Patching DOS 5 and up for the PCjr

November 29th, 2020
mbbrutman@gmail.com

[Background]

[Patching DOS 5+]
- [Theory of operation]
- [Preparing a DOS 5 disk]
- [Patching the DOS 5+ disk]
- [Create FORMATJR.COM]

[Hard drive installation]
- [Fdisk and create a DOS partition]
- [Format the hard drive partition]

[Appendix A - Wiping out a boot sector]

Background

The PCjr shipped in 1983 in two configurations: a 64KB machine with no floppy disk drive or a 128KB machine with a single floppy disk drive. The architecture of the machine has the video buffer “borrow” memory from the main memory of the machine. With the standard 16KB video buffer this makes the available RAM either 48KB or 112KB. IBM never offered a hard drive solution.

Adding extra memory to a PCjr became possible, but it required a device driver to be loaded at boot time. Microsoft, Tecmar, and other vendors provided device drivers with their memory expansions. The best of the device drivers was a shareware program called jrConfig written by Larry Newcomb. jrConfig can be downloaded from http://www.brutman.com/PCjr/pcjr_downloads.html.

The original version of DOS that shipped with the PCjr was PC DOS 2.1. DOS versions 2.1 through 3.3 work on the machine as is. To run DOS 5 or later two things are required:

- Extra memory is required. While extra memory is useful for DOS 2.1 - 3.3, DOS 5 won’t even boot on a small system. [IBM formally states the requirement is 512KB and the PCjr is not supported.]
- DOS 5 can be patched to work on a PCjr even though it is not supported.

The first half of this document describes the patching procedure for DOS 5. Specifically I use IBM PC DOS 5.02. Other versions of DOS will be similar and can use this procedure, but the offsets used for the patch may be different.

The second half of this document describes the procedure for installing DOS onto a hard drive. Aside from the patching and PCjr specific memory management device driver required, installing DOS onto a hard drive on a PCjr uses the same procedures as on any other PC.

While I am using PC DOS 5.02, the same general procedure applies to any version of DOS. We are going to make a small patch to the boot sector to allow DOS to boot on the machine. If you are using a different version of DOS things may look slightly different, but the general idea is the same.

Patching DOS 5+

Theory of operation

A PCjr can run DOS 2.1 - DOS 3.3 with no modifications. To recognize extra memory on the system above 128KB a memory management device driver must be loaded using CONFIG.SYS. The device driver optionally relocates the region of memory used by the video hardware and thentells DOS to load above that address, giving DOS a contiguous block of memory to work with. DOS requires a contiguous block of memory; it can't "jump over" a hole in the memory map, thus the need to tell DOS to load above the hole.

DOS 5 represents a "catch-22" situation; it requires more memory than a PCjr has available, but it needs it fairly early in the boot process before it is possible to load device drivers. To get around this problem we are going to alter the early boot code so that it thinks that more memory is available, letting it boot far enough to load device drivers. Once it gets that far we can use the standard memory management device driver to fix the PCjr specific memory map problem and have DOS load above the video buffer.

The BIOS data area of the PCjr has a memory location where it reports how much memory is installed on the machine. That memory location (0:413) never reports more than 112KB because the video buffer consumes the memory from 112KB to 128KB. A different memory location (0:415) has the total amount of memory on the machine, but DOS doesn't look at that location. So the patch is very simple - just copy the two bytes that are at 0:415 to 0:413 early in the boot process before DOS checks to see how much memory is available.

On a patched system DOS will be able to load far enough to run device drivers, which are required for the PCjr.

Preparing a DOS 5 disk

If you do not have a DOS 5 machine then please get somebody to create a DOS 5 disk for you following these instructions. They don't need a PCjr; they are just going to provide you with a plain DOS 5 disk that you can modify for use with the PCjr

To create a suitable DOS 5 disk use a machine running DOS 5 and do the following:

- Format the disk using

  ```
  FORMAT A: /F:360 /S /V
  ```

  That will format the disk and make it bootable by copying the DOS 5 COMMAND.COM and hidden files to it. If your 360KB floppy drive is drive B: then adjust that command line accordingly.

- Copy the following files to the new disk: DEBUG.COM, FDISK.COM, FORMAT.COM, JRCONFIG.SYS.
- Do not create CONFIG.SYS yet.

After creating the disk try to boot the machine (not the PCjr) from it. This is to ensure that the disk is bootable before we move to the PCjr.

Patching the DOS 5+ disk

You can do this on the other machine or on the PCjr - it doesn't matter, the procedure is the same. I'm going to do it on the PCjr in these instructions.

- Boot the PCjr using its DOS (DOS 2.1 to DOS 3.3) disk. You just need something that works enough to run DEBUG.COM.
- Run DEBUG.COM from the disk you booted from.
- After DEBUG.COM is loaded remove your DOS boot disk and put the unpatched DOS 5 disk in.
- Run the following debug command to load the boot sector into memory: `L 0 0 0 1`
    - The first 0 is the address to load the boot sector into.
    - The second 0 is the drive number. Use 0 for A:, 1 for B. (On a PCjr you are probably using A:)
    - The third 0 is the sector number. This is a relative sector number so it starts at 0.
    - The 1 is the number of sectors to read.
- Use the following to check the first three bytes of the sector you loaded: `U 0 l 3`
    - That says to unassemble starting at address 0 and just show three bytes worth of instructions.
    - The expected output for this version of DOS is

      ```
      08F8:0000 EB3C          JMP 003E
      08F8:0002 90                  NOP
      ```

      Note that the segment value will change depending on your system but the offsets (0000 and 0002) should remain the same. Your version of DOS might have a slightly different target for the JMP instruction; the important thing is to note the address that the JMP instruction goes to.
- If you run the `D 0 l 10` command you will see the following:

  ```
  08F8:0000 EB 3C 90 49 42 4D 20 20-35 2E 30 00 02 02 01 00 .<.IBM  5.0.....
  ```

  The first two bytes are the JMP instruction, the next byte is a NOP instruction, and the next 8 bytes are the "OEM System Name". After that the BIOS parameter block for the disk starts.
- Somewhere around offset 0x1A0 there is going to be a string that says "Non-System disk or disk error". This is the error message that gets printed when the disk is not bootable. We are going to borrow a part of that string and replace it with code to set the PCjr BIOS memory area correctly so that DOS 5 will boot on the machine. You can find where that string starts with the following command:

  ```
  S 0 200 "Non-System Disk"
  08F8:01A0
  ```

  For my copy of DOS the offset the string is found at is 01A0. Looking at the memory there you will see the following:

  ```
  D 01A0 L 30
  08F8:01A0 4E 6F 6E 2D 53 79 73 74-65 6D 20 64 69 73 6B 20 Non-System disk
  08F8:01B0 6F 72 20 64 69 73 6B 20-65 72 72 6F 72 0D 0A 52 or disk error..R
  08F8:01C0 65 70 6C 61 63 65 20 61-6E 64 20 70 72 65 73 73 eplace and press
  ```

- This is where the patch code will go. We will leave the first part of the string alone and use the area reserved for the second sentence. The two sentences of the message are separated by a carriage return (0x0D) and line feed (0x0A). If we replace the ‘R’ with a NUL character (0x00) the important part of the error message is preserved and then we can use the remaining part for the patch code:

```
E 01BF 0            // Debug command to replace the ‘R’ with a NUL character
A 01C0              // Debug command to start assembling code at offset 01C0
PUSH DS             // Patch code: Save DS register
MOV AX,0            // Patch code: Set AX register to 0
MOV DS,AX           // Patch code: Copy AX register to DS register (setting DS to 0)
MOV AX,[415]        // Patch code: Move the contents of address 0x0415 to AX register
MOV [413],AX        // Patch code: Move AX register to memory address 0x413
POP DS              // Patch code: Restore DS register
JMP 3E              // Patch code: Continue the boot process
```

  <hit enter on a blank line to stop the Assemble command>

  The offset of that last JMP instruction should be the same offset as the first JMP instruction we examined at offset 0. Substitute in the correct value for your version of DOS if it is different than what we are using here.

- Now the patch code is in memory. Verify that you entered it correctly by using the `U 01C0` command. (U is the “unassemble” command; it will show you more instructions than you entered, which is fine.)
- Remember the original JMP instruction at offset 0 that we first looked at? It needs to be altered to jump to this new code now:

```
A 0
JMP 1C0
```

  <hit enter on a blank line>

  The offset for the JMP instruction should be the same offset that you placed the patch code at, which is one byte after the NUL char that replaced the ‘R’ of “Replace.” On my disk the NUL went at offset 0x1BF and the patch code started at 0x1C0. If your disk is different then make the necessary adjustments.
- The alterations are complete in memory. Let’s write it to the disk now:

```
W 0 0 0 1
```

  Just like the Load command, the second 0 is the drive letter. Use 1 if you used drive B.
- Quit debug.com with the Q command: `Q`

While the disk is patched, it is still not ready for the PCjr. We need to create a CONFIG.SYS file on the disk to set an important parameter and to load the JRCONFIG.SYS device driver:

```
COPY CON: CONFIG.SYS
STACKS=0,0
DEVICE=JRCONFIG.SYS
```

<Hit Ctrl-Z to write the file to disk>

JRCONFIG.SYS is the memory manager required for recognizing extra memory on a PCjr above 128KB. The STACKS directive is required to disable interrupt switching in DOS. (Something about interrupt switching is incompatible with the PCjr.)

Now you can reboot the PCjr and watch it boot from DOS 5!

When using JRCONFIG.SYS the machine will appear to boot twice. It actually is, and this is expected. During the first boot JRCONFIG.SYS fixes up the PCjr video memory hole and tells DOS where to load again. During the second boot DOS loads as expected.

Create FORMATJR.COM

FORMAT.COM has a copy of the boot sector in it so disks prepared with FORMAT.COM are not compatible with the PCjr. This means every time you want a bootable disk you will have to do the patch procedure, and that gets old quickly.

To make things more convenient we are going to copy FORMAT.COM to FORMATJR.COM and then patch the boot sector that resides inside of FORMATJR.COM. You will be able to use FORMAT.COM to create a plain DOS 5 bootable disk for other machines and FORMATJR.COM to create bootable disks for the PCjr.

We can't use the exact same procedure as before because DEBUG.COM will try to be clever and convert our relative jumps, which are now in a different place. So we will just copy the raw bytes from the first patch we made and skip the fancy assembler in DEBUG.COM that will mess up the offsets.

Assuming you are still booted from DOS 5 and the disk is still in the drive …

- First, get the bytes we need from the boot sector of the patched DOS 5 disk:

  ```
  DEBUG.COM
  L 0 0 0 1
  D 0 L 3
  E9 BD 01
  D 1BF L 11
  08F8:01BF                                              00                .
  08F8:01C0 1E B8 00 00 8E D8 A1 15-04 A3 13 04 1F E9 6E FE .............n.
  Q
  ```

  If your offsets and values are different because you started with a different version of DOS, write them down somewhere.

- Create FORMATJR.COM: `COPY FORMAT.COM FORMATJR.COM`
- Note the file size. On this version of DOS it is 34415 bytes, or 0x866F in hexadecimal.

- Patch FORMATJR.COM:

  ```
  DEBUG.COM FORMATJR.COM
  S 100 L 866F EB 3C 90
  291D:0D7D
  ```

  The S command searches memory for the three byte sequence EB 3C 90, which is the original three bytes from location 0 of the boot sector. If the first three bytes of your original boot sector were different search for those instead. On my system inside of FORMATJR.COM those bytes are found at offset 0x0D7D.
- Confirm that this location is correct by dumping the bytes. You should see the same original JMP instruction bytes, the NOP instruction bytes, and the OEM System Name bytes:

  ```
  D 0D7D L 10
  291D:0D7D                                        EB 3C 90                .<.
  291D:0D80 49 42 4D 20 20 35 2e 30-00 02 08 01 00          IBM  5.0.....
  ```

  Note that the bytes after the "IBM 5.0" might be different than what was in the boot sector of the disk. This is because those bytes will vary depending on the type of disk format being used. We don't care about anything except the JMP instruction so differences here are not an issue.

- Confirm the patch location is the same. Do this by taking the offset you just found (0x0D7D in my example) and adding 0x01BF to it, resulting in offset 0x0F3C. Dumping that address should show you the string that we are going to modify to create the patch:

  ```
  D 0F3C L 10
  291D:0F3C                                     52 65 70 6C             Repl
  291D:0F40 61 63 65 20 61 6E 64 20-70 72 65 73             ace and pres
  ```

  Remember, we replaced the 'R' with a NULL character and put code across the rest of the string.

- Assuming everything lines up make the patches:

  ```
  E 0D7D E9 BD 01      // Replace the JMP instruction at the start of the boot sector
  // Replace the 'R' with a NUL character and enter the same bytes for the patch.
  E 0F3C 00 1E B8 00 00 8E D8 A1 15 04 A3 13 04 1F E9 6E FE
  W                    // Write the changes to FORMATJR.COM
  Q                    // Quit debug
  ```

Remember, use FORMAT.COM to create disks for normal machines and FORMATJR.COM to create disks for a PCjr. The good news is that now that FORMATJR.COM has a patched boot sector inside of it, you will never need to patch a disk again.

SYS.COM also has a copy of the boot sector inside of it. If you wanted to be thorough you could use the same process to create SYSJR.COM and patch it. SYS.COM is rarely used, so if the need comes up you can use it as-is and then patch the resulting boot disk.

Hard drive installation

Fdisk and create a DOS partition

Using FDISK.COM from the DOS 5 disk:

- Select menu option 4 to see what partitions are already defined.
- DOS will let you delete DOS partitions but not other partition types. If you need to, use the program in Appendix A to wipe out the boot sector of the drive, then run FDISK.COM again to partition the drive.
- Select menu option 1 to create a primary DOS partition
  - Choose a reasonable size; PCjrs are slow and DOS 5 can take forever to compute the free space on a partition. Over 100MB and things will be pretty slow.
  - Be patient - FDISK.COM can be slow while it examines the hard drive
- Select menu option 2 to make the newly created DOS partition active. If you miss that step you will not be able to boot from it.

Fdisk will want the system to reboot. Boot from the floppy drive again, because the hard drive only has a partition now. It still needs to be formatted and have DOS transferred to it.

Format the hard drive partition

Using your bootable DOS 5 floppy disk run the FORMATJR.COM command to format the new hard drive partition:

```
FORMATJR C: /S /V
```

Remember that FORMATJR.COM has a copy of a patched boot sector inside of it. If you use it you do not need to patch the boot sector of the hard drive in a separate step. If you forget and use the regular FORMAT.COM command you will have to patch the boot sector of the hard drive, just like you patched the boot sector of the floppy disk.

Hopefully you chose a reasonably sized partition. Otherwise, prepare to wait.

At this point DOS can see drive C. We just need to copy JRCONFIG.SYS to drive C, create a CONFIG.SYS file, and it should be ready to boot:

```
COPY JRCONFIG.SYS C:
COPY CON: C:CONFIG.SYS
STACKS=0,0
DEVICE=JRCONFIG.SYS
FILES 15
BUFFERS 30
```
<Use Ctrl-Z here to mark the end of file and have it written to C:>

After that, you can remove the floppy disk and try booting from the hard drive.

Appendix A - Wiping out a boot sector

If FDISK.COM refuses to display partition data it might be because it does not recognize what is in the first sector of the drive. It is also possible that FDISK.COM can see everything but it will refuse to allow you to delete partitions because a non-DOS partition is on the drive.

If either of these situations arises you can wipe the boot sector clean, which will make FDISK.COM think it is working on an empty hard drive. If you do this you will not be able to recover any data on the hard drive, so proceed with caution. The program is annotated so you can see exactly what it is doing.

`DEBUG`	Start DEBUG.COM
`F 200 L 200 0`	Fill location 200 with 512 bytes of zeros
`A CS:100`	Assemble a program starting at location 100
`MOV AX,0301`	Setup for Int 13h - Function is AH=03 (Write sectors), AL=01 (sectors to write)
`MOV BX,0200`	BX will have the offset to the buffer full of zeros
`MOV CX,1`	Track/cylinder (CH) is 0, Sector (CL) is 1. (Sectors always start with 1, not zero)
`MOV DX,80`	Select the first hard disk. Use 81 for the second hard disk
`INT 13`	Call BIOS interrupt 13h
`INT 20`	Return to DOS using interrupt 20h
	No input … just press enter to tell DEBUG you are done entering the program
`G`	Tell debug to run the program

When you run the program the hard drive should have one sector of zeros written to the boot sector. After this you should be able to run FDISK.COM and not encounter any resistance.