October 1989   THE RAINBOW 82

Feature:

Getting started with cgfx functions and improving your system
with a fast compiler utility

What You Should Know About Your C Compiler

BY Numa David

The C compiler release predates the Color Computer 3. hard disk drives for the CoCo, OS-9 Level II. and the Level II Development System with its vdd (RAM disk). Neither the compiler nor the manual has been updated to achieve the high performance and compiling speed possible with the new hardware or software.

While the C compiler was being written, the CoCo did not have sufficient disk space to keep all the necessary files on one disk drive. The compiler was coded to look for files in the DEFS and Lib directories on Drive /d1, and the manual stated that the DEFS and LIB directories were on that drive. Disk space limitation no longer exists due to the recent improvements in hardware and software.

If your system has a 40- or 8O-track. double-sided floppy drive as /d0. your disk has space for more files. The compiler can be patched to look for its DEF and LIB directories there instead of on /d1, letting you keep all your system, commands and compiler files on one disk. If your system has a hard disk drive, you may not only keep all your files there but all file access including the compiler files.

Better yet. if your system includes the Level II Development System with the vdd device driver (RAM disk), there are patches and procedures that give you the high performance and speed of RAM-based compiling instead of disk-based compiling.

The instructions that cause the compiler to look on Drive /d1 for the DEFS and LIB directories are coded in the compiler files cc1 and c.prep. These files can be patched to cause the compiler to look for the DEFS and LIB directories on any drive you choose, including /r0 (vdd RAM disk) available with the Level II Development System.

The remainder of this article guides you through steps necessary to optimize the compiler to your system. Some initial notes to remember are:

- Perform the following on backup copies of your system and compiler disks. The system and compiler disks are modified, and it is possible for patch utilities to destroy important data on your disks.

- Drive /h0 is used in some of the following examples. Substitute /d0 if your system does not include a hard disk drive.

- The EZGen utility used is available from Burke & Burke, as advertised in THE RAINBOW. It is possible to use OS-9 commands (modpatch with os9qen OR cobbler) if you prefer; but, after considering the low price, the readers this will interest, and the inevitable complications that will be avoided. I concluded that EZGen is the practical choice for these examples. The objective here is to show you the simplest approach to a practical problem.

October 1989 THE RAINBOW 83

Compiler Patches

If you have a hard disk drive you can Patch a custom version of your compiler that searches Drive /h0 for the DEFS and LIB Directories as follows:

```
OS9: chd /h0/cmds
OS9: ezgen c.prep
l c.prep
c 135c 68
c 135d 30
v
q
OS9: ezgen cc1
l cc1
c 0EE5 68
c 0EE6 30
v
q
```

If you have a 40- or 80-track, double-Sided disk drive as /d0, you can patch a custom version of your compiler that searches Drive /d0 for the DEFS and LIB directories as follows:

```
OS9: ezgen c.prep
l c.prep
c 135d 30
v
q
OS9: ezgen cc1
l cc1
c 0EE6 30
v
q
```

If you have the OS-9 Level II Development System, your compiler can be optimized to run at maximum speed using the vdd (RAM disk) as follows:

Boot patches:

To add the r0_192k.dd device descriptor to your boot file:

```
OS9: ezgen os9boot
b
i /d1/modules/r0_192k.dd
q
```

To set Default Drive /dd to /h0 (omit this if you don't have a hard disk drive):

```
OS9: ezgen os9boot
l dd
u /d1/modules/h0
q
```

You'll want to use the cgfx functions from your Level II Development System. They are C graphics functions similar to the gfx and gfx2 functions in BASIC09. The manuals furnished by Tandy fail to give the necessary instructions required to compile programs with cgfx functions.

To compile cgfx functions rlink must be renamed c.link, and rma must be renamed c.asm as follows:

```
 OS9: chd cmds
 OS9: del c.asm
 OS9: del c.link
```

Now let's rename the headers for the MODULES directory.

```
 OS9: ezgen rma
 l rma
 r c.asm
 q
 OS9: rename rma c.asm
 OS9: ezgen rlink
 l rlink
 r c.link
 q
 OS9: rename rlink c.link
```

The above patches, using EZGen, act directly on the disk files; you do not use os9gen or cobbler. You simply reboot your system.

Note: Except for initial access to disk drives to move files to /r0 for compiling, the above compiles and links entirely in RAM − the disk drives do not run.

Merging the Library Files

The following merged cgfx library files are also merged with your programs when they are linked by the fast c.link procedure shown later.

```
 OS9: merge cgfx.l clib.l sys.l
 >merged.l
```

Include merged.l in the LIB directory. To ensure compatibility, be sure to use only the new linker supplied with the Development System.

Preparing the Initializer

Use the editor to prepare cc1_init, a procedure file to initialize the fast compiler, as follows:

```
 * cc1_init *
 iniz r0
 chd /r0
 makdir LIB
 chd LIB
 copy /h0/lib/merged.l merged.l
 chd /h0/defs
 dsave /h0 /r0 ! shell
 chx /h0/cmds
 load cc1
 load c.prep
 load c.pass1
 load c.pass2
 load c.asm
 load c.link
```

Setting Up the Fast Linker

Use the editor to prepare fast_link, a procedure file you can use to put specific programs in for linking. Using a procedure file avoids typing a long list of commands each time you recompile and relink your application.

The quantity of relocatable files merged below serves as only an example. The exact quantity depends upon the number of programs linked to form your application. Here is a typical procedure file:

```
 * fast_link *
 chd /h0/sources
 merge prog1.r prog2.r prog3.r >temp1
 merge prog4.r prog5.r prog6.r >temp2
 merge prog7.r prog8.r prog9.r >temp3
 merge temp1 temp2 temp3  >/r0/lib/
 prog.l

* The following compiler line links the      *
* CGFX functions in merged.l to your program *

c.link cstart.r -l=prog.l -l=merged.l -o=prog
```

This completes preparation of the system for fast compiling. Reset your computer and reboot.

To Use Your Fast Compiler:

Compile each of the source programs of your group of source programs to a relocatable object file as follows:

- Initialize the fast compiler by typing at the OS9 prompt:

```
  OS9: cc1_init
```

- Copy the program to /r0:

```
  OS9: chd /r0
  OS9: copy /h0/sources/prog.c prog.c
```

- Compile the program.

```
        OS9: cc1 prog.c -ro
```

Note. Debug and recompile if errors occur during compiling.

- Copy the program back to /h0:

```
OS9: copy proq.r /h0/sources/prog.r
OS9: del prog.r
OS9: del prog.c
```

You now have a group of relocatable object files that must be linked to form an executable object file as follows:

To Use Your Fast Linker:

To link the group of relocatable files to an executable object file, simply type at the OS9 prompt:

```
OS9: fast_link
```

If fast_link produces errors, debug the offending source program, delete the offending relocatable file, and repeat the compiler steps above. The executable program is saved in the CMDS directory. To run the program type at the OS9 prompt:

```
OS9: prog
```

Summary

The keys to this fast compiler are the patches that cause the compiler to look for DEFS and LIB directories on /r0 instead of /d1 and keeping the compiler commands loaded in memory for immediate execution instead of loading from disk drives.

Beyond that, many approaches and variations are possible for setting up the system for fast compiling. Enhancements and improving convenience and utility will undoubtedly occur to you. You can develop a completely interactive, menu-driven, fast compiler utility.

C Graphics Library

Now for the C graphics library. Your C compiler has available a new graphics library that expands the original C library to a state-of-the-art graphics programming language. C language graphics library functions similar to the gfx2 functions in BASIC09 are provided on the OS-9 Level II Development System disk as cgfx functions for the C compiler. You will want to use your cgfx commands.

However, essential steps required before using cgfx with the compiler are not included in the manual – the kind of steps that probably never occur to even experienced programmers. The following gives you the information needed to get started with cgfx functions.

Use of cgfx functions requires a Color Computer 3 with the following software: OS-9 Level II Operating System. OS-9 Level II Development System. C compiler and C library, and Multi-Vue. (You can use cgfx functions without Multi-Vue. but your cgfx documentation is in the Multi-Vue manual.)

If you haven't compiled a program using cgfx functions yet, the following will spare you some time, frustration and confusion:

Pages 10-1 and 10-2 of the Multi-Vue manual advise you to link the cgfx library along with other libraries to your C program, and give instructions along with a command line example (that does not work yet) as follows:

```
 OS9: cc1 prog.c -r
 OS9: c.link /d1/lib/cstart.r prog.r
       -l=/d1/lib/cgfx.l -l=/d1/lib/clib.l
       -l=/d1/lib/sys.l -o=prog
```

How frustrated a programmer can get if no one tells him that the cc1 and c.link modules used above are not the ones that came with the compiler and do not work until they are changed. You can't be expected to know this because it's not in the manual. Tandy knows about this specific problem – and one of its capable technical representatives will explain it if you call Tandy's Fort Worth headquarters. But how long does a programmer troubleshoot a command line example before he resorts to that? (Have a heart. Tandy – we need addenda for this one.)

The manual fails to advise that the old c.link and c.asm must be deleted from the CMDS directory, and that r.link must be renamed c.link and rma must be renamed c.asm before using the compiler with the above command line as follows. (Warning: Perform the following on a backup disk. Important compiler modules will be changed.)

It is assumed the rma and r.link commands from the Development System disk as well as c.asm and c.link from C compiler disk, are on commands directory on Drive /d0.

```
 OS9: chd /d0/cmds
 OS9: del c.asm c.link
 OS9: rename r.link c.link
 OS9: rename rma c.asm
```

Now you are ready to proceed according to the instructions and examples on Page 10-2 of the Multi-Vue manual However, I suggest first merging the library files as follows, assuming the library files from the Development System disk are in the LIB directory on /d1:

```
 OS9: chd /d1/lib
 OS9: merge cgfx.l clib.l sys.l
       >merged.l
```

Keep merged.l in the LIB directory. The rather long linker line above can be shorten in all future call as follows, provided your source code is on a directory named SOURCES on /d0 and the LIB and DEFS directory are on /d1:

```
 OS9: chd /d0/sources
 OS9: cc1 prog.c -r
 OS9: c.link .d1/lib/cstart.r prog.r
       -l=/d1/lib/merged.l -o=prog
```

Debugging Your Manual

It may save you more time and confusion to know the manual contains errors in some cgfx command line examples. The following, from Page 10-21 of Multi-Vue, will help you debug your manual.

```
 SetGc (path,grpnum,bufnum)  Wrong
       SetGC (path,grpnum,bufnum) Right
```

If you are uncertain about path simply use 1 to indicate standard output.

I don't know how many cgfx command line errors are in the manual, but when (not if) you run into other cases where everything seems OK but you get an Unresolved References Error, you can determine whether the command from the manual is correct by using rdump as follows:

To dump system command headers to your screen using rdump from the Development System:

```
 OS9: chd /h0/lib
 OS9: rdump cgfx.l -a
```

or

To produce a printout so you can compare all the cgfx commands on the system disk with all the cgfx commands in the manual:

```
 OS9: rdump cgfx.l -a >/p
```

The information in the dump you are interested in has the exact spelling of the command in question, including uppercase, lowercase, underscores, etc. as listed under "global symbols defined." If the spelling from the dump differs from the manual, use the spelling from the dump and note the correct command in the manual. Otherwise you will find gfx functions to be as simple, straightforward and useful as their BASIC09 counterparts.

These functions are fundamental to graphics programming in the C language. CoCo users are fortunate that Tandy chose an industry standard, state-of-the-art operating system, languages and powerful features such as cgfx for the Color Computer. You don't want to do without them.

Numa David, an architect and planning consultant, uses a CoCo 3 to process demographic and other data into graphic output for feasibility studies for contemplated real-estate development projects. He is currently writing a full-featured CAD (Computer-Aided Design > application in the C language for the CoCo 3.

(Questions or comments concerning this article may be addressed to the author at 5305 Grand Lake. Bellaire. TX 77401; (713) 664-9529. Please enclose an SASE when requesting a reply.)