

THE LATEST WORD ON THE EXCITING STRUGGLE TO OVERCOME THE ELUSIVE
BLOCK*GONE*FROM*FILE*IN*MAP/CHANGE*UNIQUE*NAME*OF*FILE PROBLEM
(CONSIDERED IN CONJUNCTION WITH TURNING MAPS ON AND OFF)

Maps now have two counts on the compiled part and a new flag on the logical part

- 1) a local BADMAP count
- 2) a local COMPACTION count
- 3) a map on/off flag.

There is in addition a new flag on subprocesses, but it is very elusive. It says ~~either~~ or not the subprocess is suffering from a pending map error. *whether*

To begin with the map on/off flag

- 1) An action to turn off the map of a specified subprocess will be produced in due course. It will decrement the map count on all ~~blocks~~ file blocks used by the map and set the bit to off. (One gets an error for trying to turn off the map of a subprocess currently in the full path. Is that OK with everybody?)
- 2) ~~trying to~~ doing anything that might cause a subprocess with its map turned off to swap in will cause an error, as discussed below.
- 3) The operation to turn the map back on will be fraught with all sorts of hazards stemming from missing blocks and files, but if one is lucky, it will find everything present that is necessary and increment the map count on all the relevant file blocks and turn the bit off.

When one changes unique names on a file, if the file has a block in a map, the map count on the block is cleared and a global BADMAP count is incremented. This leaves some map, somewhere, sitting around with one of its files ripped off. This may later lead to an error as discussed below.

I regret that I must also mention that some careless code may callously ^{kill} destroy a c-list that is the local c-list of some innocent subprocess, thereby causing said innocent subprocess grave embarrassment. (When the current swapper tries to bring in such a subprocess, it destroys the process!) But have no fear, relief is at hand.

How is one to see one's way out of this quagmire? Well, let's start with the hard-working swapping code, MAPOUT/MAPIN, which do the bulk of the systems swapping work. Here comes this subprocess to be swapped out/in. If the two local counts on the compiled map are up-to-date, ^{and the map is on,} the swap proceeds. But, if a count is off, further action is taken

- 0) if the map is off, an error is signaled to the caller (no swapping)
- 1) if the COMPACTION count is off, the map is recompiled (or compiling)
- 2) if the COMPACTION count is OK, but the BADMAP count is off, a check is made to see iff all files in the logical map are still present; if so, the count is updated and the swap proceeds, but if not, the map is recompiled.

Whenever the map compiler encounters a missing file in a logical map, it zeros the map entry and proceeds with the compilation. ~~XX~~ It ^{later} exits with a signal if a file was gone. MAPOUT/MAPIN return this signal to whomever called them. The map is then swapped (with a possibly newly zeroed entry).

closest to the current running subprocess is reported.

Well, I sure am glad to have that off my mind. Oh yes, I forgot to mention that when the map compiler encounters a missing block when it's compiling a logical map entry, it is still a DISASTER.