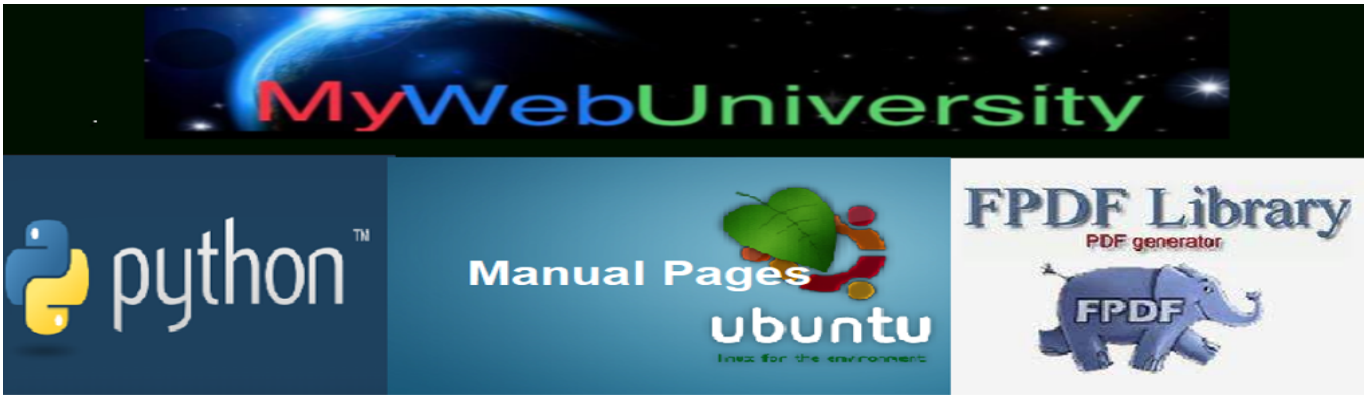

Rocky Enterprise Linux 9.2 Manual Pages on command 'mailcap.5'

$ man mailcap.5

MAILCAP(5) File Formats Manual MAILCAP(5)

NAME

mailcap - metamail capabilities file

DESCRIPTION

The mailcap file is read by the metamail program to determine how to display non-text at the local site.

The syntax of a mailcap file is quite simple, at least compared to termcap files. Any line that starts with "#" is a comment. Blank lines are ignored. Otherwise, each line defines a single mailcap entry for a single content type. Long lines may be continued by ending them with a backslash character, \.

Each individual mailcap entry consists of a content-type specification, a command to exe? cute, and (possibly) a set of optional "flag" values. For example, a very simple mailcap entry (which is actually a built-in default behavior for metamail) would look like this:

text/plain; cat %s

The optional flags can be used to specify additional information about the mail-handling command. For example:

text/plain; cat %s; copiousoutput

can be used to indicate that the output of the 'cat' command may be voluminous, requiring either a scrolling window, a pager, or some other appropriate coping mechanism.

The "type" field (text/plain, in the above example) is simply any legal content type name, as defined by informational RFC 1524. In practice, this is almost any string. It is the string that will be matched against the "Content-type" header (or the value passed in with -c) to decide if this is the mailcap entry that matches the current message. Addition?

ally, the type field may specify a subtype (e.g. "text/ISO-8859-1") or a wildcard to match all subtypes (e.g. "image/*").

The "command" field is any UNIX command ("cat %s" in the above example), and is used to specify the interpreter for the given type of message. It will be passed to the shell via the system(3) facility. Semicolons and backslashes within the command must be quoted with backslashes. If the command contains "%s", those two characters will be replaced by the name of a file that contains the body of the message. If it contains "%t", those two char? acters will be replaced by the content-type field, including the subtype, if any. (That is, if the content-type was "image/pbm; opt1=something-else", then "%t" would be replaced by "image/pbm".) If the command field contains "%{" followed by a parameter name and a closing "}", then all those characters will be replaced by the value of the named parame? ter, if any, from the Content-type header. Thus, in the previous example, "%{opt1}" will be replaced by "something-else". Finally, if the command contains "\%", those two charac? ters will be replaced by a single % character. (In fact, the backslash can be used to quote any character, including itself.)

If no "%s" appears in the command field, then instead of placing the message body in a temporary file, metamail will pass the body to the command on the standard input. This is helpful in saving /tmp file space, but can be problematic for window-oriented applications under some window systems such as MGR.

Two special codes can appear in the viewing command for objects of type multipart (any subtype). These are "%n" and "%F". %n will be replaced by the number of parts within the multipart object. %F will be replaced by a series of arguments, two for each part, giving first the content-type and then the name of the temporary file where the decoded part has been stored. In addition, for each file created by %F, a second file is created, with the same name followed by "H", which contains the header information for that body part. This will not be needed by most multipart handlers, but it is there if you ever need it.

The "notes=xxx" field is an uninterpreted string that is used to specify the name of the person who installed this entry in the mailcap file. (The "xxx" may be replaced by any text string.)

The "test=xxx" field is a command that is executed to determine whether or not the mailcap line actually applies. That is, if the content-type field matches the content-type on the message, but a "test=" field is present, then the test must succeed before the mailcap line is considered to "match" the message being viewed. The command may be any UNIX com?

mand, using the same syntax and the same %-escapes as for the viewing command, as de? scribed above. A command is considered to succeed if it exits with a zero exit status, and to fail otherwise.

The "print=xxx" field is a command that is executed to print the data instead of display it interactively. This behavior is usually a consequence of invoking metamail with the "-h" switch.

The "textualnewlines" field can be used in the rather obscure case where metamail's de? fault rules for treating newlines in base64-encoded data are unsatisfactory. By default, metamail will translate CRLF to the local newline character in decoded base64 output if the content-type is "text" (any subtype), but will not do so otherwise. A mailcap entry with a field of "textualnewlines=1" will force such translation for the specified content- type, while "textualnewlines=0" will guarantee that the translation does not take place even for textual content-types.

The "compose" field may be used to specify a program that can be used to compose a new body or body part in the given format. Its intended use is to support mail composing agents that support the composition of multiple types of mail using external composing agents. As with the view-command, the compose command will be executed after replacing certain escape sequences starting with "%". In particular, %s should be replaced by the name of a file to which the composed data is to be written by the specified composing pro? gram, thus allowing the calling program (e.g. metamail) to tell the called program where to store the composed data. If %s does not appear, then the composed data will be assumed to be written by the composing programs to standard output. The result of the composing program may be data that is NOT yet suitable for mail transport -- that is, a Content- Transfer-Encoding may still need to be applied to the data.

The "composetyped" field is similar to the "compose" field, but is to be used when the composing program needs to specify the Content-type header field to be applied to the com? posed data. The "compose" field is simpler, and is preferred for use with existing (non- mail-oriented) programs for composing data in a given format. The "composetyped" field is necessary when the Content-type information must include auxiliary parameters, and the composition program must then know enough about mail formats to produce output that in? cludes the mail type information, and to apply any necessary Content-Transfer-Encoding. Conceptually, "compose" specifies a program that simply outputs the specified type of data in its raw form, while "composetyped" specifies a program that outputs the data as a MIME

object, with all necessary Content-* headers already in place.

needsterminal

If this flag is given, the named interpreter needs to interact with the user on a terminal. In some environments (e.g. a window-oriented mail reader under X11) this will require the creation of a new terminal emulation window, while in most environments it will not. If the mailcap entry specifies "needsterminal" and metamail is not running on a terminal (as determined by isatty(3), the -x option, and the MM_NOTTTY environment variable) then metamail will try to run the command in a new terminal emulation window. Currently, metamail knows how to create new windows under the X11, SunTools, and WM window systems.

copiousoutput

This flag should be given whenever the interpreter is capable of producing more than a few lines of output on stdout, and does no interaction with the user. If the mailcap entry specifies copiousoutput, and pagination has been requested via the "-p" command, then the output of the command being executed will be piped through a pagination program ("more" by default, but this can be overridden with the METAMAIL_PAGER environment variable).

BUILT-IN CONTENT-TYPE SUPPORT

The metamail program has built-in support for a few key content-types. In particular, it supports the text type, the multipart and multipart/alternative type, and the mes? sage/rfc822 types. This support is incomplete for many subtypes -- for example, it only supports US-ASCII text in general. This kind of built-in support can be OVERRIDDEN by an entry in any mailcap file on the user's search path. Metamail also has rudimentary built-in support for types that are totally unrecognized -- i.e. for which no mailcap entry or built-in handler exists. For such unrecognized types, metamail will write a file with a "clean" copy of the data -- i.e. a copy in which all mail headers have been removed, and in which any 7-bit transport encoding has been decoded.

FILES

$HOME/.mailcap:/etc/mailcap:/usr/share/etc/mailcap:/usr/local/etc/mailcap -- default path for mailcap files.

SEE ALSO

run-mailcap(1), mailcap.order(5), update-mime(8)

RFC 1524 (<http://tools.ietf.org/html/rfc1524>)

COPYRIGHT

AUTHOR

Nathaniel S. Borenstein