FROM: G. Schroeder

DATE: October 25, 1965

SUBJ: Proposed Changes to SCNDIR and STICKY for System Speed-Up

SCNDIR:

There are two parameters to SCNDIR, NDIR and NREAD. NDIR is the number of directories which can be open simultaneously. Each directory that is open costs 435 words. NDIR is currently set to one. NREAD is the length of a read buffer for file directory searching. NREAD must be a multiple of the length of one file directory entry (7). NREAD is now set to 70. (10 x 7)

The system could probably be speeded up significantly by increasing the size of the read buffer and the number of buffers used in directory swaps.

The worst case on closing out an old directory and opening a new one can be summarized as follows:

	Track Read of Write	
for one buffer:	1	old user's U.F.D.
absorbed by stickies always	3	M.F.D. search (average; M.F.D. 6 tracks long)
	1	write (update old U.F.D. entry in M.F.D.)
absorbed always by stickies	2	read and write 1st track of M.F.D. (lcount)
	2	search M.F.D. for comfil 2 (average)
stickies	1	read of comfil 2 (average)
	1	write of comfil 2
	3	M.F.D. search
	1	write
	2	read and write first track of M.F.D.
	2	search M.F.D. for new user (average)
	1	new user's U.F.D.
	10 @ app 52 ms each	
	=.5 sec.	

This represents the worst case when the system is so heavily loaded that the stickies are little help. When we are running with 30 users, we swap about every 2 seconds so this number (.5) is quite significant.

If the number of buffers were increased to three, the M.F.D. would probably be kept in one, comfil 2 in another, and users would share the third. The summary would then be:

	Track Read or Write	
for three buffers:	1	old user's U.F.D.
	2	search M.F.D. for new user
	1	new user's U.F.D.
	4 @ 52 ms	
	= .2 sec.	

In order to implement this change, the parameter NDIR must be changed in SCNDIR and the usage algorithm which determines which buffer is used when a new directory is to be read in would have to be improved.

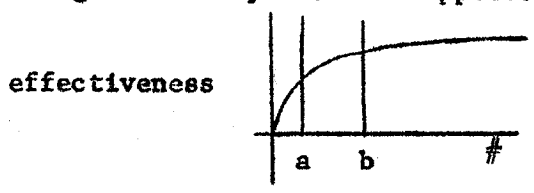
The 70-word read buffer could also be increased to 700 - it must be a multiple of seven. This would cause file directories to be read 2 tracks at a time. One waits almost the same amount of time to read one track as two so one might as well read two. Any more than two might become inefficient because of the average size of directories. If one read 4 tracks at a time, for instance, half the time one would read 2 tracks unnecessarily - so there is a trade-off.

It would be very useful to have a monitor in SCNDIR which tells how often directory swaps take place. In order to obtain any realistic notions as to the effect of increasing the number of buffers, this is necessary. It can easily be installed. It would be useful also to know how much time is spent in directory swapping, but because of the recursive nature of the search module a timer is difficult to insert.

One more suggestion for speed-up: On updates the M.F.D. date and time last used is updated. This could be done less often than it is by making the compare between last entry and current time a little less exact (mask off some number of less significant bits). This would cause no ill effects and would save a considerable number of writes.

STICKY REGISTERS:

It is difficult to say what the effect would be of increasing the number of sticky registers. It might be possible to find out, however, by crippling sticky registers progressively and plotting the number of effective sticky hits against number of sticky registers. Stan says the crippling is easily done by patching and the effectiveness monitoring is already in. He suppects the curve looks like this:



We now have 58 stickies. If 58 is at a, it is worth the effort to increase the number of sticky registers. If 58 is at b, it is probably not worthwhile.

The number of sticky registers can be increased by changing the parameter NSTIK.

Another change which requires little programming is to change the usage algorithm. Now when a sticky is needed, the one with lowest usage is flushed. Perhaps one could look for the one with lowest usage and requires no writing to flush - with some default, of course, when there are none that require no writing. Stan suspects that the ones that require writing will be flushed regularly enough by calls to update.

Redundant updating of U.F.D. and M.F.D. entries is being done because STICKY cannot tell when changes have cancelled each other. For example, when file is opened for reading, the STICKY entry is changed because ILOCK is set. Subsequently the file is closed, ILOCK is reset, the sticky entry is changed again. If that entry has not been updated meanwhile, the two changes have cancelled each other but STICKY cannot tell. If one could monitor the number of times a file is opened for reading and the number of times any M.F.D. entry is re-written due to swapping of U.F.D.'s, one could get some idea of how much redundant re-writing is taking place.

This redundant re-writing could be avoided if the sticky registers were each increased 5 registers (they are now each 13), and STICKY could see the state of the entry on the disk and the correct state.

This is a significant programming change to STICKY.