APPENDIX C

EXAMPLE
AN UNCONVENTIONAL TEXT EDITOR

```
/*************************************************************
 *      EXAMPLE 4.5: AN UNCONVENTIONAL TEXT EDITOR           *
 *************************************************************/

autocalc = 0;

proc _autocalc: autocalc
{
        if (autocalc) {
                while (calc_list#) {
                        *calc_list[1];
                        shift calc_list;
                }
        }
}


                /* --- constants --- */
TopOfFile is "top of file";
EndOfFile is "end of file";
MAXCOL is 100;
EOF = -1;

                /* --- make the ruler --- */
proc repeatchar         /* a string of repeated char */
        /* $1 = no. of char; $2 = char */
{
        auto s, i;

        s = substr("", 1, $1);
        if ($# > 1)
                for (i = 1; i <= $1; i++) {
                        s[i] = $2;
                }
        return s;
}
RULER = repeatchar(MAXCOL, '.');
for(i = 5 ; i <= MAXCOL; i += 10) RULER[i] = ':';
for(i = 10; i <= MAXCOL; i += 10) RULER[i] = char(i / 10 % 10 + '0');
CUR_RULER is substr(RULER, win_left, win_right);

                /* --- cursor position --- */
line = 1;
col  = 1;
text = [""];
CUR_LINE is (line > text#) ? "" : text[line];
CUR_CHAR is (col > CUR_LINE#) ? ' ' : CUR_LINE[col];

                /* --- window set up --- */
WIN_WIDTH  = 60;
WIN_HEIGHT = 20;
win_top = 1;
win_bottom is win_top + WIN_HEIGHT - 1;
win_left = 1;
win_right is win_left + WIN_WIDTH - 1;
win_offset_x is 0;
```

```
win_offset_y is 1;

        /* --- editing insert_mode --- */
insert_mode = 1;    /* default insert_mode = ON */

        /* --- utility functions --- */
func min { return $1 < $2 ? $1 : $2 ; }
func max { return $1 > $2 ? $1 : $2 ; }
func isprint    /* Is it a printable char ? */
/* char $1 */
{
        return ($1 >= ' ') && ($1 < 127);
}

        /* --- screen utilities --- */
proc clear    { wclear(stdscr); }
proc move     { wmove(stdscr, $1, $2); }
func getch    { return char(wgetch(stdscr)); }
proc addch    { waddch(stdscr, $1); }
proc insch    { winsch(stdscr, $1); }
proc insertln { winsertln(stdscr); }
proc delch    { wdelch(stdscr); }
proc addstr   { waddstr(stdscr, $1); }
proc clrtoeol { wclrtoeol(stdscr); }
proc refresh  { wrefresh(stdscr); }

proc at      /* at position relative to window offset */
        /* $1 = line; $2 = col */
{
        move(
                $1 - win_top  + win_offset_y,
                $2 - win_left + win_offset_x
        );
}
proc update_scrn        /* update the change on screen */
{
        at(line, col);
        refresh();
}

                /* --- cursor movement --- */
func up                 /* cursor up */
{
        if (line > 1)
                line--;
        else
                message = TopOfFile;
}

func down               /* cursor down */
{
        if (line < text#)
                line++;
        else
                message = EndOfFile;
}
```

```
func left               /* cursor left */
{
        if (col > 1)
                col--;
        else
                message = "left margin";
}

func right              /* cursor right */
{
        if (col < MAXCOL)
                col++;
        else
                message = "right margin";
}

func vscroll            /* scroll vertically */
{
        auto    j;

        if (line > win_bottom || line < win_top) {
                j = max(1, line - WIN_HEIGHT/2);
                if (j + WIN_HEIGHT - 1 > text# && text# >= WIN_HEIGHT)
                        j = text# - WIN_HEIGHT + 1;
                win_top = j;
        }
}

proc hscroll            /* scroll horizontally */
{
        if (col > win_right || col < win_left)
                win_left = min(max(1, col-WIN_WIDTH/2), MAXCOL-WIN_WIDTH+1);
}

proc tab                /* tabulate */
{
        do {
                if (col == MAXCOL) {
                        message = "right margin";
                        return;
                }
                if (!insert_mode || CUR_CHAR == ' ')
                        col++;
                else
                        insert_char(' ');
        } while (col % 8 != 1);
}

proc print_line         /* print a line */
        /* $1 = line to be printed */
{
        auto    i;

        at($1, win_left);
        if ($1 <= text#)
```

```
                addstr(substr(text[$1], win_left, win_right));
        else
                clrtoeol();
}

proc renew_scrn         /* renew the entire edit window */
{
        auto    L;

        for(L = win_top; L <= win_bottom; L++)
                print_line(L);
}

proc print_eos          /* print to end of screen */
{
        auto    L;

        for(L = line; L <= win_bottom; L++)
                print_line(L);
        update_scrn();
}

proc insert_newline     /* insert a newline */
{
        insert text, line + 1, substr(CUR_LINE, col, CUR_LINE#);
        text[line] = substr(CUR_LINE, 1, col - 1);
        print_eos();
}

proc cr                 /* carriage return */
{
        if (insert_mode) {
                while (line > text#)
                        append text, "";
                insert_newline();
        }
        col = 1;
        down();
}

proc insert_char        /* insert a char under cursor */
        /* $1 = char */
{
        while (line > text#)
                append text, "";

        if (CUR_LINE# >= MAXCOL) {
                message = "line too long";
                return;
        }

        text[line] = substr(CUR_LINE, 1, col - 1)
                  // $1
                  // substr(CUR_LINE, col, CUR_LINE#);
        insch($1);
        at(line, win_right + 1);
```

```
        addch(' ');
        right();
}

proc replace_char       /* replace a char under cursor */
        /* $1 = char */
{
        if (col <= CUR_LINE#) {
                text[line][col] = $1;
                addch($1);
                right();
        } else
                insert_char($1);
}

proc delete_line        /* delete a line */
        /* $1 = line */
{
        if ($1 <= text#)
                delete text, $1;
        else
                message = EndOfFile;

        /*---- inside the window ? ----*/
        if (win_top <= $1 && $1 <= win_bottom)
                print_eos();
}

proc delete_current_line  /* delete the current line */
{
        delete_line(line);
}

/* delete the char on the left of the cursor */
proc delete_char
{
        auto    c, l;
        if (col == 1) {         /* at the begin of line */
                if (line > 1) {
                        if (CUR_LINE# + text[line-1]# > MAXCOL) {
                                message = "line too long to join";
                                return;
                        }
                        up();
                        col = CUR_LINE# + 1;
                        text[line] = CUR_LINE // text[line+1];
                        delete_line(line + 1);
                        if (line > text#) up();
                } else
                        message = TopOfFile;
        } else if (col <= CUR_LINE# + 1) {
                text[line] = substr(CUR_LINE, 1, col - 2)
                          // substr(CUR_LINE, col, CUR_LINE#);
                left();
                print_line(line);
                update_scrn();
```

```
        } else {        /* longer than the line */
                left();
        }
}

proc print_insert_mode  /* print the editing mode */
{
        move(22, 33);
        addstr(insert_mode ? " insert" : "replace" );
}

proc print_line_number  /* print the line number */
{
        move(22, 5);
        addstr(substr(str(line), 1, 4));
}

proc print_message      /* print the message */
{
        if (message == @)
                return;
        move(23, 0);
        addstr(message);
        addch('\n');
}

proc print_ruler        /* print the ruler */
{
        auto    j;

        move(win_offset_y - 1, win_offset_x);
        addstr(CUR_RULER);
        move(win_offset_y + WIN_HEIGHT, win_offset_x);
        addstr(CUR_RULER);
}

proc cbreak     /* set cbreak mode */
{
/*
        raw();
        I don't know how to use raw mode.
        After calling this function,
        the adm5e terminal can't get some chars,
        but will work on tvi and sun terminals.
        It does well in cbreak mode (i.e crmode).
        However in cbreak mode can't get some control chars (e.g. ^O, ^V).
*/
        crmode();
        noecho();
        nonl();
}

proc reset
{
        noraw();
        nocrmode();
```

```
        nl();
        echo();
        endwin();
}

func edit
{
        auto    c;

        cbreak();
        clear();
        print_ruler();
        move(22,0);
        addstr("line=");
        print_line_number();
        print_message();
        print_insert_mode();
        renew_scrn();
        update_scrn();
        while ((c = fgetc(stdin)) != EOF) {
                if (message != "")
                        message = "";
                if (c < ' ') {
                        switch (c + 'A' - 1) {
                        case 'C':       /* ^C */
                                reset();
                                move(23, 0);
                                refresh();
                                exit();
                        case 'D':       /* ^D */
                                reset();
                                move(23, 0);
                                refresh();
                                return;
                        case 'E':
                                col = text[line]# + 1;
                                break;
                        case 'H':         /* ^H */
                                left();
                                break;
                        case 'J':       /* ^J */
                        case 'V':       /* ^V down arrow */
                                down();
                                break;
                        case 'K':         /* ^K */
                                up();
                                break;
                        case 'L':         /* ^L */
                                right();
                                break;
                        case 'M':         /* CR */
                                cr();
                                break;
                        case 'N':
                                if (line == win_bottom)
                                        down();
```

```
                                else
                                        line = min(text#, win_bottom);
                                break;
                        case 'O':
                                if (line == win_top)
                                        up();
                                else
                                        line = win_top;
                                break;
                        case 'I':
                                tab();
                                break;
                        case 'T':        /* ^T */
                                insert_mode = ! insert_mode;
                                break;
                        case 'X':       /* DELETE LINE */
                                delete_current_line();
                                col = 1;
                                if (line > text#)
                                        up();
                                break;
                        }
                } else if (c == 127) {  /* DEL */
                        delete_char();
                } else if (isprint(c)) {
                        c = char(c);
                        if (insert_mode)
                                insert_char(c);
                        else
                                replace_char(c);
                }
        }
}

proc readfile   /* filename, text */
{
        auto    s, f;
        auto    count;

        count = 0;
        f = fopen($1, "r");
        if (f == 0) {
                writeln("can't read file ", $1);
                exit();
        }
        s = repeatchar(256);
        while (fgets(s, 256, f) != 0) {
                if (s[s#] == '\n')
                        s[s#] = '\0';
                append *$2, s;
                count++;
        }
        fclose(f);
        writeln("file ", $1, ": ", count, " lines read");
}
```

```
/*------------ DEFINE CONNECTIONS --------------*/
col ~> [hscroll, update_scrn];
print_line_number ~> [update_scrn];
line ~> [vscroll, print_line_number];
win_top ~> [renew_scrn, update_scrn];
CUR_RULER ~> [print_ruler, renew_scrn, update_scrn];
insert_mode ~> [print_insert_mode, update_scrn];
message ~> [print_message, update_scrn];

/*------------ READ A FILE ---------------------*/
filename = repeatchar(256);
write("filename? ");
gets(filename);

/*--------------------------------------------*/
initscr();
readfile(filename, &text);
text=[];
autocalc = 1;
edit();
```