource name> identifies the attribute you want to change and <value> is the value you want to give to the attribute. Sometimes <value> has to be a long text string, and in that case it can be split over several lines using a back-slash character at the end of each line but the last.

Some resource settings, referred to as application resources, are specific to xpp. The other resource settings in the example resource file are widget resources, i.e., they affect generic attributes of the Motif widgets that make up xpp. The application resources and the widget resources that are particularly significant in xpp are discussed in more detail in section 4.2 below Users familiar with setting up resource files for Motif applications are cautioned that the treatment of fontList and translations resources for text and text field widgets is special in xpp, because it arranges for these widget types to have a consistent appearance and behaviour. See section 4.2.2 for details.

Users typically configure xpp by adjusting the <value> settings in the example resource file supplied. For reference, a listing of the complete widget hierarchy for xpp is provided in the

file `widgets.txt`. This may be used to control the appearance of `xpp` in fine detail, e.g., to set the colour of individual menu entries. Consult the Motif manual pages for the various widget classes for more information.

4.2 Xpp-Specific Resources

4.2.1 Options Settings

The following resource settings correspond to options that you can also change while `xpp` is running using the Options Tool that you can bring up with the Tools menu.

`Xpp*take-backups.set` Allowed values: "True" or "False". See section 5.2 below.

`Xpp*delete-backups.set` Allowed values: "True" or "False". See section 5.2 below.

`Xpp*ignore-case.set` Allowed values: "True" or "False". When this option is "True", the Search and Replace Tool will ignore the case of letters when matching the search pattern with the text (i.e., 'a' and 'A' will be considered to be equal, etc.).

`Xpp*journal-max.value` Allowed values: a decimal number. In a command interface session, this limits the number of bytes of data that is remembered in the journal window[1]. Values less than 2000 are mapped to 2000.

`Xpp*read-only.set` Allowed values: "True" or "False". When this option is "True", you will be warned if you try to make changes to the text in the script window or try to save the file. This option may be set from the `xpp` command line using the `-readonly` option keyword. See section 5.5 for more details.

`Xpp*use-regular-expressions.set` Allowed values: "True" or "False". When this option is "True", the Search and Replace Tool will interpret the search pattern as a regular expression. See section 5.8 for more details.

`Xpp.addNewLineMode` Allowed values: 0, 1, or 2. In a command interface session, if the text selection does not end in a new-line character when you try to execute it, `xpp` may take special action depending on the value of this resource. 0 causes `xpp` automatically to add a new-line; 1 causes `xpp` to put up a dialogue box asking you what to do; 2 causes `xpp` to execute the selection unchanged without any intervention on your part. Values outside the range 0, 1, 2 are mapped to the nearest value in that range.

[1] If you scroll to the top of the journal window, you will see a message like "`**** Text lost when buffer exceeded 20000 bytes ****`" if the limit has been exceeded.

4.2.2 Other Settings

The following settings are processed when xpp first starts running and are not changeable via the Options Tool later.

`Xpp.commandLineList` Allowed values: a list of text strings separated by "`\n`". The Command Line Tool maintains a list of useful commands that you can add to as you go along. This resource specifies the initial contents of the list.

`Xpp*command-text.translations` Allowed values: a string representing what is known a translation table in the X Toolkit Intrinsics library (see the example files for the format). A translation table associates "actions" with key strokes. The text field containing the command in the Command Line Tool maintains a history of up to 40 commands that have been exeuted. To allow you to scroll through this history, xpp provides actions "`command-history-down`" and "`command-history-up`" in addition to the repertoire of standard Motif actions for a text field widget. These actions are assigned to the Page Down and Page Up keys in the example resource files.

`Xpp*journal.editable` Allowed values: "True" or "False". In a command session, this resource controls what happens if you try to enter text into the journal window. If the resource is false, the journal window will never get the keyboard input focus and any text typed will go into the script you are editing. If the resource is true, then the command line tool will pop up and the text will be diverted to the text area in the command line tool. Setting the resource to true may not work well with some window manager configurations, so you may need to experiment to see which setting is best for your configuration. The resource is set to false in the supplied example resource file.

`Xpp*mainpanes.orientation` Allowed values: "VERTICAL" or "HORIZONTAL". This resource determines whether the script window and journal window are positioned one above the other or side by side. If you do not specify this resource explicitly, the value is taken as "VERTICAL", i.e., the windows are positioned one above the other.

`Xpp*script.fontList` Allowed values: an X Windows font name (typically one of the fonts supplied with ProofPower). This is a standard Motif resource that is treated specially by xpp. The value of this resource is copied from the script window widget to all the text widgets that may be used for entering or displaying mathematical symbols. So, for example, if you try to set a different value for the `fontList` resource for the command window widget, it will have no effect.

`Xpp*startup-command-line.value` Allowed values: any string that is a valid UN*X command line, e.g., `pp -d hol`. When you start xpp from the New Command Session item in the Tools menu or by specifiying an empty string as the value of the `-command` line option, it starts up displaying a dialogue box asking you to type in the command to run. This resource gives a default value for the command line, which you can edit or just accept when you see this dialogue.

`Xpp.templates` Allowed values: a list of triples each comprising a bitmap file name, an insertion text string and a description text string; the values within each triple and the triples are separated by slash characters. This resource defines the behaviour of the Templates Tool. Each triple defines one button in the Templates Tool. The bitmap gives the icon used as the label for the button, the insertion text is the text that is entered into the script window when the button is pressed, and the description string is the descriptive text associated with that icon in the Templates Tool help dialogue.

The Templates Tool is limited to a maximum of 100 template buttons. If you specify more, entries after the first 100 will be ignored. The library function `XmGetPixmap` used to load the bitmap files has a rather complex search algorithm. If the environment variable `XBMLANGPATH` is set, then it gives a search path listing directories to be searched. If this environment variable is not set, a somewhat long list of directories including subdirectories of the directory identified by the environment variable `XAPPLRESDIR`, if set, is used. If neither of these environment variables is set, then the directories `$HOME` and `$HOME/bitmaps` will be searched as well as some other directories in the X Windows installation. If all else fails, `xpp` will look for the bitmaps in the bitmaps subdirectory of the ProofPower installation directory.

`Xpp.textTranslations` Allowed values: a string representing what is known a translation table in the X Toolkit Intrinsics library (see the example files for the format). A translation table associates "actions" with key strokes. This resource is used to override the translation table for all the various text areas used for text that may include mathematical symbols.

This resource is used in `xpp` to give keyboard short-cuts for entering text (typically mathematical symbols) and for executing commands. For the latter purpose, `xpp` adds an additional action "`execute`" to the repertoire of standard Motif actions in the script window widget. With no parameters this action causes the text selected in the script window to be executed (just like Execute Selection in the Command menu). With parameters, the action executes the text given as parameters.

The example resource file uses a `#include` directive to include the file named `XppKeyboard`. See section 4.3 below for more information.

`Xpp.defaultCommand` Allowed values: a string giving a UNIX command line to be executed if `xpp` is invoked with command line options other than the X WIndows toolkit options or the options defined for `xpp`. The example resource file sets this to `pp`, so that, for example, if called with the command line `xpp -d hol`, `xpp` will take `pp -d hol` as the command line to run.

`Xpp.argumentChecker` Allowed values: a string giving a UNIX command line to be executed if `xpp` is invoked with command line options other than the X WIndows toolkit options or the options defined for `xpp`. The other options are appended to this string and the result is executed as a UNIX command line before bringing up the graphical user interface. `xpp` exits if the command line exits with a non-zero response code. The intention is that the command line supplied should validate the options and exit with a non-zero response if the command line specified by the resource `Xpp.defaultCommand` is likely to fail. The example resource file sets this to `pp -V`.

`Xpp.optionString` Allowed values: a string specifying the `xpp` and command option syntax in the `getopts(1)` format. The default is appropriate for the ProofPower program `pp`. This string must include "`bcf:hr`" so that the `xpp` options are recognised correctly.

4.3 Keyboard Layout

The X Windows system defines a number of logical modifier keys: keys that affect the interpretation of other keys. By setting up suitable modifier keys and, if necessary, adjusting the resource setting `Xpp.textTranslations`, you can arrange to enter all the mathematical symbols in the ProofPower extended character set using the keyboard. The mapping of physical keys on your keyboard to modifier keys can be displayed and changed using the `xmodmap` program.

As keyboard layouts vary widely and some subsystems, such as window managers, impose constraints on the use of some keys, you may need to experiment a little to get a workable layout. With most modern keyboards, it is possible to arrange for four extended character shift key combinations (so that each key can generate six different symbols). The supplied resource files include an examnple file `XppKeyboard` that gives a mapping that works well with many keyboards.

If you run it with no command line options, `xmodmap` will display the current settings for the eight logical modifier keys supported by X Windows.

The keyboard layout `XppKeyboard` provided as the default is shown in figure 4. Each key cap in the diagram is labelled with the characters which that key can produce. The "Shift+key" combination is not shown, since it will be specific to the keyboard for the numeric keys. The bottom right part of the diagram shows which combination of modifier keys produce which character. Since the four shift combinations give some spare capacity, some keys have been mapped to multiple character sequences as shown in the table at the bottom left of the diagram. The example uses the logical modifier keys "Mod4", "Mod5" and "Shift" to give four extended character shift combinations. See the comments in the file `XppKeyboard` for some hints on how to proceed on systems with keyboards that offer more limited possibilities for the modifier keys. You may wish to change the details of the assignments of symbols to key combinations in `XppKeyboard`. The octal codes you will need for this purpose are shown in the table in section 6 below.

5 SYSTEM INTERFACES

Section 5.1 to 5.8 below explain some of the interactions between `xpp`, the operating system and the X Windows System.

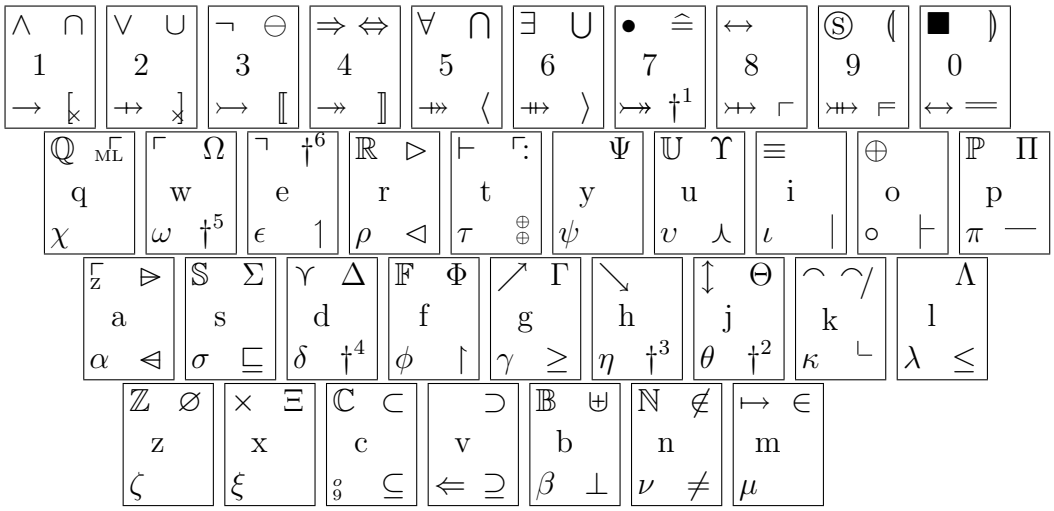

Figure 4: Example Keyboard Layout: `XppKeyboard`

5.1 Invocation and Environment

The behaviour of xpp and can be influenced by environment variables and by the configuration of the X server. These can be set up before xpp is invoked, but if xpp is installed in the normal way, it will provide defaults which should be suitable for most users. This process is explained in more detail below.

If the variable PPHOME has not been set in the calling environment, xpp will attempt to locate the ProofPower installation directory and then set up its PATH environment variable by prefixing the setting in the calling environment with the full path name of the bin subdirectory of the installation directory. If PPHOME has been set, xpp will use its value as the name of the installation directory in any subsequent actions that require it, but will leave the PATH environment variable unchanged.

If neither of the environment variables XUSERFILESEARCHPATH of XAPPLRESDIR has been set in the calling environment, xpp will set XUSERFILESEARCHPATH in its environment to the value $HOME/app-defaults/%N:$HOME/%N:<inst_dir>/app-defaults/%N, where <inst_dir> stands the name of the installation directory as determined above. This means that if the calling environment has not set XUSERFILESEARCHPATH or XAPPLRESDIR, the resource file for this run of xpp will be found by searching for a file named Xpp in the app-defaults directory of the user's home directory, then in the user's home directory, then in the app-defaults subdirectory of the installation directory.

If the -havefonts option has not been specified, xpp will check if the ProofPower fonts are available and if not, will add the fonts subdirectory of the installation directory to the font search path (by making the library calls equivalent to an xset(1) command with an appropriate fp option). If the -havefonts option has been specified, xpp will not change the font search path.

xpp attempts to load the bitmaps for the templates tool as discussed in section 4.2.2 above using the bitmaps subdirectory of the installation directory as the default location for any bitmap that is not found by the standard XmGetPixmap search algorithm.

5.2 Backups

When you select the Save function in the file menu, xpp may take a backup copy before it overwrites your file. Note that this only applies to the Save function, the Save As and Save Selection As functions will always prompt you before overwriting an existing file, but will not take a backup for you if you decide to overwrite the file.

What xpp does depends on the value of the resources named Xpp*take-backups.set and Xpp*delete-backups.set. Both these resources are set to "True" in the example resource file supplied with xpp and these are the recommended settings for normal use. See

section 4.2.1 for how to change these settings, if you need to.

If `Xpp*take-backups.set` is"True", the Save function will take backups. `xpp` derives the name of the backup file by appending "`.xpp.backup`" to the name of your file.

If `Xpp*delete-backups.set` is "True", the Save function will delete the backup file if your file has been successfully overwritten. The backup file will not be deleted if `xpp` was unable to overwrite your file (e.g., if the file system is full).

Some Unix and Linux file system types, e.g., MS-DOS file systems, may not support the file names that `xpp` uses for its backup files. `xpp` will detect this situation and prompt you for confirmation if the backup cannot be taken for this reason.

5.3 Signals

This section assumes you know a little about Unix and Linux signals. E.g., see the `man` page for `kill(1)`. `Xpp` recognises all the signals supported on your operating sytem that are defined by the Single UNIX Standard (SUS V3) and also some others that are specific to particular operating systems. The response of `xpp` to various types of signal is shown in the following table.

Non-fatal signals:	
SIGINT	`Xpp` behaves as if you had selected the Quit function from the file menu, i.e., if the file you are editing has not been saved or if the application is still running, `Xpp` will ask you for confirmation whether to quit, and will exit with a 0 return status. if you choose to quit.
SIGHUP	`Xpp` ignores this signal.
SIGTSTP SIGWINCH Etc.	`Xpp` takes the default action specified by the operating system for the signal in question.
Fatal signals:	
SIGBUS SIGFPE SIGSEGV SIGSYS Etc.	`Xpp` will try to make an emergency backup of your file. The emergency backup file will have a name of the form "`xpp.panic.`*XXXXXX*" where "*XXXXXX*" is some combination of letters and numbers. `Xpp` will then send an error message, including the name of the emergency backup file to the standard error channel and exit with a non-zero return status.

5.4 Application-Modal Dialogues

Often xpp puts up a dialogue, e.g., to ask you to select a file, and you must respond to the dialogue before xpp can do anything else. These are called *application-modal dialogues* in the X Windows System terminology.

Most window managers ensure that application-modal dialogues are never obscured by any other window in the application itself. However, some window managers can be configured in such a way that an application-modal dialogue can get hidden behind another xpp window. When this happens xpp will appear to freeze. You can solve this by sliding the visible xpp windows to the edge of the screen until you find the hidden dialogue window.

5.5 Read-only Option and File Access Permissions

xpp has a read-only option which is intended to help you work with files for which you do not have write access. You can also use this option to protect against accidental changes to files that you do not wish to change. This option can be set when you start xpp using the -readonly option keyword or interactively using the Options Tool. The option can also be set in the resource file (see section 4) and doing so is equivalent to setting it on the command line. The option is set automatically by xpp when you open a file for which you do not have write access.

When the read-only option is turned on you will be warned if you try to make a change to the text in the script window or if you try to save the file using Save in the File menu. You will not be warned if you save the file under a new name using Save As in the File menu.

If you do not specify -readonly when you start xpp, you will be warned whenever you open a file for which you do not have write access. The read-only option will be set automatically to match your write access for the file you are currently editing. When you save a file using Save As in the File menu, the read-only option is automatically turned off if the save operation succeeds.

If you are planning to work with several files for which you do not have write access and do not wish to be warned every time you open one of them, specifying -readonly when you start xpp will suppress the warnings. If you do this, turning the read-only option off using the Options Tool will enable the warnings.

Some editors have a facility for overriding the access permissions of a file (e.g., the w! command in vi). This kind of facility is not provided by xpp, which never attempts to change the access permissions of a file. If you need to change permissions, then use the chmod(1) command from the Unix or Linux shell.

If you are running as the super-user, xpp will consider any file you open to be read-only unless

it is owned by the super-user and has owner write-permission.

5.6 Working with Microsoft and Apple Macintosh Files

xpp lets you work with text files created using the Microsoft and the Apple Macintosh operating systems as well as with text files created on Unix. The three text file formats are distinguished by the characters used to terminate lines of text (line-feed on Unix, carriage-return on Macintosh, and carriage-return/line-feed pairs on Microsoft operating systems).

When a file is opened, xpp checks the line terminators to determine the type of the file, which is recorded as one of "Unix", "MS-DOS" or "Macintosh" in the File Type menu in the Options Tool. If the file being opened uses a mixture of line terminatiors, xpp will report the problem and record the file type as "Unix". When a file is saved, xpp uses the setting of the File Type menu in the Options Tool to select which line terminators to use. You may change this setting to convert between the different formats.

5.7 Working with Binary Files

xpp is not designed as an editor for files containg binary data: i.e., control characters other than tab, line-feed and carriage return. It can be used to inspect such files, but any uneditable characters are converted into question mark characters and will be saved as question mark characters if you save the file. This is intended, for example, to let you recover text from a file that has been partially corrupted. If you open a file containing binary data, you will be warned that the uneditable characters have been changed and the read-only option will be set to remind you that saving the file may cause problems.

5.8 Regular Expressions

When the option `Xpp*use-regular-expressions.set` is "True", xpp treats the search pattern in the Search and Replace Tool as a regular expression. xpp uses the POSIX-compliant regular expression library supplied with your system. Some of the more useful regular expressions forms are shown in the following table:

c	matches the character *c*, provided *c* is a character, such as a letter, digit, space, tab or new-line that does not have a special meaning.
c	matches the character *c*, whether or not *c* is a character such as * which would otherwise have a special meaning.
.	matches any character other than new-line.
[*list*]	matches any character in *list*. *list* may include character ranges such as a-z and character classes such as [:alpha:], and [:digit:].
[^*list*]	matches any character other than new-line that is not in *list*.
^	matches the empty string at the beginning of a line.
$	matches the empty string at the end of a line.
*e**	matches zero or more occurrences of the regular expression *e*.
e+	matches one or more occurrences of the regular expression *e*.
e{*m*,*n*}	matches any sequence of at least *m* and at most *n* occurrences of the regular expression *e*.
*e*1|*e*2	matches anything that matches either or both of the regular expressions e_1 and e_2. The | operator has lower precedence than the postfix operators *, etc.
(*e*)	matches anything that matches the regular expression *e*. Brackets enable you to control the operator precedence. E.g., a|b* matches either a single 'a' or any string of 'b's, whereas (a|b)* matches any string of 'a's and 'b's.

In xpp, matches with empty strings are only allowed for sub-expressions of the search pattern. The Search and Replace Tool considers a search operation that matches an empty string to be unsuccessful.

In replace operations in xpp, '&' and '\' have special meanings in the replacement text when regular expression searching is turned on. The character '&' stands for a copy of the text being replaced. Xpp has a set of registers numbered 0 to 9 to hold information about the last successful search. The contents of these reegisters can be included in the replacement text using a '\' followed by the register number. After a search, register 0 holds the entire matched text (so '\0' is the same as '&' immediately after a search), while registers 1, 2, ... and 9 hold the string that matched the first, second, ..., and ninth bracketed sub-expressions of the search pattern. If *c* is not a decimal digit, then '*c*' stands for the character *c*, so that you can include a backslash or an ampersand in your replacement using '\\' or '&'.

Consult your local UNIX or Linux manual pages (typically under regex(5) or regex(7)) for more information on the regular expression syntax supported on your system. xpp uses the POSIX extended regular expression syntax.

6 THE MATHEMATICAL FONTS

`Xpp` is supplied with three fonts which each provide the same mathematical character set: `holnormal` is a 16-point Roman font; `holsans10` is a 10-point sans-serif font and `holdouble` is a 32-point Roman font. The character codes and their corresponding images are shown in the following table:

Hex	Octal	Image	Hex	Octal	Image	Hex	Octal	Image	Hex	Octal	Image
80	200	$\subseteq$	A0	240	$\subset$	C0	300	$\cup$	E0	340	$\rightarrowtail$
81	201	$\trianglerighteq$	A1	241	$\cap$	C1	301	α	E1	341	$\trianglelefteq$
82	202	$\uplus$	A2	242	$\rangle$	C2	302	β	E2	342	$\bot$
83	203	$\mathbb{U}$	A3	243	$\ominus$	C3	303	$\sqsubseteq$	E3	343	$\Leftarrow$
84	204	Δ	A4	244	$\Leftrightarrow$	C4	304	δ	E4	344	$\supset$
85	205	$\circ$	A5	245	$\bigcap$	C5	305	ϵ	E5	345	$\supseteq$
86	206	Φ	A6	246	$\widehat{=}$	C6	306	ϕ	E6	346	$\mathbb{F}$
87	207	Γ	A7	247	$\langle$	C7	307	γ	E7	347	$\nearrow$
88	210	$\llcorner$	A8	250	$\langle\!\langle$	C8	310	η	E8	350	$\searrow$
89	211	Υ	A9	251	$\rangle\!\rangle$	C9	311	ι	E9	351	$\equiv$
8A	212	Θ	AA	252	$\leftrightarrow$	CA	312	θ	EA	352	$\updownarrow$
8B	213	$\frown\!\!\!/$	AB	253	$\oplus$	CB	313	κ	EB	353	$\frown$
8C	214	Λ	AC	254	$\ulcorner$	CC	314	λ	EC	354	$\upharpoonleft$
8D	215	$\in$	AD	255	$\rightarrow$	CD	315	μ	ED	355	$\mapsto$
8E	216	$\notin$	AE	256	$\urcorner$	CE	316	ν	EE	356	$\mathbb{N}$
8F	217	$\rightarrowtail\!\!\!\!\rightarrow$	AF	257	$\mathbb{R}$	CF	317	$\twoheadrightarrow\!\!\!\!\rightarrow$	EF	357	$\twoheadrightarrow$
90	220	Π	B0	260	$\blacksquare$	D0	320	π	F0	360	$\mathbb{P}$
91	221	$\mathrm{M}\!\!\ulcorner\mathrm{L}$	B1	261	$\wedge$	D1	321	χ	F1	361	$\ulcorner\!\!\!\mathrm{z}$
92	222	$\triangleright$	B2	262	$\vee$	D2	322	ρ	F2	362	$\triangleleft$
93	223	Σ	B3	263	$\neg$	D3	323	σ	F3	363	$\mathbb{Q}$
94	224	$\ulcorner:$	B4	264	$\Rightarrow$	D4	324	τ	F4	364	$\vdash$
95	225	Υ	B5	265	$\forall$	D5	325	υ	F5	365	$($
96	226	$\mathbb{B}$	B6	266	$\exists$	D6	326	$\mathbb{C}$	F6	366	$\overline{)}$
97	227	Ω	B7	267	$\bullet$	D7	327	ω	F7	367	$\vdash$
98	230	Ξ	B8	270	$\times$	D8	330	ξ	F8	370	[spare]
99	231	Ψ	B9	271	$\circledS$	D9	331	ψ	F9	371	$\upharpoonright$
9A	232	$\varnothing$	BA	272	$\overset{\oplus}{\oplus}$	DA	332	ζ	FA	372	$\mathbb{Z}$
9B	233	$\curlywedge$	BB	273	$\overset{\circ}{9}$	DB	333	$\lfloor\!\times$	FB	373	$[\![$
9C	234	$=\!\!=$	BC	274	$\leq$	DC	334	$\mid$	FC	374	—
9D	235	$\models$	BD	275	$\neq$	DD	335	$\rfloor\!\!\searrow$	FD	375	$]\!]$
9E	236	$\rightarrowtail\!\!\!\!\twoheadrightarrow$	BE	276	$\geq$	DE	336	$\bigcup$	FE	376	$\rightarrowtail\!\!\!\!\rightarrow$
9F	237	$\twoheadrightarrow$	BF	277	$\mathbb{S}$	DF	337	$\twoheadrightarrow$	FF	377	$\ulcorner$