Inter-Office Memorandum

To	Distribution	Date	December 1, 1977
From	John Wick	Location	Palo Alto
Subject	Code links meeting	Organization	SDD/SD

XEROX

DRAFT

Filed on: [MAXC]<WICK>CODELINKS1.BRAVO

The Mesa Working Group met on Wednesday, November 30, 1977 to discuss the implementation of code links proposed in the following memo:

Wick. *External links in Mesa.* November 29, 1977.

The following revisions and additions to the proposal were discussed.

Allocation of External Links

We decided not to optimize access through imported frame handles (by putting them unconditionally in the frame instead of the linkage vector), and to therefore avoid splitting the linkage vector and complicating the BCD format and the UNNEW operation. It was felt that the improvement in code space for procedure calls was very small, and that most data access involved initialization, for which code space is not very critical. This should be revisited later, when we have more data on how imported frames are used.

Initialization Code

A proposal was made to allow identification of temporary (initialization or other deletable) modules in configuration descriptions. The idea is that when a configuration is instantiated, two separate segments are allocated for the global frames, one containing all the temporary frames and the other containing all remaining (permanent) frames. Word zero of the permanent segment would point to the temporary one (if any). The operation DELETE could be executed from any of the permanent frames (perhaps from outside the configuration?) and would do an UNNEW on each frame in the corresponding temporary frame segment. The binder should probably check that there are no bindings from permanent frames to temporary ones and at least issue a warning.

It was not entirely clear how a series of nested configurations, each with its own set of temporary frames, should be handled (in general, a user of a configuration won't know if it has temporaries or not). A DELETE executed from within a nested configuration could be postponed until a delete at the outermost level is performed (which would apply to the entire configuration). Some method of identifying the outermost DELETE must be devised.

An UNNEW on a permanent frame would then become illegal (or it could perhaps do everything except free the frame). This will ensure that a configuration can always be unloaded, without fear that any of its UNNEWed global frames might be in use. It is the user's responsibility to insure that no references to the configuration's frames are on any call stack.

Binder/Loader Modifications

We next addressed the question of how and when the option for code links should be specified. The central problem is that if links are determined at load time as they are now, the loading time will be increased substantially by rewriting code segments. The idea is to somehow determine the links at binding time, so they can be added to the code as it is being rewritten to make room for the links. The following possibilities were identified:

0. Current "soft" scheme (all *gfis* are relocatable until loaded).

1. Bind as is currently done, but make room in the code segment for links, and mark the module table entry in the BCD to indicate this. The move code option must be specified.

Pros: Simplest to implement.

Cons: Loading will be slowed down considerably, as code containing links will have to be swapped in and rewritten. (Currently, the code is not touched at all during loading.)

2. Bind to an image file (and perhaps some number of BCDs which have been "hard" bound to the same image file). Hard binding means that all code and frame links are assigned real *gfis*.

Pros: Fast loading; no other BCDs need to be read.

Cons: Depends on the image file; no rebinding is possible. Real *gfis* can conflict with those assigned to other BCDs. Requires work on the binder to input image files and BCDs with real *gfis*.

3. "Hard" binding only for modules with code links. Relocatable links will be kept in the frame fragments regardless of where the links are to be eventually stored.

Pros: As fast as case 2 if all modules have code links. Loader can rebind to other image environments if necessary (perhaps by providing frame links, perhaps by rewriting the code).

Cons: As slow as case 0 if any module has frame links. Real *gfis* can conflict with those assigned to other BCDs. Requires work on the binder to input image files and BCDs with real *gfis*.

At this point the scribe became confused, and failed to record the remainder of the discussion. The next meeting will address the resolution of this issue.

Distribution:
- Geschke
- Lampson
- Mitchell
- Satterthwaite
- Mesa Group