My Lee

SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

COMPUTING DIVISION

DISTRIBUTED INTERACTIVE COMPUTING NOTE 767

PERQ UNIX Implementation Note $ 56
Evaluation of porting DED to PERQ.

Issued by
James Collis

14 December 1982

DISTRIBUTION:	R W Witty
	K Robinson
	C Prosser
	A S Williams
	L O Ford
	T Watson
	E V C Fielding
	J C Malone
	P J Smith
	A J Kinroy
	J M Loveluck
	C P Wadsworth
	James Collis
	M Ritchie ICL Dalkeith
	P Palmer ICL Dalkeith
	RL Support/PERQ/Unix Implementation Notes File

1. Introduction

This report summarizes the difficulties in moving DED to the PERQ, looks at the simplest way of enhancing DED to utilise the tablet, and possible further extensions. It finally evaluates the DED screen editor as it is.

2. Summary

The DED code appears to be fairly machine independent, and would take about three man weeks to set up on the PERQ. To set it up with primitive extensions to allow movement via the puck, would extend the time to 5 man weeks. To give the puck more comprehensive capabilities, offering fluent movement throughout the file, (only previously available from the keyboard), and new text selection based commands, would take another man month and more, thus requiring at least two man months in total. Even then, it is doubtful whether DED would be either powerful or attractive enough to be used.

Note: Time estimates include testing, documentation & contingencies.

3. Porting Considerations

For DED to work on the PERQ it would be necessary to alter the 'site' and 'driver' entries at the beginning of the make file. The 'site' file should be straightforward to set up, using the default terminal settings as given in 'def.template', and hardware support for clearscreen and reset cursor to top left hand corner. There are plenty of examples of other terminal structures, showing what can be configured. Allowing time for familiarisation with DED, these files will take about two man weeks to complete.

Some machine dependent instructions are used in the code. For example, in tmpfile.c function addline right shifts a signed integer, which the PDP fills with the sign, but as 'tmp offset' ought not to go negative, this should not be a problem. On the PERQ running Accent UNIX, function pointers cannot be assigned to integers and back, because they are 5 words wide, (this is not a problem with microcode UNIX). I have traced a selection of these and the coding has consistently stored and passed them correctly. Though really they should all be checked thoroughly to avoid possible problems later. Lint warns about some operations with pointers and integers that should be watched with care.

The routines written in PDP 11 assembler that are distributed with DED, do not appear to be called when running on version 7 UNIX, so again should not be a problem.

The storage allocator for the screen in 'alloc.c' relies on 'int' having more restrictive alignment properties than '(char *)', and if this is not the case, will require alteration.

Estimated time for resolving potential bugs is one man week, giving a total time of three man weeks for DED to be working correctly.

Although the code appears fairly machine independent, it is worth noting that with several 'goto' statements, insufficient helpful comments, some sections of complex code, combined with the size of the program (10,000 lines), it could prove very time consuming to find any bugs should they appear.

4. Simple Method for getting DED to Utilise the Tablet

An easy way to allow cursor movement using the puck (mouse or pen), would be to create a new command to move, and then read from the GPIB (Tablet). It would also be simpler to restrict this command to when the editor is in input mode, which is also when it would be most useful.

To set up a new command in 'ictab', (input mode character table) - (see "chars.def").

eg. change ^P 020 to read PUCK instead of IGN (ignore).

Then in ded.h

```
    #define PUCK 20   /* movement via puck */
```

add comment to char.h

```
    /* ^P (020) is PUCK */
```

"mainloop.c" can now be altered for the extra case :

```
eg. case PUCK:

           {    /* something like :- */

                *pcur = READGPIB();
                /* check to see if new cursor
                   position is in the text area */
                prow = pcur -> row;
                pcol = pcur -> col;
                if (( prow < toprow ) ||
                   ( prow > bottomrow ) ||
                   ( pcol > rmargin(prow)) ||
                   ( pcol < leftcol ))
                   complain;
                else /* new position is in
                                text area      */
                {
                   vrow = prow;
                   vcol = pcol;
                }
                break;
                /* control now passes to the end
                   of the switch, where the new
                   position will be fixed at
                   vrow, vcol.                */
           }
```

For the above code to work, the procedure READGPIB must return a pointer to a structure of type cursor.

```
/* something like :- */
struct CURSOR *READGPIB()
{   display a pointer different to cursor ;
    while ( no buttons pressed )
        track the puck around the screen ;
    remove pointer from the screen ;
    return ( a pointer to a structure of
             type cursor, containing the
             last position of the puck
             as row and column. );
}
```

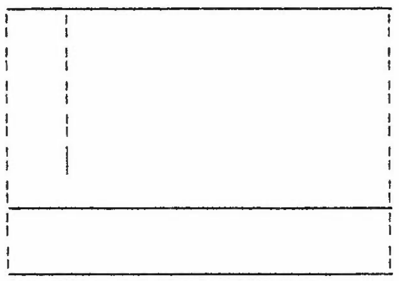
This would provide an awkward feel to using the puck, and will corrupt the dlog file unless the control P is not stored in the dlog file, (see dlogout and ttyin at end of tty.c). The dlog file protects against a DED or UNIX crash during an editing session, it works by storing each typed character along with other useful information, which can be replayed later when DED or UNIX is revived.

Another method of utilising the puck would be to read from the tablet while the editor is waiting for some character input during 'ttyin'. For this, extra code would be required for updating the position in the file from the cursor position, (which DED presently does the opposite way around). It would be necessary to know where ttyin was called from. For example, an interactive search and replace command requires character input, and the cursor position must not be allowed to change in this case. Finally, because 'ttyin' is used when in edit or input mode, cursor movement could be allowed along the command line as well as in the text region (see below), provided the movement is validated depending on the current mode.

This would be harder to implement, but by not requiring the ^P and the button pressing to start and finish the cursor movement, it would provide for much smoother more natural usage of the puck.

Time to implement simple puck movement is a further two weeks. Total time spent so far is now 5 man weeks, possibly more if the correctness of the dlog file is to be maintained, (because no matter where the tablet is read from, if the cursor position is altered, the dlog file will be corrupted unless the equivalent DED commands to move the cursor are calculated and stored in the dlog file).

The DED display is split into three areas, a command line at the bottom of the screen, with the area above it divided into a left hand margin and the largest area for displaying the text.



To extend use of the puck still further, it may be possible to interpret button commands differently, according to the region of the screen in which the cursor appears. For example :

In the left hand margin

1 Scroll up continuously until released
2 Scroll down continuously until released
3 Display next screenful of text
4 Display previous screenful of text

In the text region

No buttons required for altering the cursor position by tracing the puck around the tablet.

1 Select this character
2 Select this word
3 Select this line
4 Extend the selection to here

On the command line

1 Movement to approximate region of file using the command line as a horizontal representation of the length of the file.

For the operations in the left hand margin and the command line, either new procedures to do the work, or algorithms to translate the puck operation into the equivalent DED commands, would be required. These operations would then allow easy movement to any particular place in a file using the puck.

For the operations in the text region, extra routines would be needed to display the effect, say by underlining the selected text. From there it would be complicated to alter DED commands to make use of the information provided, because they are heavily based on a range and regular expression type format. So either extra routines are required for new instructions, such as moving or deleting the selected text, or, the operations should merely be alternative ways of calling other DED commands, such as delete to end or beginning of line.

To make these interpretations more obvious to the user, the cursor should appear differently in each region. When DED shifts into edit mode the cursor should again appear differently on the command line, where it may be moved by the puck. Commands such as scrolling cannot be allowed in edit mode because they would alter the current line in the text region which was set when input mode was left. The same restriction allready exists for similar DED keyboard commands.

For these more major enhancements, it would take at least a further man month, bringing the total to a minimum of two man months.

5. DED as a Screen Editor.

DED is not a popular editor at the Rutherford Laboratory, in general people prefer to use Pepper or the Pos editor on the PERQ, as opposed to DED on the PDP. The user interface to DED is unpleasant in comparison to Pepper. The mnemonics for DED are as bad if not worse than the emacs style commands for Pepper, but if DED were set up to use the puck , then as with Pepper some of the commands-such as :-

^U or ^H	: move left a character
^Q	: move right one character
^A	: move to start of next word
^B	: move to start of current/previous word
"	: go to next screenful of text

would be superfluous. The display only shows the text, a margin and a command line, whereas Pepper displays current information on all insert, search, replace and kill buffers , and has a help file available containing all the instructions.

DED only allows one file to edited at a time, and moving text between files is limited to appending a line range to another file. With Pepper up to nine files may be edited at a time, and it is easy to transfer text between files to specific places, with visual feedback. Both these editors have similar commands for deleting, searching and replacing text, and DED has powerful but complex regular expression matching capabilities.

Even if DED were altered to to use the puck as detailed, it is unlikely to be any more desirable to the user than an emacs style editor like Pepper.