New Technical Notes

Macintosh

Developer Support

MPW: {$LOAD}; _DataInit;%_MethTables

Platforms & Tools

Revised by: Jim Friedlander January 1987
March 1988
Written by: Jim Friedlander November 1986

This technical note discusses the Pascal `{$LOAD}` directive as well as how to unload the `_DataInit` and `%_MethTables` segments.

{$LOAD}

MPW Pascal has a `{$LOAD}` directive that can dramatically speed up compiles.

```
{$LOAD HD:MPW:PLibraries:PasSymDump}
```

will combine symbol tables of all units following this directive (until another `{$LOAD}` directive is encountered), and dump them out to `HD:MPW:PLibraries:PasSymDump`. In order to avoid using fully specified pathnames, you can use `{$LOAD}` in conjunction with the `-k` option for Pascal:

```
Pascal -k "{PLibraries}" myfile
```

combined with the following lines in `myfile`:

```
USES
    {$LOAD PasSymDump}
        MemTypes,QuickDraw, OSIntf, ToolIntf, PackIntf,
    {$LOAD} {This "turns off" $LOAD for the next unit}
        NonOptimized,
    {$LOAD MyLibDump}
        MyLib;
```

will do the following: the first time a program containing these lines is compiled, two symbol table dump files (in this case `PasSymDump` and `MyLibDump)` will be created in the directory specified by the `-k` option (in this case `{PLibraries})`. No dump file will be generated for the unit `NonOptimized`. The compiler will compile `MemTypes, QuickDraw, OSIntf, ToolIntf, PackIntf` (quite time consuming) and dump those units' symbols to `PasSymDump` and it will compile the interface to `MyLib` and dump its symbols to `MyLib`. For subsequent compiles of this program (or any program that uses the same dump file(s)), the interface files won't be recompiled, the compiler will simply read in the symbol table.

Compiling a sample five line program on a Macintosh Plus/HD20SC takes 62 seconds without using the `{$LOAD}` directive. The same program takes 10 seconds to compile using the `{$LOAD}` directive (once the dump file exists). For further details about this topic, please see the *MPW Pascal Reference Manual* .

Note: If any of the units that are dumped into a dump file change, you need to make sure that the dump file is deleted, so that it can be regenerated by the Pascal compiler with the correct information. The best way to do this is to use a makefile to check the dump file against the files it depends on, and delete the dump file if it is out of date with respect to any of the units that it contains. An excellent (and well commented) example of doing this is in the *MPW Workshop Manual*.

The _DataInit Segment

The Linker will generate a segment whose resource name is `%A5Init` for any program compiled by the C or Pascal compilers. This segment is called by a program's main segment. This segment is loaded into the application heap and locked in place. It is up to your program to unload this segment (otherwise, it will remain locked in memory, possibly causing heap fragmentation). To do this from Pascal, use the following lines:

```
PROCEDURE _DataInit;EXTERNAL;
...

BEGIN  {main PROGRAM}
UnloadSeg(@_DataInit);
{remove data initialization code before any allocations}
...
```

From C, use the following lines:

```
extern _DataInit();
...
{ /* main */
       UnloadSeg(_DataInit);
       /*remove data initialization code before any allocations*/
...
```

For further details about Data Initialization, see the *MPW Reference Manual*.

%_MethTables and %_SelProcs

Object use in Pascal produces two segments which can cause heap problems. These are `%_MethTables` and `%_SelProcs` which are used when method calls are made. MacApp deals with them correctly, so this only applies to Object Pascal programs that don't use MacApp. You can make the segments locked and preloaded (probably the easiest route), so they will be loaded low in the heap, or you can unload them temporarily while you are doing heap initialization. In the latter case, make sure there are no method calls while they are unloaded. To reload `%_MethTables` and `%_SelProcs`, call the dummy procedure `%_InitObj`. `%_InitObj` loads `%MethTables` —calling any method will then load `%_SelProcs`.

Reminder: The linker is case sensitive when dealing with module names. Pascal converts all module names to upper-case (unless a routine is declared to be a C routine). The Assembler default is the same as the Pascal default, though it can be changed with the CASE directive. C preserves the case of module names (unless a routine is declared to be `pascal`, in which case the module name is converted to upper- case letters).

Make sure that any external routines that you reference are capitalized the same in both the external routine and the external declaration (especially in C). If the capitalization differs, you will get the following link error (library routine = `findme`, program declaration = `extern FindMe();`):

```
### Link: Error  Undefined entry, name: FindMe
```

Further Reference:

- MPW Reference Manuals