PROMPTREMINDERS -- to be periodically reminded of things

File: <lispusers>PromptReminders.tty
Revised: Feb 21, 1983, and Apr 15, 1984, by JonL White

One may want to be periodically "reminded" of important things by a message which is aggressively "winked" and "flashed" in a prompt window (or on primary output for systems without bitmap display, such as Interlisp-10 and Interlisp/VAX). It will desist the wink/flash "hassling" only after it has been acknowledged by user response, or after a pre-set interval of "hassling" time has elapsed. Such is implemented using an Interlisp-D background process (or for the non-Interlisp-D systems, the facility of PROMPTCHARFORMS).

If the MESSAGE given for the reminder (see description of the function SETREMINDER below) is a listp, then when the reminder "goes off", that listp will be EVAL'd rather than any of the "winking", "flashing", or "hassling" mentioned above.

In Interlisp-D, the global variable REMINDERSTREAM holds the window (or, indeed, stream!) that the "winking/flashing" is to occur; if not set by the user, it will default to PROMPTWINDOW. After the "hassling" has completed, the window (if indeed REMINDERSTREAM holds a window) will be closed, depending on the value of CLOSEREMINDERSTREAMFLG.

REMINDERS is a filepkg type, so that they may be easily saved on files, and so that the general typed-definition facilities may be used. On any file which uses the REMINDERS filepkgcom, it is advisable to precede this command with a command
(FILES (SYSLOAD COMPILED FROM LISPUSERS) PROMPTREMINDERS)
since this package is not in the initial Lisp loadup. When initially defining a reminder, it is preferable for the user to call SETREMINDER rather than PUTDEF; but HASDEF is the accepted way to ask if some name currently defines a "reminder", and DELDEF is the accepted way to cancel an existing "reminder".

In the first example below, the user wants to be reminded every 30 minutes that he ought to be using MAKEFILE to save his work; in the second example, he merely wants to be told once, at precisely 4:00PM to call home; in the third, he merely checks every 10 minutes to see if there is a process called LISTFILES. Examples:

```
(SETREMINDER NIL (ITIMES 30 60) "Have you MADEFILE recently?")

(SETREMINDER 'WOOF NIL "Don't forget to inform wife of dinner plans."
             "8-Jan-83 4:00PM")

(SETREMINDER NIL 600
        '(PROGN (AND (FIND.PROCESS 'LISTFILES) (add FREQ 1))
                (add TOTAL 1)))
```

Functions:

(SETREMINDER NAME PERIOD MESSAGE INITIALDELAY EXPIRATION)

This will create and install a "reminder" with the name NAME (NIL given for a name will be replaced by a gensym), which will be executed every PERIOD number of seconds by winking the string MESSAGE into the prompt window; if MESSAGE is null, then NAME is winked; if MESSAGE is a listp, then it is EVAL'd and no "winking" takes place. "Winking" means alternately printing the message and clearing the window in which it was printed, at a rate designed to attract the eye's attention.

The first such execution will occur at PERIOD seconds after

the call to SETREMINDER unless INITIALDELAY is non-NIL, in which case that time will be used; a fixp value for INITIALDELAY is interpreted as an offset in seconds from the time of the call to SETREMINDER, and a stringp value is an absolute date/time string.

If PERIOD is null, then the reminder is to be run precisely once. If EXPIRATION is non-null, then a fixp means that that number of seconds after the first execution, the timer will be deleted; a stringp means a precise date/time at which to delete the timer.

Optional 6th and 7th arguments -- called REMINDINGDURATION and WINKINGDURATION -- permit one to vary the amount of time spent in one cycle of the wink/flash loop, and the amount of time spent winking before initiating a "flash". The attention-attracting action will continue for REMINDINGDURATION seconds (default: the value of the global variable DEFAULT.REMINDER.DURATION which is initialized to 60), or until the user types something on the keyboard; care is taken not to consume the typed character. Type-ahead does not release the winking. In case the user has become "drowsy", or otherwise fails to notice the winking, then every WINKINGDURATION seconds (default: the value of the global variable DEFAULT.REMINDER.WINKINGDURATION which is initialized to 10) during the "reminding", the whole display videocolor will be wagged back and forth a few times, which effects a most obnoxious stimulus (for non-bitmap systems, this just types some <bell>'s).

Returns the name (note above when NIL is supplied for the name).

(ACTIVEREMINDERNAMES)
No arguments; self-explanatory.

(REMINDER.NEXTREMINDDATE NAME)
Returns the time (in GDATE format) at which the next reminding from the named reminder will occur; NIL if NAME isn't a REMINDERS.
(REMINDER.NEXTREMINDDATE NAME Date/Time.string)
Sets the time at which the reminder is next to be executed.

(REMINDER.EXPIRATIONDATE NAME)
Returns the time (in GDATE format) at which the reminder will be automatically deleted.
(REMINDER.EXPIRATIONDATE NAME Date/Time.string)
Sets the expiration time.

(INSPECTREMINDER NAME)
In Interlisp-D, this will call INSPECT on the definition of the named reminder; in other systems, it merely calls SHOWDEF.

Function in Interlisp-D only:

(UNTILKEYDOWNP FN INTERVAL.SECS DURATION.SECS
subCycleDuration.secs subCycleFN)
For a period of up to DURATION.SECS seconds, the function FN will be "run" (i.e., applied to no arguments) every INTERVAL.SECS seconds; both of these time durations may be floatp's, and hence specify fractional parts of a second. The process is stopped whenever there is any change in the state of the keyboard; except that CTRL, LOCK, LSHIFT, RSHIFT, and LEFT and RIGHT (tow of the three mouse buttons) don't count.

The function FN, of course, may save "state" in order to do different things cyclically on the various "runnings". But a common structure is to have a major cycle and a minor cycle for which some activity is to be performed. The two optional arguments, subCycleDuration.secs and subCycleFN, allow for this second function to be "run" at the end of the subCycle duration

time.

For example, INTERVAL.SECS may be 0.5 in order to cause a "winking" of a message in the PROMPTWINDOW every second or so, and subCycleDuration.secs may be 10.0 in order to cause a "flashing" of the whole screen every 10 seconds or so. In this case, the major cycle is the "fast rate winking" every second, whereas the minor cycle is the "slow rate flashing" every 10 secs.