10. SESSIONS, INITIALIZATION, AND RECOVERY

This section of the manual describes the life-cycle of an edit session. We begin with the definition of an edit session and what that means to elvis. This is followed by sections discussing initialization and recovery after a crash.

10.1 Sessions

Elvis is eventually expected to meet the COSE standards, which require (among other things) that programs be able to save their state so that they can be restarted later. It isn't required to restart in *exactly* the same state, but it should come as close as possible.

For elvis, this means that edit sessions should be restartable. It is possible to begin an edit session with one elvis process, exit that process, and then later start a new elvis process which resumes the previous edit session.

To accomplish this, elvis stores its state in a file, called the session file. For all practical purposes, the session file **is** the session.

The name of the session file is stored in the session option. By default, this will be a file in your home directory, named "elvis*.ses", where "*" represents a number chosen at run-time to make the file name unique. You can specify some other name for the session file via the **-s***session* command-line flag.

If the session file doesn't already exist when elvis starts running, then elvis will create it.

When elvis exits, it will normally delete the session file if this is the elvis process that created it. If the session file was left over from some other elvis process, then elvis will not delete it upon exiting. This is controlled by the tempsession option; if you don't like elvis' default behavior then you can change it.

10.2 Initialization

Before discussing elvis' initialization, let me just say that if you're having trouble configuring elvis, you might want to try invoking elvis with the command line flag **-VVV**, which causes elvis to write status information to stdout/stderr so you can see what it is doing. The flag **-o***logfile* will redirect this information to a file named *logfile*. Windows programs such as **WinElvis.exe** aren't allowed to write anything to stdout, so you *must* use **-o***logfile* any time you use **-VVV**. Now, back to the topic at hand...

Elvis begins by initializing some options to hardcoded values.

Elvis then chooses which user interface it should use. Elvis does this by scanning the command line arguments for a **-G***gui* flag; if there is no such flag, then elvis tests each user interface and uses the best one that is expected to work. (For example, the "x11" interface is expected to work if there is a DISPLAY environment variable and the X server is accessible. If not, then the "x11" interface is rejected and some other interface is used.)

The session file is then opened or created. For preexisting session files, elvis scans the session file for any buffers in it, and adds them to its internal list. Elvis can even reload the "undo" versions of some buffers.

Elvis searches through the directories named in the elvispath option for a file named "elvis.ini". If it finds that file, then it loads it into a buffer named "Elvis initialization" and executes its contents as a series of ex commands. See section 10.2.1 for description of the default contents of this file.

After that, it attempts to similarly load some other files, but they aren't executed. Some of them will be executed later. These files are:

FILE NAME	BUFFER NAME	PURPOSE
elvis.msg	Elvis messages	used to translate messages
elvis.brf	Elvis before reading	executed before loading file
elvis.arf	Elvis after reading	executed after loading file
elvis.bwf	Elvis before writing	executed before saving file
elvis.awf	Elvis after writing	executed after saving file

The "elvis.msg" file is described in section 11: Messages. The other files are described later in this section.

The next step in initialization is to load the first file and display it in a window. To do this, it first creates an empty buffer with the same name as the file. It then executes the "Elvis before reading" buffer (if it exists) on the empty buffer. The file's contents are then read into the buffer. Then the "Elvis after reading" buffer (if it exists) is executed on the new buffer. Finally, elvis creates a new window that shows the new buffer.

If the **-a** flag was given on the command line, then elvis will repeat the above steps for each file named on the command line. On the other hand, if no filenames were given on the command line then elvis will simply create a single untitled buffer and a window that shows it.

10.2.1 The "elvis.ini" file

The "elvis.ini" file is loaded into a buffer named "Elvis initialization". That buffer is then executed before any other initialization files are loaded. If the session file is later restarted, this script will be executed again at that time. Here's a line-by-line analysis of the default "elvis.ini" file...

```
" DEFINE SOME DIGRAPHS
if os=="msdos" || os=="os2" || (os=="win32" && gui!="windows")
```

```
then source! (elvispath("elvis.pc8"))
else source! (elvispath("elvis.lat"))
```

This attempts to locate the "elvis.lat" or "elvis.pc8" file and execute it. Those files contain ex scripts, consisting of a bunch of :digraph commands that set up the digraph table appropriately for the Latin-1 symbol set. The "!" at the end of the :source command name causes :source to silently ignore errors.

```
" CHOOSE SOME DEFAULT OPTION VALUES BASED ON THE INVOCATION NAME
let p=tolower(basename(program))
if p == "ex" || p == "edit"
then set! initialstate=ex
if p == "view"
then set! defaultreadonly
if p == "edit" || p == "vedit"
then set! novice
if home == ""
then let home=dirdir(program)
```

These lines initialize certain options according to the name by which elvis was invoked. Traditionally, invoking vi by the name "ex" causes it to start up in ex mode instead of vi mode, and "view" causes the files to be treated as readonly.

```
" SYSTEM TWEAKS GO HERE
"
" The Linux console can't handle colors and underlining.
if gui=="termcap"
then {
 if term=="linux"
 then set! nottyunderline
}
```

This is an attempt to work around a bug in the Linux console driver. The Linux console can't mix color attributes with the underline attribute.

```
" WINDOWS DEFAULT COLORS GO HERE (may be overridden in elvis.rc file)
if gui=="windows"
then {
 color e green
 color i magenta
 color u blue
 color f red
}
" X11 DEFAULT COLORS AND TOOLBAR GO HERE (may be overridden in .exrc fil
e)
if gui=="x11"
then so! (elvispath("elvis.x11"))
```

These lines set the defaults for the "windows" and "x11" user interfaces.

Note that "x11" configuration commands are actually stored in a separate file. This is because there are large number of commands for setting up the toolbar, and I didn't want to force other GUIs to read them just to ignore them. You should set the defaults in "elvis.x11", and *not* in an app-defaults file. If you aren't using the "x11" user interface, then these lines have no effect.

```
" EXECUTE THE STANDARD CUSTOMIZATION SCRIPTS
let f=(os=="unix" ? ".elvisrc" : "elvis.rc")
if $EXINIT
then eval $EXINIT
else source! (exists("~"/f) ? "~"/f : "~/.exrc")
if exrc && getcwd()!=home
then safer! (exists(f) ? f : ".exrc")
set f=""
```

These lines set the <u>f</u> option to either ".elvisrc" or "elvis.rc", whichever is appropriate for your operating system. They then check whether an environment variable named "EXINIT" is set to a non-empty value. If so, then the value of EXINIT is executed as an ex command line; otherwise the ".elvisrc" or "elvis.rc" file in your home directory is executed, if it exists. If that file doesn't exist, then it tries ".exrc"... which probably only makes sense for Unix, but it is quicker to try & fail then to test before trying. The "~" notation is UNIX's conventional alias for referring to files in your home directory; elvis handles it correctly on non-UNIX systems too.

Note: There is a hardcoded limit of (normally) 1023 characters for the result of an expression. If your EXINIT environment variable's value is longer than that, elvis won't be able to execute it.

If EXINIT or .elvisrc/elvis.rc/.exrc (whichever was executed) has set the <u>exrc</u> option then elvis will execute ".elvisrc" or "elvis.rc" in the current directory, if it exists; if not, then it tries ".exrc". Elvis uses <u>:safer</u> instead of <u>:source</u> to execute the file for security reasons.

```
" X11 INTERFACE DEFAULT FONTS GO HERE
if gui == "x11"
then if normalfont == ""
then {
 set! normalfont="*-courier-medium-r-*-18-*"
 set! boldfont="*-courier-bold-r-*-18-*"
 set! italicfont="*-courier-medium-o-*-18-*"
}
```

These cause the x11 interface to use 18-point courier fonts, if you don't explicitly name some other font on the command line (**-font** *fontname*) or by setting the normalfont option in your .exrc file.

10.2.2 The "elvis.brf" file

The "elvis.brf" file is loaded into a buffer named "Elvis before reading". That buffer is executed immediately before loading any user file into a user buffer.

```
" TAKE A GUESS AT THE BUFFER'S TYPE
let! readeol=fileeol(filename)
```

This line tries to guess whether the file is binary or not. This must be done before the file is loaded because for non-binary files elvis converts newlines to linefeeds as it reads the file.

10.2.3 The "elvis.arf" file

The "elvis.arf" file is loaded into a buffer named "Elvis after reading". That buffer is automatically executed immediately after a user file has been loaded into a user buffer.

```
" TAKE A GUESS AT THE BUFFER'S PREFERRED DISPLAY MODE
let e=tolower(dirext(filename))
if knownsyntax(filename)
then set! bufdisplay=syntax
if os=="unix" && buflines >= 1
then 1s/^#! *[^ ]*\/\([^ ]\+\).*/set! bufdisplay="syntax \1"/x
if !newfile
then {
 if readeol=="binary" && bufdisplay=="normal"
 then set! bufdisplay=hex
 if e==".man"
 then set! bufdisplay=man
 if strlen(e)==2 && isnumber(e>>1) && buflines>=1
 then 1s/^\./set! bufdisplay=man/x
 if e==".tex"
 then set! bufdisplay=tex
 if e<<4==".htm"
 then set! bufdisplay=html
 if buflines >= 1 && bufdisplay=="hex"
 then 1s/^<[HIThit!]/set! bufdisplay=html/x
 if (filename<<5=="http:" || filename<<4=="ftp:")
                                  && strlen(e)<4 && bd=="hex"
 then set! bufdisplay=normal
 if bufdisplay=="normal" && buflines >= 1
 then 1s/^From .*/set! bufdisplay="syntax email"/x
 if dirdir(filename)=="/tmp" || dirdir(filename)=="/var/tmp"
 then set! bufdisplay="syntax email"
}
```

These lines try to guess the preferred display mode for the file. First it checks to see if the filename's extension is listed in the <u>elvis.syn</u> file; if so, then the buffer is shown in the <u>syntax</u> display mode. Then, for UNIX, if the first line of the file starts with "#!shell", elvis will use the <u>syntax</u> display mode for that named shell. This is followed by many special cases.

```
" EXECUTE MODELINES, IF "modelines" OPTION IS SET
if modelines && buflines >= 1 && buflines <= modelines * 2
then %s/ex:\(.*\):/\1/x
if modelines && buflines > modelines * 2
then {
 eval 1,(modelines)s/[ev][xi]:\\\(.*\\\):/\1/x
 eval (buflines - modelines + 1),(buflines)
                                  s/[ev][xi]:\\\(.*\\\):/\1/x
}
```

These commands search for modelines in the newly loaded file, if the modelines option is set. The modelines are executed via the new "x" option to the :s command.

Note: The second "eval" line is split above merely as a typographical convenience. In the real "elvis.arf" file, the "eval" line and "s" line are actually a single line.

10.2.4 The "elvis.bwf" file

The "elvis.bwf" file is loaded into a buffer named "Elvis before writing". That buffer is executed as a series of ex commands immediately before writing the entire contents of a buffer out over its original file.

```
if backup && !newfile
then {
 if os=="unix"
 then eval ! cp (filename) (filename).bak
 else eval ! copy (filename) (basename(filename)).bak >NUL
}
```

These lines copy the original version of the file to a "*.bak" file. Note that we implement separate Unix and non-Unix versions of the copy command here.

10.2.5 The "elvis.awf" file

The "elvis.awf" file is loaded into a buffer named "Elvis after writing". That buffer is executed as a series of ex commands immediately after writing the entire contents of a buffer out over its original file.

There is no default "elvis.awf" file, because I haven't found any need for one yet.

10.3 Recovery

If elvis ever dies an unnatural death, the session file will be left behind. This session file contains all of the changes you've made during your edit session, so you should be able to start a new elvis process on the old session file and recover all of your changes.

Only one elvis process at a time is allowed to use a given session file. To enforce this, when elvis starts up it sets an "in use" flag in the session file's header. Any later elvis process will test that flag, and refuse to use a session file which is already in use.

When elvis crashes, it leaves the "in use" flag set, even though the process that was using it has died. You must restart your edit session via "elvis -r". The -r flag tells elvis to ignore the "in use" flag. If you aren't using the default session file, then you'll need to add a "-f sessionfile" flag to tell elvis which session file it should recover from.

If you always use the default session file, and allow several old files to accumulate after crashes, then "elvis -r" will always recover from the lowest-numbered one. The command "elvis -r -Gquit" will tell you its name. If you prefer to recover form a different session file, you can either delete the lower-numbered session files, or use the "-f sessionfile" flag to make elvis use a different one.

When this new elvis process starts up, it will be displaying a new, empty buffer. **Don't panic!** Your edit buffers are still intact; they just don't happen to be displayed in the initial window.

After a crash, the session file might not be entirely self-consistent. Because of this, it is dangerous to edit the file using this session file. **You should save your old buffer to a file immediately, and then exit elvis.** To save your old buffer give elvis the command ":(*buffer*)w *filename*" where *buffer* is the name of your buffer (usually the same as the original file name) and *filename* is the name of a new file where you wish to store the text. Note that the buffer name should be in parentheses! And for safety's sake, you should not write the salvaged buffer out over the top of the original text file.

Under normal circumstances elvis automatically deletes the session file when it exits, but when recovering after a crash elvis is more cautious. It never deletes a recovered session file itself. After recovering your text and exiting elvis, you should manually delete the session file via "rm /var/tmp/elvis*.ses", or whatever the session file's name is. For DOS/Windows users, the command would be "DEL \TEMP\ELVIS*.SES".

If you can figure out how to reproduce the problem, please let me know! My email address is *kirkenda@cs.pdx.edu*

10.4 Other files

The following configuration files aren't necessarily related to initialization or sessions, but since we've discussed so many configuration files in this chapter already, we might as well finish it off.

***.man**
: These files are Unix-style "man pages" describing each of the programs. You can view them with elvis' "man" display mode, or you can print them via "troff -man ..." or the local equivalent.

elvis.ftp or ~/.netrc
: This file stores account names and passwords to be used when contacting certain FTP sites. It is described in Chapter 15: The Internet.

elvis*.html
: These files store the on-line interactive manual for elvis. When you use the :help command, elvis locates the necessary file and loads it. These files are written in HTML so you can also view/print them using a Web browser such as Netscape.

howto.html

This file contains a lot of "How To" discussions for various features. It is meant to be searched via the ":howto" alias defined in elvis.ali. Most of the discussions contain links into the manual, so it is important for this file to be located in the same directory as all of the elvis*.html files.

elvis.lat
This file contains a bunch of :digraph commands for setting up the digraph table for the Latin-1 symbol set. The default elvis.ini file interprets this file's contents automatically.

elvis.pc8
This file contains a bunch of :digraph commands for setting up the digraph table for the PC-8 symbol set (which corresponds to IBM Code Page 437). The default elvis.ini file interprets this file's contents automatically for MS-DOS, OS/2 and text-mode Win32.

elvis.ali
This contains an assortment of aliases. If your copy of elvis is configured to support aliases (and all versions are, except for MS-DOS) then this file will be automatically loaded via the elvis.ini script, each time you run elvis.

elvis.msg
This file stores a translation table, which allows you to customize elvis' messages. This file is described in the Messages chapter.

elvistrs.msg
This contains a rough list of nearly all of elvis' terse messages. You can use this as a resource when constructing an elvis.msg file. The idea here is that you'll copy a line from elvistrs.msg into elvis.msg, and then append a ":" and the new message text.

elvis.net
This tells elvis which sites can be accessed directly, and which can only be accessed via proxy servers. It is described in Chapter 15: The Internet.

elvis.ps
The PostScript printer drivers (lptype=ps or ps2) include this file's contents in the printer output. This file should contain PostScript code which defines the symbols ElvisN, ElvisB, and ElvisI as 12-point monospaced fonts to be used for normal text, bold text, and italic text, respectively. It also defines ElvisPage, ElvisLeftPage, and ElvisRightPage procedures for setting the size and position of a page's text on the paper. If this file doesn't exist or is unreadable, elvis will use the following definitions:

```
/ElvisN /Courier findfont 12 scalefont def
/ElvisB /Courier-Bold findfont 12 scalefont def
/ElvisI /Courier-Oblique findfont 12 scalefont def
/ElvisPage { 12 36 translate } def
```

```
/ElvisLeftPage { 12 750 translate -90 rotate 0.58 0.75 scale } def
/ElvisRightPage { newpath 12 394 moveto 576 0 rlineto stroke
                  12 366 translate -90 rotate 0.58 0.75 scale } def
```

elvis.syn
This contains descriptions of all languages supported by the syntax display mode. For a full description of this file, see the Language Specification section in the Display Modes chapter.

elvis.xbm
This stores a two-color Elvis icon, in the X-Windows XBM format.

elvis.xpm
This stores a four-color Elvis icon, in the X-Windows XPM format.

printdoc.bat
This contains a series of program invocations for printing all of the elvis documentation in the correct sequence. This file should be executable under MS-DOS, Win32, and Unix. *You must install elvis before this will work!*