# lDOS boot documentation



This document has been compiled on 2022-11-07.

# Contents

# Section 1: lDOS boot protocols

## 1.1 Sector to iniload protocol

The iniload kernel is loaded to an arbitrary segment. The segment must be at least 60h. Common choices are 60h, 70h, and 200h. At least 1536 bytes of the file must be loaded. Current loaders will load at least 8192 bytes if the file is as large or larger than that. The entrypoint is found by applying no segment adjustment (0) and choosing the offset 400h (1024).

### 1.1.1 Signatures

At offset 1020 (3FCh) there is the signature '`lD`'. Behind that there are two bytes with printable non-blank ASCII codepoints. Currently the following signatures are defined:

'`lDOS`'

lDOS kernel (not yet in use)

'`lDRx`'

RxDOS kernel

'`lDFD`'

FreeDOS kernel wrapped in iniload (fdkernpl.asm)

'`lDeb`'

lDebug

'`lDDb`'

lDDebug (debuggable lDebug)

'`lDbC`'

lCDebug (conditionally debuggable lDebug)

'`lDTP`'

lDOS test payload kernel (testpl.asm)

'`lDTW`'

lDOS test result writer kernel (testwrit.asm)

### 1.1.2 Load Stack Variables (LSV)

Under this protocol, the pointer '`ss:bp`' is passed. It points to a boot sector with (E)BPB. The

stack pointer must be at most ‘`bp - 10h`’. Below the pointed to location there live the Load Stack Variables. These follow this structure:

```
        struc LOADSTACKVARS, -10h
lsvFirstCluster:        resd 1
lsvFATSector:           resd 1
lsvFATSeg:              resw 1
lsvLoadSeg:             resw 1
lsvDataStart:           resd 1
        endstruc
```

lsvFirstCluster

(FAT12, FAT16) Low word gives starting cluster of file. High word uninitialised.

(FAT32) Dword gives starting cluster of file.

(else) Should be zero.

lsvFATSector

(FAT16) Low word gives loaded sector-in-FAT. -1 if none loaded yet. High word uninitialised.

(FAT32) Dword gives loaded sector-in-FAT. -1 if none loaded yet.

(FAT12, else) Unused.

lsvFATSeg

(FAT16, FAT32) Word gives segment of FAT buffer if word/dword [lsvFATSector] != -1.

(FAT12) Word gives segment of FAT buffer. Zero if none. Otherwise, buffer holds entire FAT data, up to 6 KiB.

lsvLoadSeg

Word points to segment beyond last loaded paragraph. Allows iniload to determine how much of it is already loaded.

lsvDataStart

Dword gives sector-in-partition of first cluster's data.

An LSV extension allows to pass a command line to the kernel. The stack pointer must be at most ‘`bp - 114h`’ then. This follows the structure like this:

```
lsvclSignature          equ "CL"
lsvclBufferLength       equ 256

        struc LSVCMDLINE, LOADSTACKVARS - lsvclBufferLength - 4
lsvCommandLine:
.start:         resb lsvclBufferLength
.signature:     resw 1
lsvExtra:       resw 1
        endstruc
```

lsvCommandLine.start

Command line buffer. Contains zero-terminated command line string.

lsvCommandLine.signature

Contains the signature value '`CL`' if command line is given.

lsvExtra

Used internally by iniload. Space for this must be reserved when passing a command line.

If no command line is passed then either the stack pointer must be '`bp - 10h`', or '`bp - 12h`', or the word in the lsvCommandLine.signature variable (`word [ss:bp - 14h]`) must not equal the string '`CL`'.

- dosemu2's RxDOS.3 support sets '`sp = bp - 10h`'
- ldosboot boot.asm (FAT12/FAT16) loader uses the variable for a 'paragraphs per sector' value which is always a power of two and always below-or-equal 200h.
- ldosboot boot32.asm (FAT32) loader uses the variable for an 'entries per sector' value which is always a power of two and always below-or-equal 100h.
- lDebug with protocol options `cmdline=0 push_dpt=0` sets '`sp = bp - 10h`'

### 1.1.3 Memory map

The initial loader part that is loaded must be loaded at above or equal to linear 00600h. The FAT buffer segment (if used) must also be stored at above or equal to linear 00600h. The stack (which should extend at least 512 bytes below '`ss:bp`') and boot sector (pointed to by '`ss:bp`', at least 512 bytes length) should also be stored at above or equal to linear 00600h.

There is an additional memory area, the Low Memory Area top reservation, which should be unused by the load protocol at handoff time but be at least 20 KiB in size. It is located below the usable Low Memory Area top. That is, directly below the EBDA, RPL-reserved memory, video memory, or otherwise UMA. This area is reserved in order to facilitate initial loader operation.

None of the memory areas may overlap. This does not include the FAT buffer in case it is uninitialised.

### 1.1.4 Load filename in the boot sector

The boot sector area may be expected to contain a valid 8.3 format (blank-padded FCB) filename behind the (E)BPB. This name should not contain blanks other than trailing in the file name portion or trailing in the file extension portion. It should consist of printable ASCII codepoints. That is, byte values between 20h and 7Eh inclusive. It should not consist of eleven times the same byte value. Additional FAT Short File Name restrictions may be assumed.

Although a loader should not depend on this for crucial operation, it may want to detect the kernel name it was presumably loaded from for informational or optional purposes. The canonical implementation of this is currently the function '`findname`' in the `testpl.asm` test payload kernel. It is found within the ldosboot repo. This handling is based on the function of the same name in the instsect application.

## 1.2 Iniload to payload protocol

The payload is loaded to an arbitrary segment. The segment must be at least 60h. The entire payload must be loaded. The size of the payload is determined at iniload build time. The entrypoint is found by applying a segment adjustment and choosing an offset. The segment adjustment is specified at iniload build time by the numeric define `_EXEC_SEGMENT` (default 0), and the offset by the define `_EXEC_OFFSET` (default 0).

### 1.2.1 Extended BIO Parameter Block (EBPB)

Above the LSV, `ss:bp` points to an EBPB and surrrounding boot sector. Note that this is always a FAT32-style EBPB. If the filesystem that is loaded from is not FAT32, and is therefore FAT16 or FAT12, then the FAT16/FAT12 BPBN structure is moved up. It is placed where the FAT32 BPBN is usually expected. In this case, the entire boot sector contents behind the BPBN are also moved up by the size of the FAT32-specific fields. The FAT32-specific fields are filled with zeros, except for the FAT32 'sectors per FAT' field. It is filled with the contents of the FAT16/FAT12 'sectors per FAT' field.

### 1.2.2 Load Stack Variables (LSV)

Refer to section 1.1.2.

### 1.2.3 Load Data 1 (LD)

Below the LSV, iniload passes the LOADDATA (1) structure.

```
        struc LOADDATA, LOADSTACKVARS - 10h
ldMemoryTop:    resw 1
ldLoadTop:      resw 1
ldSectorSeg:    resw 1
ldFATType:      resb 1
ldHasLBA:       resb 1
ldClusterSize:  resw 1
ldParaPerSector:resw 1
ldLoadingSeg:   resw 1
ldLoadUntilSeg: resw 1
        endstruc
```

ldMemoryTop

> Word. Segment pointer to behind usable memory. Points at the first of the EBDA, RPL-reserved memory, or video memory or otherwise UMA. Indicates how much memory may be used by a typical kernel. (lDebug detects the EBDA to move that below where it installs.)

ldLoadTop

> Word. Segment pointer to lowest lDOS boot memory in use. All memory between linear 600h and the segment indicated here is usable by the payload. Only the payload itself is stored in this area. The other buffers, stack, and structures passed by iniload must live above this segment.

ldSectorSeg

> Word. Segment pointer to an 8 KiB transfer buffer. It is insured that this buffer does not cross

a 64 KiB boundary. This may be needed by some disk units. The buffer is not initialised to anything generally.

ldFATType

Byte. Indicates length of FAT entry in bits. 12 indicates FAT12, 16 FAT16, 32 FAT32. It is planned to allow zero for non-FAT filesystems.

ldHasLBA

Byte. Only least significant bit used. Bit on indicates LBA extensions available for the load disk unit. Bit off indicates LBA extensions not available.

ldClusterSize

Word. Contains amount of sectors per cluster. Unlike the byte field for the same purpose in the BPB, this field can encode 256 (EDR-DOS compatible) without any masking. May be given as zero for non-FAT filesystems.

ldParaPerSector

Word. Contains amount of paragraphs per sector. Must be a power of two between 2 (32 B/s) and 200h (8192 B/s). May be given as zero for non-FAT filesystems.

ldLoadingSeg

Word. Internally used by iniload. Available for re-use by payload.

ldLoadUntilSeg

Word. Internally used by iniload. Available for re-use by payload.

### 1.2.4 Load Command Line (LCL)

Below the LOADDATA structure, iniload passes the LOADCMDLINE structure.

```
lsvclBufferLength       equ 256

        struc LOADCMDLINE, LOADDATA - lsvclBufferLength
ldCommandLine:
.start:         resb lsvclBufferLength
        endstruc
```

This buffer is always initialised to an ASCIZ string. At most 255 bytes may be initialised to string data. At most the 256th byte is a zero.

If the first word of the buffer is equal to 0FF00h, that is there is an empty command line the terminator of which is followed by a byte with the value 0FFh, then no command line was passed to iniload. Currently lDebug can pass a command line to iniload when loading with its lDOS, RxDOS.2, RxDOS.3, or FreeDOS protocols. When iniload is loaded as a Multiboot1 or Multiboot2 specification kernel, it is also assumed that a command line can be passed.

# Source Control Revision ID

hg e88f6a6424a3, from commit on at 2022-11-07 20:28:44 +0100

If this is in ecm's repository, you can find it at https://hg.pushbx.org/ecm/ldosboot/rev/e88f6a6424a3