**By Or Idgar**

on windows

----------

1. install dosbox

2. dosbox - mount directory with turbo c and turbo assembler (drive c) (add to dosbox configuration)

3. dosbox - also mount the rml code directory to dosbox (drive d)

4. from here you can compile the kernel code without the boot loader.

the code is compiled with the linkbin.bat

d:

cd \obj

..\LINKBIN.BAT

5. linux machine on virtualbox with shared folder to rml code

6. install build-essential and nasm packages to compile the boot loader.

7. from here you can compile and prepare the floppy file.

just run make in the root directory of the code.

**By Boris Shpilyuck**

**1.**

**Source code**

The RML code is placed to C:\rml\rmLinux16 (location is arbitrary)

**2.**

**Tools for compiling and linking**

Download tcc, tasm and tlink to C:\rml\tcasm (location is arbitrary). The folder should contain tasm and tc folders that nave all the needed binaries.

**3.**

**Dosbox – environment where the C and tasm code is compiled and linked**

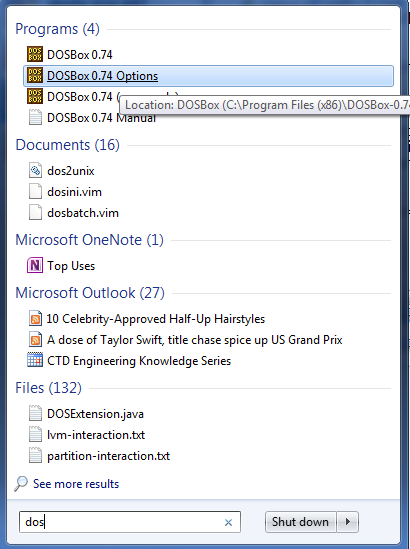
3.1

Download and install Dosbox on Windows (the following is used now DOSBox0.74-win32-installer.exe)

3.2

Configure Dosbox

Go to Start menu on Windows and select DosBox0.74 Options



Add the following to the end of the opened file ----

#####################################################

mount c: C:\rml\tcasm

mount d: C:\rml\rmLinux16

set path=%path%;c:\tc\bin;c:\tasm\bin

D:

cd obj

####################################################

So within the DosBox the drive c is mapped to C:\rml\tcasm on Windows. This folder on my local Windows contains all the needed tools for compiling and linking – tcc, tasm, tlink.

Drive d is mapped to C:\rml\rmLinux16 – this is the directory that contains all the source code RML.

**4.**

**Compiling and linking the code**

In the DosBox change one level up to in directory tree and run the script that

will do all the work ---

..\LINKBIN.BAT

The result is placed in C:\rml\rmLinux16\obj

**5.**

**Creating a final binary file**

The file will contain all the parts of the program – kernel, init, shell etc.

5.1

Prepare VirtualBox Ubuntu VM with C:\rml\rmLinux16 shared folder. So this VM will have an access to all the RML code and scripts.

Run the following in order to execute instructions in the C:\rml\rmLinux16\Makefile ---make

5.2

The result of “make” is C:\rml\rmLinux16\obj\disk1.img

**6.**

**Debugging with Bochs**

6.1

Download and install Bochs-2.6.8.exe

6.2

The following file has the required configuration to run and debug the prepared disk1.img - C:\rml\rmLinux16\bochsrc2.bxrc. Just double click it to start a debugger.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The file content specifies the location of the binary, size, log etc..

megs: 32

floppya: 1\_44=.\obj\disk1.img, status=inserted

boot: a

log: c:\bochsout.txt

mouse: enabled=0

display\_library: win32

debug\_symbols: file=bochs\_debug.sym