20 April '70

ADDITION TO DIRTY BIT SPECS

Howard points out that it should be mentioned that move block carries the dirty bit along with the block.

KARL'S OBJECTION

With respect to the blurb on allocation last time, Karl says that the objects in ECS are set up so that one object can be moved and without necessitating the relocation and/or massaging of some large fraction of ECS and that's all there is to incremental compacting; he was annoyed that I said incremental compacting was left as an exercise, since it is obvious, trivial, etc., how to do it.

Karl is right about this. What I was objecting to was a lack of any sort of general description of the problems that were supposed to be handled by incremental compacting, like speed freaks, interrupt code, compacting during the idle loop, etc. This lack is no doubt not ascribable to Karl as it seems to be a more or less general quality of the documentation. I'm putting together a little blurb on the compactor which may be out next week.

Meanwhile, if Karl isn't mollified by this, he can submit his own disclaimer for next week. OK?

ALLOCATION DECISIONS

Last week, we decided to do whatever was necessary to

1) fix the existing bugs
2) free one end of ECS
3) provide an incremental compactor which will:
   a) massage some number of real cells or collect some number of free cells or both (see compacting document)
   b) do the normal I.WAIT/I.LOCK logic to allow interrupts
   c) monitor a new cell (I.COOL?) which will cause the compactor to suspend compacting when the cell gets set and take a special exit (for to run speed freaks).

It was decided not to do the engineering necessary to make it possible to create arbitrary size objects (unless it somehow falls out).

BENT FILES

All talk about speedfreaks, incremental compacting, etc., is vacuous unless we get to the bottom of the bent file problem. I would like to hear exactly what the problem is and a decision as to what's to be done about it.

CPU TIME ACCOUNTING

I would like to have Jim Gray order Howard and I to figur out how CPU accounting is going to be done. Discussion of the properties and problems of CPU accounting is in order.

GLOBAL INTERRUPT INHIBIT

Solid proposals and decisions for implementing the GIIB are in order. Also, the configuration of the user's process tree vis-a-vis interrupt handling could use some clarification.

DIRECT ACCESS REVISITED

At the 17 April disk system meeting, the old headache of DAE's was discussed again. It was realized that we were into giant headaches and going around in various size circles, so we backed off and started all over again. What the hell are DAE's good for anyway? The only use that the participants could suggest and defend was to give the user access to a large, fast address space. (Proponents of other uses, please step forward at once.)

On the basis of the "large block" theory, the following decisions were tentatively made:

1) At the ECS level, blocks have to be created nudged, as opposed to created first and then nudged later. This avoids fairly unpleasant problems encountered when trying to find space to relocate a large block.
2) At the disk level, big 0-level files and only big 0-level files are always created nudged. This is fairly restrictive, but it is adequate to the only use so far proposed for DAE's. (A side kludge is that big directories will be implemented as multi-level files unless we want them nudged, but that seems OK.) "Big" remains to be defined exactly.

REALLOCATION

Currently, when an object is being reallocated, if it can't be expanded in place, it is briefly charged to it's father allocation block twice (while another place for it is being found and it is being relocated). This has at least two bad consequences:

1) It makes it slightly difficult for you to control accurately the space used by some untrustworthy subprocess.
2) For every user in the system, the disk system has to either give him space for 2 process descriptors or be very tricky about allocation/changes to the process descriptor.

The reason behind the double charge is to avoid locking up ECS 'cause of space problems. Mechanisms for avoiding the double charge and not causing a disaster are under consideration. There seem to be only 2½ kinds of things which get reallocated

1) Process descriptors
2) Operations
2½) Maybe nudged blocks, pending the outcome of the DAE debate.

Since 1 & 2 are small, they could usually be handled by "hiding" some number of cells of ECS and counting on these for the relocation of small objects; larger objects could be doubly charged as at present, or handled out of some kind of system space pool (with the possibility of failure, since there may not be enough system space).

Anyone have a nice solid idea?