DRAFT

TO: N. ADLEMAN
C. CLINGEN
F. CORBATO
R. DALEY
R. FREIBURGHOUSE
C. GARMAN
J. GINTELL
R. GRAHAM
A. KVILEKVAL
M. MEER
J. SALTZER ✓
K. SHIH
B. WOLMAN

FROM: G. S. STOLLER

DATE: JUNE 2, 1969

SUBJ: NEW MULTICS BINDER

Here are two of the MSPMs describing the Multics binder.

Please note an interesting fact about the binding strategy of the new binder. In contradistinction to the current binder, what the current binder calls "post-binding" and performs after the relocation-pass, the new binder will do before the relocation-pass (something like "pre-binding").

Identification

Overview of Binding

G. S. Stoller

Purpose

Multics modules (e.g. MST editor, linker, listener) are usually encoded as several distinct procedures. Several reasons contribute to this, mainly: checkout-ease, limits imposed by the language processors and the system, logical submodules, etc. Among the (problematical) effects of this are: lost core-space due to breakage, multiplicity of segments (and segment-faults), additional linkage-faults, and other minor effects (e.g. replication of constants and compiler-generated internal subroutines).

Binding solves several of these problems, namely: breakage, multiplicity of segments (and segment-faults), and additional linkage-faults. "Binding" is the name of the process of combining one or more object segments into components of a bound-object-segment with each of the inter-segment references originating and terminating in the bound-segment replaced by a direct reference (i.e. a specifically intra-segment reference).

Glossary

1. "linkage-block" is that portion of a linkage-section in which the links and (major) entry-points are kept; i.e. the linkage-section except for the header and internal static.

2. "link-snap-info" is the information used for snapping links; it consists of the definitions (entry-point and segdef names), external names, type-pairs, expression words, and trap words.

3. "linkage-info" consists of both linkage-block and link-snap-info.

4. "ego-" is used as a prefix to describe that part of a segment or section used only intra-segmentally; i.e. omit the linkage-info from these considerations.

Design Objectives

The binder is to produce an object segment from several (input) object segments. Its output object segment is to resemble as closely as possible the result had we "assembled together" (ignoring differences of language used for encoding) the source of all the object segments being bound.

In view of this objective, the resultant bound segment will have exactly one linkage-block and one link-snap-info block. All of the internal static will be gathered together into one contiguous block of memory.

User Interface

In order to bind several segments, all that the user has to do is to type

bind x

(possibly this is followed by some option specifications) where "x" is a segment-name. For simplest usage, "x" is the name of an archive segment and all object-segments within the archive-segment will be bound together, the resultant bound segment will be in object-format.

Greater detail is given in a description of the "bind" command (See BX.14).

Local Binding Stuff

Each segment has associated with it some information to enable it to be bound. This information is produced by the translator that produced the segment; the information specifies the relocation-base for each half-word.

Section BD.2.01 of the MSPM gives a fuller description of this information.

Global Binding Stuff

Actual binding is done by the procedure-module <bind> which interprets the local binding stuff to direct it in producing the bound-segment. In addition to the bound-segment, a map of the bound-segment is produced.

During the binding process, all linkage-information is unravelled and decomposed into atoms (that are component-independent); this will automatically achieve coalescence. The linkage-information is rebuilt prior to being inserted in the bound-segment.

Constructs not allowing Relocation

It is quite easy for a coder, particularly an assembly language coder, to produce unbindable code. Just by using the vfd-pseudo-op (to produce an unusual size field) or adding symbols attached to different bases (e.g. "lp|x + y" where x is an offset in the linkage-section and y is an offset in the text-section) a coder can create a datum for which none of our translators can produce proper relocation-information.

Binding Obstructions

In some cases, a binder may not allow coalescence to proceed so far that a link is deleted. This is the case when the location of the link is referred to, rather than passing through the link by indirection. See BV.5.

Other obstructions occur in the case of a section that is to grow during usage by a process.