From: kent@lloyd.camex.uucp (Kent Borg)
Subject: LightSpeed Pascal pretty-printing
Date: 3 Nov 88 15:41:46 GMT
Organization: Camex, Inc., Boston, Mass USA

In article <1151@usfvax2.EDU> pollock@usfvax2.UUCP (Wayne Pollock)
writes:
>Below is a LSP program (re-pretty-printed to save screen space; why oh why
>can't I turn this option off?). ...

Because it is not an `option'.

Rich explained this a while back. What I remember: One of the reasons LSP has such a fast turn around is that when you say `Compile' a lot of (all of?) the parsing of your program is already done, it happened when you typed the `;' for that line. The compiler is storing your program in some tight, tokenized, internal form. This has the natural side effect striping out formatting information that the compiler has no use for. Remember back when you were playing around with interpreted BASIC? You typed in `print' (or `?') but the listing said `PRINT'. Similar effect.

Certainly the pretty printing could be avoided without loosing the fast turn around and source code debugging LSP does so very well, but I can see how these changes would be far from trivial and not without cost in time and space. That coupled with idea that pretty-printing might be a _feature_ helps explain why it is the way it is.

LSP 2.0 is supposed to allow the user lots of opportunity to twiddle the pretty printing parameters--but it is still part of the product.

Personally, though I like the pretty printing, I do have some gripes:

It will sometimes looking at my work too soon.

On the other side it is sometimes hard to get it to reconsider a line once I have fixed a typo. This was a particular problem for a class I recently taught at the Boston Computer Society where LSP was used for examples. Students would have a perfectly fine Pascal statement sitting on the screen, but the editor was still upset over an earlier typo. In this case it is looking at our work too late.

This incremental stuff is complicated, and when I think about it, I am impressed with how gracefully they do it.

Further consideration for all you grumblers: LSP is great and there are ways they could make it still better. Getting rid of the pretty printing will not be trivial. Do you really want them to take time away from making substantial improvements just to fix this? Wouldn't you rather get inline assembly code, or something else that actually adds power to the product? If so, say so, and don't complain so much.

(Personal LSP request: Editor support of arrow keys, option arrow keys, command arrow keys, shift option arrow keys, etc.)
--
Kent Borg
kent@lloyd.uucp
or
hscfvax!lloyd!kent

From: singer@endor.harvard.edu (Rich Siegel)
Subject: Re: LightSpeed Pascal pretty-printing
Date: 5 Nov 88 00:11:54 GMT
Organization: Symantec/THINK Technologies, Bedford, MA

In article <244@lloyd.camex.uucp> kent@lloyd.UUCP (Kent Borg) writes: >(Personal LSP request: Editor support of arrow keys, option arrow >keys, command arrow keys, shift option arrow keys, etc.)

Consider this: In version 2.0

Arrow keys move as you expect.

Option-Left and and Option-Right arrow move t oprevious and next semantic unit (a token, that is). Option-Up and Option-Down move to the previous or next procedure/function.

Command-Left and Command-Right arrow move to beginning/end of current line, Command-Up and Command-Down arrow move to top/bottom of window.

Command-Mouse Down in the title bar gives a list (in order of implementation) of all procedures in the file. Command-Option mouse down gives the same, sorted alphabetically. Selecting an item takes you to the beginning of that prorcedure. Option-Mouse gives a listing of units USED by the current file, and choosing an item opens the source file that contains the unit.

--Rich

--
Rich Siegel
Staff Software Developer
THINK Technologies Division, Symantec Corp.
Internet: singer@endor.harvard.edu
UUCP: ..harvard!endor!singer
Phone: (617) 275-4800 x305

From: netnews@caen.engin.umich.edu (CAEN Netnews)
Subject: Re: Lightspeed Pascal
Date: 9 Nov 88 05:43:00 GMT
Organization: University of Michigan, Ann Arbor

>'Files which contain computed references can not be converted.' this surprised
>me because the Black and White version had converted succesfully. The color
>From: billkatt@sol.engin.umich.edu (Steven James Bollinger)
Path: sol.engin.umich.edu!billkatt

> The fact that one program converts and one doesn't means nothing.

> The reason that the Lightspeed Pascal .O converter cannot convert
>the aforementioned .O file is due to the design of the Lightspeed Pascal
>linker, which doesn't support computed references. The LightspeedC linker,
>which is of newer design, does so, as far as I am aware.

I figured that it was a limitation in the definition of the libraries and how the linker handles them. Revisions to the linker would be major, instead of minor changes to the converter. But that is still a problem. Lightspeed Pascal should have been revised to fix the problem long ago (at least a year). I also only mentioned that the C converter would convert it to show that it was not a corrupted file and that it is reasonable to expect a non-MPW language to support computed references.

>> It really surprises me that I continue to stick with a company which
>>constantly hangs its Pascal customers out to dry in favor of c programmers.
>
> Frankly, this is a patently unfair statement. We ofer no favor towards
>users of either language. It is true that upgrade patches and so forth have
>been more abundantfor C than they have for Pascal, but this does not mean
>that we hate Pascal programmers, or that we love C programmers, or that
>we're trying to hang anyone out to dry. With respect to Pascal, we
>have not been idle, and I resent any implication to the contrary (since I

>am one of these non-idle programmers). It was our decision to concentrate
>our effort on producing Lightspeed Pascal 2.0, which is superior in
>all respects to competing products, AND to version 1.11
> rather than delay version 2 for the sake of providing intermediate patches.
> I stand behind what I said in this case. I feel you do provide favor toward THINK C users. I wouldn't call two major upgrades (1.0->3.0) and various minor upgrades comparable to LS Pascal's 1 minor upgrade (1.0->1.1) and a sub-minor upgrade (1.1->1.11, although close to my heart because I own a Mac II). And I wouldn't say you are idle (I have a friend who is betaing LS Pascal 2.0), only that you were idle for too long. If you care to disagree, I would suggest that you respond in reference to the continuing (2 years now) lack of cursor-key support, or the inablility to specify flags for CODE resources generated or programs generated (right down to the good ole' bundle bit).

Also, I wouldn't complain about things so close to the release of a major revision if I felt that the major revision were due soon. I had expected it at the end of September, but was disappointed, the end of October was disappointing to me too. I think you should have released revisions for the old one (1.11) just before you began work on 2.0, in order to tide us over.

>>updates than LS Pascal has. I suppose this will be rectified in LS Pascal
>>2.0, but I have long since given up belief in its existence. Besides, good
>
> Assuming that you've filed your upgrade form, you'll have
>version 2 in your hands extremely soon. (Unfortunately, I'm not
>allowed to say when until it happens, but when I can say more, I will.)
>

I understand the gag order, and believe in not pre-announcing products. I admire the person who fired your previous ad agency for pre-announcing THINK C 3.0. But I can't even believe you when you say extremely soon because being on a project always produces high hopes and great expectations. I am writing this message from Ann Arbor, Mi., and with FullWrite, this town invented the term 'real soon now', so I have learned to wait until I see something (actually, see it on shelves) before I believe it.

I think it is best expressed by the following situation: I have a friend who betas LS Pascal 2.0, and he thinks I am some kind of programming god. He thinks so simply because I write a cdev with INIT portion (Ignisound) in Lightspeed Pascal 1.11. It amazes him that I could possibly write such an animal in a language that has core routines that are essentially 3 years old. On a better note, he wants me to give him my code so he can test it under 2.0 and report back any incompatibilities to you, which makes him a great choice for a beta tester (hindsight is 20/20).

I am not trying to become your adversary. I did pick Lightspeed Pascal as my developing language after I recieved no support from TML. (Doing ANYTHING in TML gives you a sense of accomplishment, with its cryptic instructions and lack of debugger.) I still use it despite its creaking joints and gray hairs. I also know great things are afoot (if THINK C 3.0 is any indication). I can best sum it up as I did on my summary line: 'Its not too little, its just too late'.

If 2.0 comes out within a week I'll feel real stupid.