[illegible] DIC file.

SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

COMPUTING DIVISION

DISTRIBUTED INTERACTIVE COMPUTING NOTE 900

A Tour through Troff

Issued by
Alan J Kinroy

6 October 1983

DISTRIBUTION:	A J Kinroy
	E V C Fielding
	R W Witty
	D R Gibson
	C P Wadsworth
	S T Frost
	H K F Yeung
	P J Smith
	C Prosser

As a result of incorrect output on both PNX display and VAX fr, and because the 8-bit codes generated by the troffs on these machines differed from the 8-bit codes generated by PDP11 troff I assumed there was a common fault in the troff implementation on machines with 32-bit integers.This document summarises how I attempted to find the fault by tracing through some of the (uncommented) code in troff.

1. Troff output & Code Tracing

The output codes from each troff implementation for a standard sample input file were generated on each of the three machines.PNX output and VAX output were identical with the exception of pad characters.Both differed from PDP11 output in their horizontal motion codes (octal 0200 - 0377, see TROFF(5)).

To start tracing where these codes were output I examined the routine ptesc() in file t10.c. With the insertion of suitable diagnostics I found the value of variable esc was incorrect and consequently so was the output code. (It is not clear why the procedure call oput(~i) is commented out as the two lines of code which follow it are textually identical to the code in the procedure body.)

Next I inserted diagnostics to find out where the escapement value esc was being incorrectly set and I discovered it was in the routine ptout(i) in t10.c. Tracing through ptout I found this was due to ds, de, temp, esct and inith all having incorrect values because the values in the integer array oline were incorrect.(At least the integer values were different e.g. -32744 PDP 32792 PERQ).As the only place oline has things added to it is at the start of the routine ptout and it adds its integer argument if the bottom byte is not an newline character I then had to trace where ptout was called.

Ptout is called from pchar1 in n2.c which in turn is called from pchar in n2.c and newline in n7.c. Further investigation revealed the error was whatever was calling pchar with the 16th (MOT - motion) bit set.

Tracing this back further led into n7.c and the routine horiz(i) which calls pchar(makem(i)). Examining makem(i) in t6.c showed why there were negative integers stored in oline. Makem does a bitwise OR of its argument with MOT which sets the 16th bit which is the sign bit on a PDP11 making the number look negative.Comparing some of the bit patterns in oline on the different machines showed that they matched.

I still had a problem as I had discovered the first call to horiz passed the wrong argument value.The call that was wrong was horiz(un) in tbreak in n7.c. I discovered that un had an incorrect value in nofill() in n7.c. Further diagnostics indicated that nel was being incorrectly set in storeline(c,w) as the result of a call to width(c).

Examining width(c) in t6.c showed that for arguments with the MOT (16th) bit on only the bottom 13 bits were returned as the result.For other arguments a call to getcw (get character width?) could arise. Getcw looks up a character array by using pointers & indirection and does a bitwise AND to get the bottom 8-bits

```
      k = *(p+i) & BMASK ;
```

where p is char*, i and k are int, BMASK is 0377. Printing out the values returned on PDP & PERQ showed that they were quite different even allowing for signed characters on a PDP and unsigned on a PERQ. Initially I thought these data values were read into the array in routine casefp in t6.c but this only happens in an explicit .fp command is used, otherwise the widths for the four default fonts Roman, Italic, Bold, Special are compiled in with troff in tab3.o.

The routine casefp() has to skip over header information in the semi-compiled stripped files which form the width tables.To ensure this is done correctly it should do an lseek (k,sizeof(struct exec),0) and t6.c should have a dependency on a.out.h in the makefile.On PNX there may be an extra structure with loader version & time information and the data may be aligned on a 512 byte boundary.

Checking the tab3.c file on my PERQ showed that it was the original BELL V7 version and did not have the correct width information for the Hershey fonts. If I had done my installation of Hershey font files using the makefile instead of by hand this would have been detected and tab3.c would have been updated.

Initially I thought I had correctly diff'ed the PNX troff srcs against the PDP11 troff srcs and found no difference in tab3.c. Somehow I managed to do this incorrectly, there are problems in that one set of files has to be transferred to the other machine and this is error prone.A distributed file system (e.g.Newcastle Connection) might make this process more reliable.

2. PDP11 Troff

To make a version of troff from sources on the PDP11 which does not print diagnostics on file descriptor 2 you need to use

```
     get -r3.1 /usr/src/cmd/troff/s.n1.c
     get -r2.1 /usr/src/cmd/troff/s.n3.c
     get -r2.1 /usr/src/cmd/troff/s.n4.c
     get -r2.1 /usr/src/cmd/troff/s.n9.c
```

The binary made from these sources is not identical to the one in /bin and I don't know how the /bin binary was made.Running tests on several examples using the new binary gave the same output as the /bin binary.

These anomalies will be removed when I insert some comments in the source code and remove the diagnostic printing calls.

The tmake and nmake makefiles have to be modified to add -i to CFLAGS and consequently suftab.o and hytab.o must be forced to remain in the data segment by commenting out the ed .. < textscript lines. The nroff/troff makefile does not specify dependencies on the .o files it only tests for existence of the relevant binary file.Thus even if the .c files have been updated (e.g. tab3.c) troff will not be remade.This needs to be corrected on all machines.

There is also a problem that the existence of a .o file does not allow detection of whether it was compiled with -DNROFF on the CFLAGS command line or not.The safest way to overcome this is to always remove .o files and force nroff/troff to be remade.

3. Conclusions

I used a process of backward reasoning in assuming the fault was within troff when a simple analysis of the symptoms of the problem and some careful thought should have indicated the fault lay within the width tables.

If I had read Kernighan's paper "A Typesetter Independent Troff" more closely I would have discovered the encoding for motions and characters at a much earlier date.

I was not careful enough about using the tools diff and make which would have allowed the problem to be detected and corrected earlier.

Although I have wasted quite a lot of time finding the cause of the problem I have gained some understanding of the troff code and of its use of bit patterns.