MPW 3.0: How To Build Header Files and Libraries

Revised: 12/22/89
Security: Everyone

MPW 3.0: How To Build Header Files and Libraries

==========

This article last reviewed: 12 December 1989

TOPIC ----------

When converting from MPW C 2.0 to MPW C 3.0, I ran into the following problem. It seems the linker can't find some routines.

Have these been renamed or replaced? If so, the compiler accepted my calls since they appear in the #include files. I tried relinking with every library that shipped with MPW 3.0 and always got the same error.

Following is my link command and its output:

```
Link -w -t APPL -c Kran -sym on -mf C Option-D
   ... /* a bunch of my filenames */ Option-D
   "{CLibraries}"CRuntime.o Option-D
   "{Libraries}"Interface.o Option-D
   "{Libraries}"DRVRRuntime.o Option-D
   "{CLibraries}"StdCLib.o Option-D
   "{CLibraries}"CSANELib.o Option-D
   "{CLibraries}"Math.o Option-D
   "{CLibraries}"CInterface.o Option-D
   -o klink
setfile -a B klink

### Link: Error: Undefined entry, name: (Error 28) "KillIO"
  Referenced from: setupSerComm in file: INIT.c.o
### Link: Error: Undefined entry, name: (Error 28) "OpenDriver"
  Referenced from: setupSerComm in file: INIT.c.o
### Link: Error: Undefined entry, name: (Error 28) "RAMSDOpen"
  Referenced from: setupSerComm in file: INIT.c.o
### Link: Error: Undefined entry, name: (Error 28) "FlushEvents"
  Referenced from: main in file: main.c.o
### Link: Errors prevented normal completion.
### MPW Shell - Execution of input terminated.
```

DISCUSSION ----------

We were able to successfully link a short C program that made all of these calls. We suspect that not all of the necessary files had been "included", which would definitely affect the way the linker operates. The following are the steps you can use to determine what header files and libraries are necessary for a complete build of your program:

1) With your directory set to the {MPW}Interfaces:CIncludes: folder, use the search command to search all of the header files for the routine you want to call. For example, we used the command:

   search killio Option-X

   where Option-X is the keystroke that generates the MPW wildcard character for filenames. This returned:

   File "Devices.h"; Line 112 # Pascal OSErr KillIO(short refNum);
   File "Devices.h"; Line 115 # Pascal OSErr PBKillIO(ParmBlkPtr paramBlock,Boolean asynchronous);
   File "Errors.h"; Line 32 # #define abortErr -27
                                        /*IO call aborted by KillIO*/
   File "Traps.h"; Line 597 # #define _KillIO
                                        0xA006

   Therefore, the header file that must be included is Devices.h.

2) Create a simple program that artificially calls the routines you want to use. In this case, we used:

```
#include   <Types.h>
#include   <OSEvents.h>
#include   <Devices.h>
#include   <Serial.h>
main()
{
OSErr theResult;
short  *bogusRefNum;

FlushEvents(0,0);
theResult = KillIO(0);
theResult = OpenDriver("\pbogus driver name",bogusRefNum);
theResult = RamSDOpen(sPortA);
}
```

3) Use the Create Build Commands menu item from the Build menu to create a make file for your artificial program, and build the program to verify that it links properly.

(NOTE: Be sure you do not actually run the program since it is unlikely to run properly and, as in the case of our code, will crash.)

4) Open the make file generated, and you will have the necessary link command for the routines you wish to use. The make file generated for us follows:

```
#   File:       testflush.make
#   Target:     testflush
#   Sources:    testflush.c
#   Created:    Monday, November 13, 1989 3:20:14 PM

testflush.c.o Option-F testflush.make testflush.c
    C  testflush.c

SOURCES = testflush.c
OBJECTS = testflush.c.o

testflush Option-F testflush.make {OBJECTS}
   Link -w -t APPL -c '????' Option-D
      {OBJECTS} Option-D
      "{CLibraries}"CRuntime.o Option-D
      "{Libraries}"Interface.o Option-D
      "{CLibraries}"StdCLib.o Option-D
      "{CLibraries}"CSANELib.o Option-D
      "{CLibraries}"Math.o Option-D
      "{CLibraries}"CInterface.o Option-D
      -o testflush
```

Our C file was "testflush.c", and the generated make file was "testflush.make".

Tech Info Library Article Number:4949