```
(RPAQQ SEDIT-PATCHCOMS ((DECLARE\: EVAL@COMPILE DONTCOPY (FILES (SOURCE)
                                                                SEDIT-DECLS))
                        (FNS |\\\\closefn| |\\\\setup.context| |\\\\complete| |\\\\sedit|
                              |\\\\handle.completion|)))

(DECLARE\: EVAL@COMPILE DONTCOPY

(FILESLOAD (SOURCE)
       SEDIT-DECLS)
)

(DEFINEQ

(|\\\\closefn|
   (LAMBDA (|window|)                                               ; Edited  6-Jun-88 14:47 by raf

;;; to be called by the window system when SEdit windows are closed.  if there's a process behind them, wake it up (it should notice that the window is
;;; gone and do the right stuff) otherwise just trash the context.  grab the lock here, because it wasn't yet grabbed by the buttoneventfn.

       (LET* ((|context| (WINDOWPROP |window| '|EditContext|))
              (|lock| (|fetch| |ContextLock| |of| |context|)))
             (|if| |context|
                 |then|                                            ; this is a live SEdit
                     (|if| (OR (EQ :REFETCH |lock|)
                               (AND (OBTAIN.MONITORLOCK |lock| T)
                                    (TRUE (RELEASE.MONITORLOCK |lock|))))
                         |then|                                    ; release lock before waking sedit
                             (|if| (WINDOWPROP |window| 'PROCESS)
                                 |then| (|\\\\awake.command.process| |context| NIL)
                                 |else| ;; this should never happen, now, because completion is unwind protected in \\sedit, and so
                                        ;; disintegration will have happened under completion.  if by chance we screwed up, then the context
                                        ;; will still be there, so go ahead and waste it.  IT CAN HAPPEN IF SOMEBODY RETFROMs \\sedit.

                                        (|\\\\disintegrate.context| |context|))
                         |else| (|printout| (|\\\\get.prompt.window| |context|)
                                    T "Can't close.  SEdit is busy")
                             'DON\'T)))))

(|\\\\setup.context|
   (LAMBDA (|context|)                                             ; Edited  6-Apr-88 14:20 by bane

;;; confirm that this context is setup.  that means either setting up a new context or verifying the structure in an old one.

       (|if| (NULL (|fetch| |ContextLock| |of| |context|))
           |then| ;; this is a new sedit.  setup its profile, and then the context itself

               (|\\\\setup.profile| (|fetch| |Profile| |of| |context|)
                     |context|)
               (|\\\\setup.new.context| |context|)
           |elseif| (EQ (|fetch| |ContextLock| |of| |context|)
                        :REFETCH)
           |then| ;; this is a context that was shrunk, and thus we need to refetch when we verify the structure to see if there were any changes made
                  ;; while we were asleep.  Also must replace the ContextLock

               (|replace| |ContextLock| |of| |context| |with| (CREATE.MONITORLOCK (CONCAT |\\\\name|
                                                                                          (|fetch| |IconTitle|
                                                                                             |of| |context|))))
               (|\\\\verify.structure| |context|)
           |else| ;; This must be an SEdit getting restarted (eg HardReset).  Do a verify without refetching

               (|\\\\verify.structure| |context| |nil| T))))

(|\\\\complete|
   (LAMBDA (|context| |charcode| |active?|)                        ; Edited  6-Apr-88 14:20 by bane

;;; entry point into completing an sedit.  called when window is closed or process otherwise dies.  active? is T when sedit is to remain active after
;;; completion, like from the 'complete' command.  must grab lock because can be called outside of the command loop (perhaps even as programmer
;;; interface?).

       (WITH.MONITOR (|fetch| |ContextLock| |of| |context|)
             (|\\\\close.open.node| |context|)
             (|if| (AND |active?| (EQMEMB :CLOSE-ON-COMPLETION (|fetch| |EditOptions| |of| |context|)))
```

```
              |then|  ;; if we're supposed to close on completion, but his complete command says we're trying to stay active, then just close the
                      ;; window and return.  sedit will notice the window has closed and it will complete normally on the way out.

                          (CLOSEW (|fetch| |DisplayWindow| |of| |context|))
          |else| (|\\\\handle.completion| |context|)
                 (|if| |active?|
                      |then|  ;; if still open then verify structure to get edit date fix.  this is a hack.  the markaschangedfn doesn't succeed in verifying
                              ;; because sedit isn't under \\getkey since this command is running, and thus cannot be woken up.  if edit dates were
                              ;; external this could be removed.

                                (|\\\\\verify.structure| |context| NIL T)

                              ;; if we're remaining active, eg ^X, give the tty away.  no point in doing this if we're closing, because the process dying
                              ;; will give it away, and may not have the tty when closing either.

                                (TTY.PROCESS T)
                      |else| (LET ((|window| (|fetch| |DisplayWindow| |of| |context|)))
                                (WINDOWPROP |window| 'PROCESS NIL)
                                (|if| (OPENWP (WINDOWPROP |window| 'ICONWINDOW))
                                    |then|  ;; window was shrunk.  just let the region manager know, and mark the context so we know to refetch
                                            ;; when we get restarted

                                              (SEDIT.SAVE.WINDOW.REGION |context| :SHRINK)
                                              (|replace| |ContextLock| |of| |context| |with| :REFETCH)
                                    |else| ;; window wasn't shrunk, so context is now dead.

                                              (SEDIT.SAVE.WINDOW.REGION |context| :CLOSE)
                                              (|\\\\\disintegrate.context| |context|)))))
          ;; can be called as a command, so must return T

          T)))


(|\\\\\sedit|
  (LAMBDA (|context|)                                                          ; Edited  6-Apr-88 15:04 by bane

;;;; this is the driver loop.  read and process characters until the window is closed, and then exit.  The commonlisp printer flgs for atomic printing are
;;;; rebound specially here, so global changes won't affect existing contexts.  First check to see if the system is trying to restart SEdit on a dead context,
;;;; and punt if so.

    (|if| (NEQ (|fetch| |ContextLock| |of| |context|)
              '|Dead|)
        |then|
        ;; this SEdit is okay, or new

        (XCL:WITH-PROFILE
          (|fetch| |Profile| |of| |context|)
          (|\\\\setup.context| |context|)
          (|\\\\setup.window.and.process| |context|)
          (LET*
            ((|lock| (|fetch| |ContextLock| |of| |context|))
             (|default.char.handler| (|fetch| |DefaultCharHandler| |of| (|fetch| |Environment| |of| |context|)))
             (|command.table| (|fetch| |CommandTable| |of| (|fetch| |Environment| |of| |context|)))
             (|window| (|fetch| |DisplayWindow| |of| |context|))
             (|promptwindow| (GETPROMPTWINDOW |window|))
             |charcode| |command| |this.char.escaped|)
            (DECLARE (SPECVARS |this.char.escaped|))
            (|while| (OPENWP |window|)
                |do|

                ;; if something funny happens (e.g.  the window is closed) \\awake.command.process will cause \\getkey to return NIL.  If a menu item
                ;; is selected, \\getkey will return the command form as a list.

                (|if| (NULL (ERSETQ                                          ; catch errors at top of loop
                             (SETQ |charcode| (|\\\\\getkey| |context|))
                             (WITH.MONITOR |lock|
                                (|if| |charcode|
                                    |then| (\\CARET.DOWN |window|)
                                           (|\\\\\selection.down| |context|)
                                           (|if| (LISTP |charcode|)
                                               |then| ;; a command generated externally.  the variable command gets used later, so it
                                                      ;; must be set here

                                                        (SETQ |command| |charcode|)
                                                        (SETQ |this.char.escaped| NIL)
                                                        (|printout| |promptwindow| T)
                                                        (APPLY (CAR |command|)
                                                               (LIST* |context| NIL (CDR |command|)))
                                               |elseif| |this.char.escaped|
                                                   |then|                    ; an escaped char
                                                        (APPLY* |default.char.handler| |context| |charcode|)
                                                        (SETQ |this.char.escaped| NIL)
                                               |elseif| (AND (OR (SETQ |command| (|\\\\\lookup.command| |charcode|
                                                                                        |command.table|))
                                                                 (SETQ |command| (|\\\\\lookup.command| (GETSYNTAX
                                                                                                         |charcode|)
                                                                                        |command.table|)))
                                                             (APPLY (CAR |command|)
                                                                    (LIST* |context| |charcode| (CDR |command|))))
```

```
                                                 |then|  ;; this is a valid command or syntax char, and it has already been handled

                                          |else|  ;; none of the above, or else the command didn't want to run.  treat as normal input

                                             (APPLY* |default.char.handler| |context| |charcode|))
                                  (|if| (OR (NOT |command|)
                                            (NOT (FMEMB (CAR |command|)
                                                        '(|\\\\undo|  |\\\\redo|))))
                                      |then| (|replace| |UndoUndoList| |of| |context| |with| NIL)))
                            ;; unless the user is typing too fast to keep up, fix up the window

                                  (|if| (AND (OPENWP |window|)
                                             (NOT (\\SYSBUFP)))
                                      |then| (|\\\\update| |context|)))))
                    |then|  ;; on catching of errors, re-update to capture what was undone to run the command, like the current selection

                              (|\\\\update| |context| T)))
              ;; exit the loop after the window is closed (or shrunk), and thus it's complete time.

              (|\\\\complete| |context|))))))))
```

```
(|\\\\handle.completion|
   (LAMBDA (|context|)                                                            ; Edited  5-Apr-88 16:15 by bane
      (NOTIFY.EVENT (|fetch| |CompletionEvent| |of| |context|))
      (|replace| |AtomStarted| |of| |context| |with| NIL)
      (|replace| |AtomStartedUndoPointer| |of| |context| |with| NIL)
      (|if| (|fetch| |ChangedStructure?| |of| |context|)
          |then| (LET ((|fn| (|fetch| |CompletionFn| |of| |context|)))
                    (|if| |fn|
                        |then| (APPLY (|if| (LISTP |fn|)
                                          |then| (CAR |fn|)
                                          |else| |fn|)
                                   (LIST* |context| (|fetch| |Structure| |of| (|\\\\subnode| 1 (|fetch| |Root|
                                                                                                   |of| |context|)))
                                      (CDR (LISTP |fn|))))))
                 (|replace| |ChangedStructure?| |of| |context| |with| NIL))
      (|replace| |UndoList| |of| |context| |with| NIL)
      (|replace| |UndoUndoList| |of| |context| |with| NIL)))

)

(PUTPROPS SEDIT-PATCH COPYRIGHT ("Venue & Xerox Corporation" 1988 1990))
```

FUNCTION INDEX