Basic MBR Partitions

A common type of

PC-based Partitions

Carrier: Chapter 5

Additional Info

Disk Organization

There are, today, two different types of disk organization

Basic MBR

A partition schema that began in the early 1980s

Basic MBR is still the most widely used

GPT Based

A partition schema by Intel that began in 2000

Eliminates some of the limitations of Basic MBR

Originally used on servers

Now increasingly used on client systems

Basic MBR Partition Overview

PC-based Partitions refer to any of several mass storage partition schemes used on personal computers

Basic MBR Partitions

Used on many servers and clients

BIOS code is needed

GPT Partitions

*Requires EFI code (**E**xtensible **F**irmware **I**nterface) instead of BIOS code*

More on GPT later

Basic MBR Partition Overview

Basic MBR Partitions are used on

Hard disks, diskettes

Some but not all thumb drives, CDs, DVDs, ZIP…

Often referred to as "***DOS partitions***"

Carrier uses this term

There is no "official" standard or specification

OSs that use *Basic MBR Partitions* include

MSDOS, Windows, Linux, FreeBSD, Open BSD and others

Basic MBR Partition Overview

Used since the beginnings of IA32 systems in the early 1980s

Originally designed for the needs of PCs at that time

Small disks

Few partitions

Has been modified to accommodate today's disks

Much larger disks

Need for many partitions

This has resulted in MBR partitions becoming a somewhat convoluted partitioning scheme

Basic MBR Partitions

This slide set focuses on

Basic, not Dynamic partitions

MBR, not GPT or other partitions

Thus the PC-based partitions that we will discuss here are ***Basic MBR Partitions***

However much of it also applies to GPT

Basic vs. Dynamic Partitions

Basic partitions are those that consist of contiguous sectors on a single disk

Dynamic partitions are those that can be comprised of smaller non-contiguous partitions

Basic MBR Partition Organization

DOS Partition Organization

Maximum Number of Partitions

A *Basic MBR Partition* can have up to **4** ***Primary Partitions***

- *Limited by the fact that there is space for 4 partition specifications in the MBR Partition Table*
- *This became a problem as disks increased in capacity and the need for > 4 partitions grew*
- *The original schema had to be modified to increase the number of partitions*
- *This led to much of the schema's complexity*
  - More on this later

Master Boot Record (MBR)

All Basic MBR Partitions have a ***Master Boot Record*** (MBR) in the first 512-byte sector of the volume

- *Sometimes called the **Volume MBR***
- *Located in the first sector of the volume (LBA sector 0)*

The MBR contains for the volume

- *Boot code*
- *Partition Table*
- *Signature value (0x55AA)*

Format of MBR

Boot code

Bytes 00 – 445 of the MBR (0x00 – 0x01BD)

Code can extend into sectors 1 - 62 as needed

Processes the partition table

Locates the partition

Partition Table

Bytes 446 – 509 (0x01BE – 0x05E5)

Defines up to 4 partitions

Signature value

Bytes 510 – 0511 (last 2 bytes) (0x01FE – 0x01FF)

Open your Rng WinHex18.9 Shortcut

Go to your Rng 538 desktop

Open your WinHex19.8 shortcut (as administrator) that is on your desktop

If the WinHex 19.8 is not on your desktop

Go to R:\Tools\

Copy the WinHex shortcut to your RADISH desktop

Delete any file or drive images that your WinHex has open

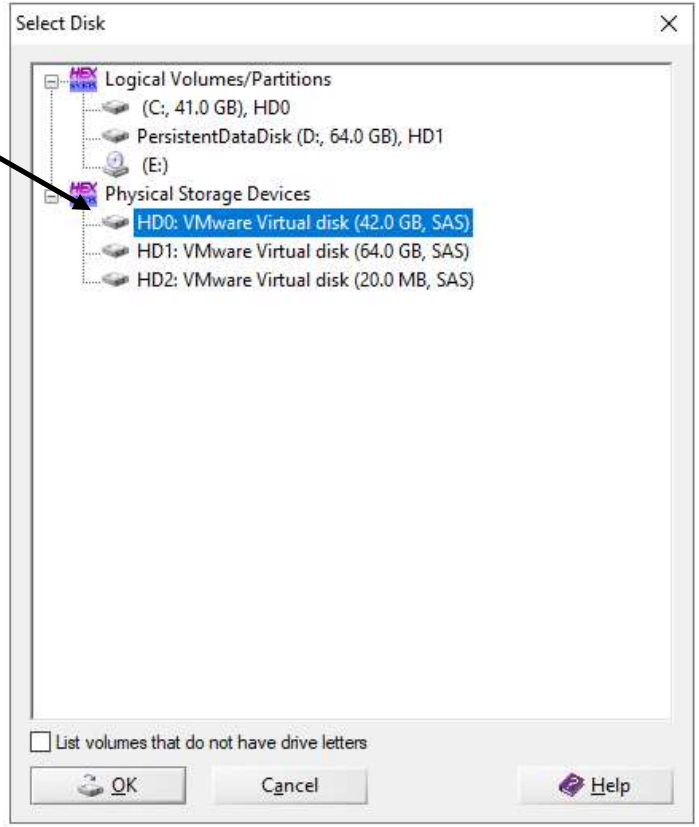
Right click on the tab and delete

Look at Your RADISH Hard Disk

Open your hard disk

Tools | Open Disk…

What do you choose?



Look at Your Hard Disk HD0 MBR

Look at the 1st sector of your hard disk (512 bytes)

The 1st sector contains the MBR

MBR

Partition Table and Signature Indicated

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000000	33	C0	8E	D0	BC	00	7C	8E	C0	8E	D8	BE	00	7C	BF	00	3ÀŽÐ¼ |ŽÀŽØ¾ |¿
00000000016	06	B9	00	02	FC	F3	A4	50	68	1C	06	CB	FB	B9	04	00	¹ üó¤Ph ËÛ¹
00000000032	BD	BE	07	80	7E	00	00	7C	0B	0F	85	0E	01	83	C5	10	½¾ €~ | … ƒÅ
00000000048	E2	F1	CD	18	88	56	00	55	C6	46	11	05	C6	46	10	00	âñÍ ˆV UÆF ÆF
00000000064	B4	41	BB	AA	55	CD	13	5D	72	0F	81	FB	55	AA	75	09	´A»ªUÍ]r ûUªu
00000000080	F7	C1	01	00	74	03	FE	46	10	66	60	80	7E	10	00	74	÷Á t þF f`€~ t
00000000096	26	66	68	00	00	00	00	66	FF	76	08	68	00	00	68	00	&fh fÿv h h
00000000112	7C	68	01	00	68	10	00	B4	42	8A	56	00	8B	F4	CD	13	|h h ´BŠV ‹ôÍ
00000000128	9F	83	C4	10	9E	EB	14	B8	01	02	BB	00	7C	8A	56	00	ŸƒÄ žë ¸ » |ŠV
00000000144	8A	76	01	8A	4E	02	8A	6E	03	CD	13	66	61	73	1C	FE	Šv ŠN Šn Í fas þ
00000000160	4E	11	75	0C	80	7E	00	80	0F	84	8A	00	B2	80	EB	84	N u €~ € „Š ²€ë„
00000000176	55	32	E4	8A	56	00	CD	13	5D	EB	9E	81	3E	FE	7D	55	U2äŠV Í]ëž >þ}U
00000000192	AA	75	6E	FF	76	00	E8	8D	00	75	17	FA	B0	D1	E6	64	ªunÿv è u ú°Ñæd
00000000208	E8	83	00	B0	DF	E6	60	E8	7C	00	B0	FF	E6	64	E8	75	èƒ °ßæ`è| °ÿædèu
00000000224	00	FB	B8	00	BB	CD	1A	66	23	C0	75	3B	66	81	FB	54	û¸ »Í f#Àu;f ûT
00000000240	43	50	41	75	32	81	F9	02	01	72	2C	66	68	07	BB	00	CPAu2 ù r,fh »
00000000256	00	66	68	00	02	00	00	66	68	08	00	00	00	66	53	66	fh fh fSf
00000000272	53	66	55	66	68	00	00	00	00	66	68	00	7C	00	00	66	SfUfh fh | f
00000000288	61	68	00	00	07	CD	1A	5A	32	F6	EA	00	7C	00	00	CD	ah Í Z2öê | Í
00000000304	18	A0	B7	07	EB	08	A0	B6	07	EB	03	A0	B5	07	32	E4	· ë ¶ ë µ 2ä
00000000320	05	00	07	8B	F0	AC	3C	00	74	09	BB	07	00	B4	0E	CD	‹ð¬< t » ´ Í
00000000336	10	EB	F2	F4	EB	FD	2B	C9	E4	64	EB	00	24	02	E0	F8	ëòôëý+Éädë $ àø
00000000352	24	02	C3	49	6E	76	61	6C	69	64	20	70	61	72	74	69	$ ÃInvalid parti
00000000368	74	69	6F	6E	20	74	61	62	6C	65	00	45	72	72	6F	72	tion table Error
00000000384	20	6C	6F	61	64	69	6E	67	20	6F	70	65	72	61	74	69	loading operati
00000000400	6E	67	20	73	79	73	74	65	6D	00	4D	69	73	73	69	6E	ng system Missin
00000000416	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	g operating syst
00000000432	65	6D	00	00	00	63	7B	9A	88	44	8D	C2	00	00	80	20	em c{šˆD Â €
00000000448	21	00	07	DD	1E	3F	00	08	00	00	00	A0	0F	00	00	DD	! Ý ? Ý
00000000464	1F	3F	07	FE	FF	FF	00	A8	0F	00	AC	B2	1E	05	00	FE	? þÿÿ ¨ ¬² þ
00000000480	FF	FF	27	FE	FF	FF	00	60	2E	05	00	88	11	00	00	00	ÿÿ'þÿÿ `. ˆ
00000000496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA	Uª
00000000512	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000000528	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Sector 0 of 88,080,384 | Offset: 509 = 0 | Block: 446 - 509 | Size: 64

Hard disk 0
Model: VMware Virtual disk
Firmware Rev.: 2.0
Bus: SAS
[Read-only mode]

Total capacity: 42.0 GB
45,097,156,608 bytes
Bytes per sector: 512
Surplus sectors at end: 6144

Partition: <1
Relative sector No.: n/a

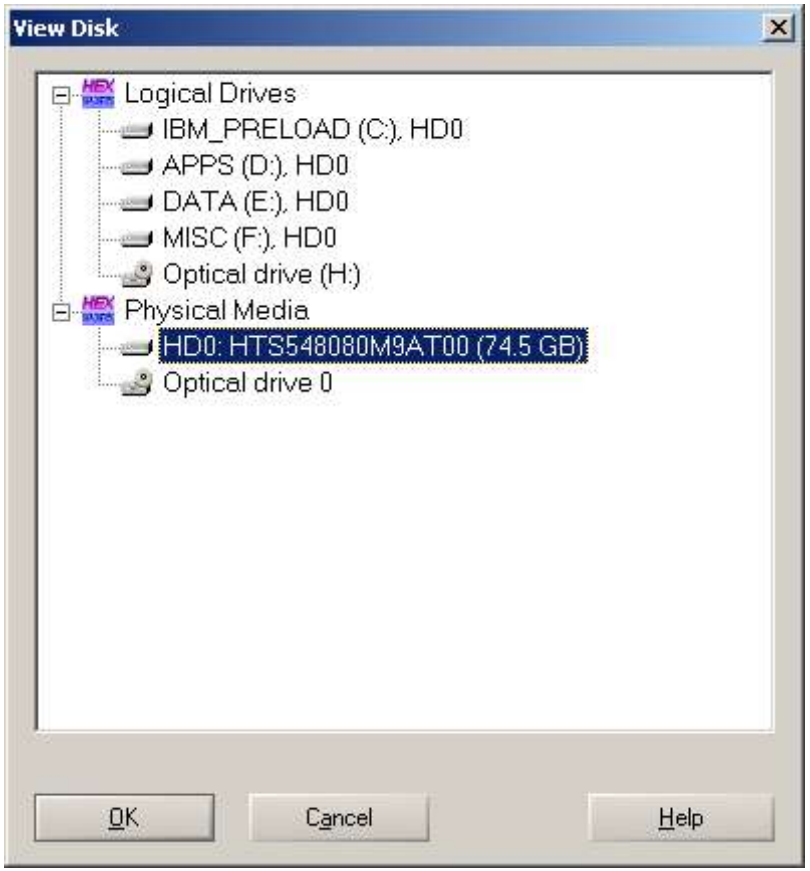
Mode: hexadecimal
Offsets: decimal
Bytes per page: 34x16=544

Window #: 1
No. of windows: 1

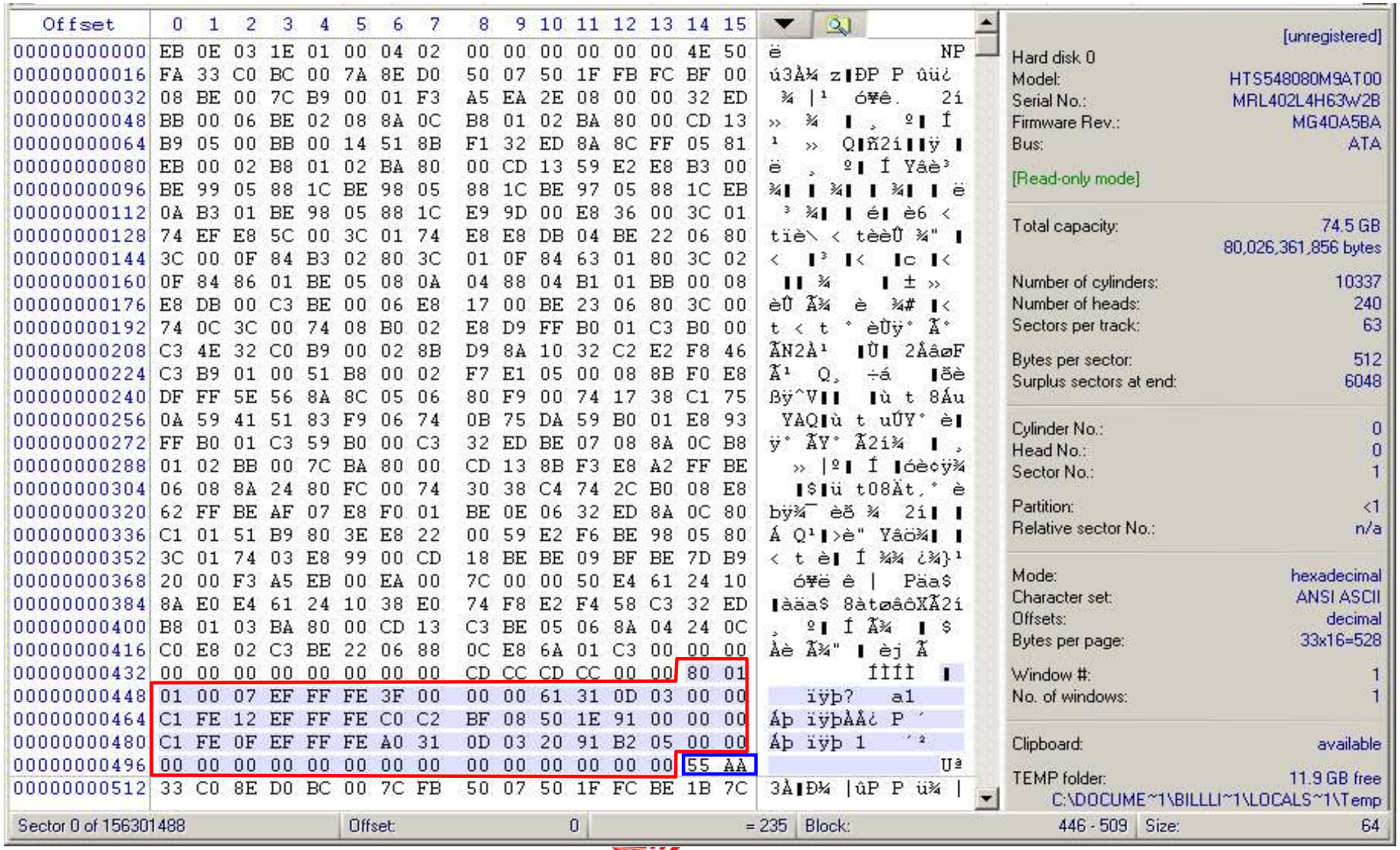
Clipboard: available
TEMP folder: 17.0 GB free
C:\Users\lidinsky\AppData\Local\Temp

Here's Another Hard Disk

More Like an MBR on Your Personal Computer



MBR
Partition Table and Signature Indicated



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00000000000	EB	0E	03	1E	01	00	04	02	00	00	00	00	00	00	4E	50
00000000016	FA	33	C0	BC	00	7A	8E	D0	50	07	50	1F	FB	FC	BF	00
00000000032	08	BE	00	7C	B9	00	01	F3	A5	EA	2E	08	00	00	32	ED
00000000048	BB	00	06	BE	02	08	8A	0C	B8	01	02	BA	80	00	CD	13
00000000064	B9	05	00	BB	00	14	51	8B	F1	32	ED	8A	8C	FF	05	81
00000000080	EB	00	02	B8	01	02	BA	80	00	CD	13	59	E2	E8	B3	00
00000000096	BE	99	05	88	1C	BE	98	05	88	1C	BE	97	05	88	1C	EB
00000000112	0A	B3	01	BE	98	05	88	1C	E9	9D	00	E8	36	00	3C	01
00000000128	74	EF	E8	5C	00	3C	01	74	E8	E8	DB	04	BE	22	06	80
00000000144	3C	00	0F	84	B3	02	80	3C	01	0F	84	63	01	80	3C	02
00000000160	0F	84	86	01	BE	05	08	0A	04	88	04	B1	01	BB	00	08
00000000176	E8	DB	00	C3	BE	00	06	E8	17	00	BE	23	06	80	3C	00
00000000192	74	0C	3C	00	74	08	B0	02	E8	D9	FF	B0	01	C3	B0	00
00000000208	C3	4E	32	C0	B9	00	02	8B	D9	8A	10	32	C2	E2	F8	46
00000000224	C3	B9	01	00	51	B8	00	02	F7	E1	05	00	08	8B	F0	E8
00000000240	DF	FF	5E	56	8A	8C	05	06	80	F9	00	74	17	38	C1	75
00000000256	0A	59	41	51	83	F9	06	74	0B	75	DA	59	B0	01	E8	93
00000000272	FF	B0	01	C3	59	B0	00	C3	32	ED	BE	07	08	8A	0C	B8
00000000288	01	02	BB	00	7C	BA	80	00	CD	13	8B	F3	E8	A2	FF	BE
00000000304	06	08	8A	24	80	FC	00	74	30	38	C4	74	2C	B0	08	E8
00000000320	62	FF	BE	AF	07	E8	F0	01	BE	0E	06	32	ED	8A	0C	80
00000000336	C1	01	51	B9	80	3E	E8	22	00	59	E2	F6	BE	98	05	80
00000000352	3C	01	74	03	E8	99	00	CD	18	BE	BE	09	BF	BE	7D	B9
00000000368	20	00	F3	A5	EB	00	EA	00	7C	00	00	50	E4	61	24	10
00000000384	8A	E0	E4	61	24	10	38	E0	74	F8	E2	F4	58	C3	32	ED
00000000400	B8	01	03	BA	80	00	CD	13	C3	BE	05	06	8A	04	24	0C
00000000416	C0	E8	02	C3	BE	22	06	88	0C	E8	6A	01	C3	00	00	00
00000000432	00	00	00	00	00	00	00	00	CD	CC	CD	CC	00	00	80	01
00000000448	01	00	07	EF	FF	FE	3F	00	00	00	61	31	0D	03	00	00
00000000464	C1	FE	12	EF	FF	FE	C0	C2	BF	08	50	1E	91	00	00	00
00000000480	C1	FE	0F	EF	FF	FE	A0	31	0D	03	20	91	B2	05	00	00
00000000496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA
00000000512	33	C0	8E	D0	BC	00	7C	FB	50	07	50	1F	FC	BE	1B	7C

[unregistered]
Hard disk 0
Model: HTS548080M9AT00
Serial No.: MRL402L4H63W2B
Firmware Rev.: MG4OA5BA
Bus: ATA
[Read-only mode]

Total capacity: 74.5 GB
80,026,361,856 bytes

Number of cylinders: 10337
Number of heads: 240
Sectors per track: 63

Bytes per sector: 512
Surplus sectors at end: 6048

Cylinder No.: 0
Head No.: 0
Sector No.: 1

Partition: <1
Relative sector No.: n/a

Mode: hexadecimal
Character set: ANSI ASCII
Offsets: decimal
Bytes per page: 33x16=528

Window #: 1
No. of windows: 1

Clipboard: available

TEMP folder: 11.9 GB free
C:\DOCUME~1\BILLLI~1\LOCALS~1\Temp

Sector 0 of 156301488 | Offset: 0 | = 235 | Block: 446 - 509 | Size: 64

MBR Partition Table

The MBR Partition Table has four entries

Each entry can describe a Basic MBR (i.e., DOS) partition

An entry that describes a DOS partition contains the following fields

- *Starting CHS sector address*
- *Ending CHS sector address*

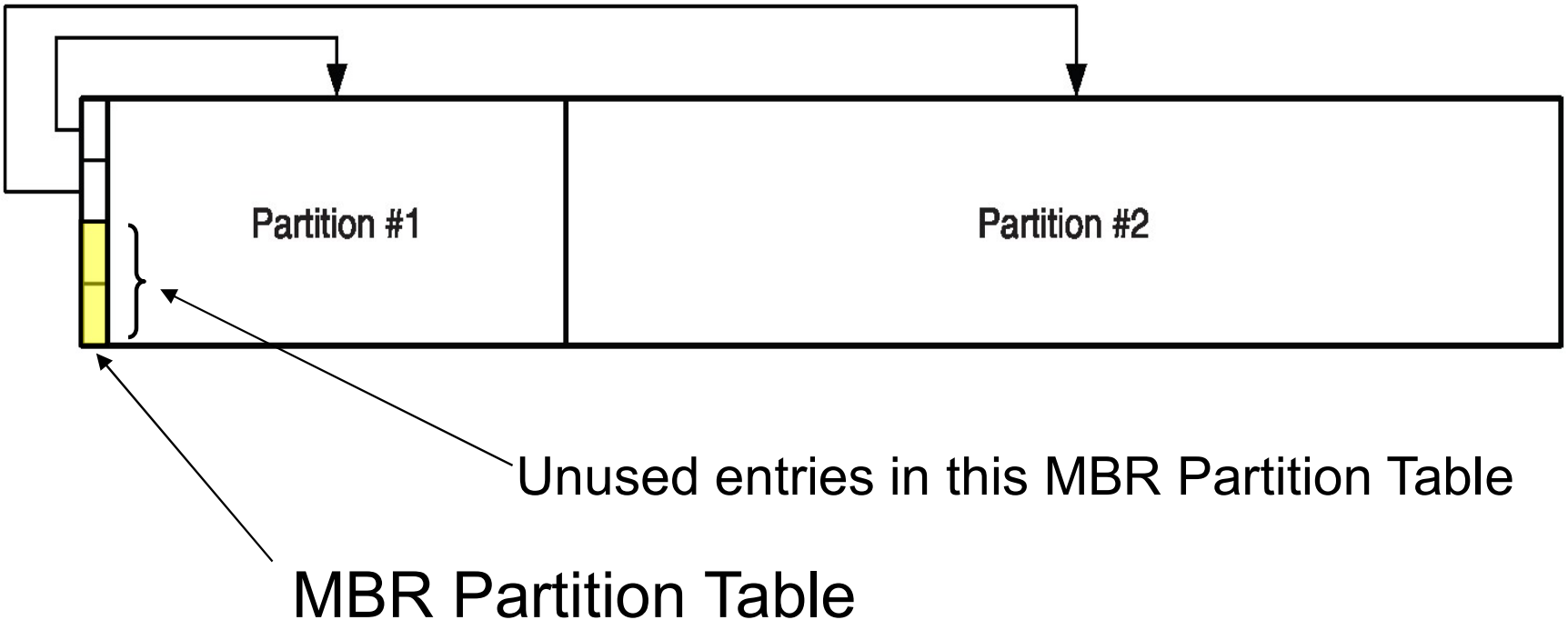
  } Accommodates legacy CHS systems < 8 GB

- *Starting LBA sector address*
- *Number of sectors in the partition*

  } Works for everything else (to terabytes)

- *Type of partition* ← File system or other
- *Flags* ← Which partition is bootable

Example of Basic MBR Partition



Notes on Partition Type*

The partition Type field describes what is in the partition

e.g., FAT12, FAT16, FAT32, NTFS, Linux, LinuxSwap, MacOSX, VmWare, VmWareSwap…

Windows OSs use the type field to decide how to interpret the contents of the partition and how to mount it

By simply changing the type field to a partition type that Windows doesn't support, Windows won't mount it

Thus, by changing the partition type that Windows does not support (e.g., Linux), a whole partition can be hidden from a Windows OS

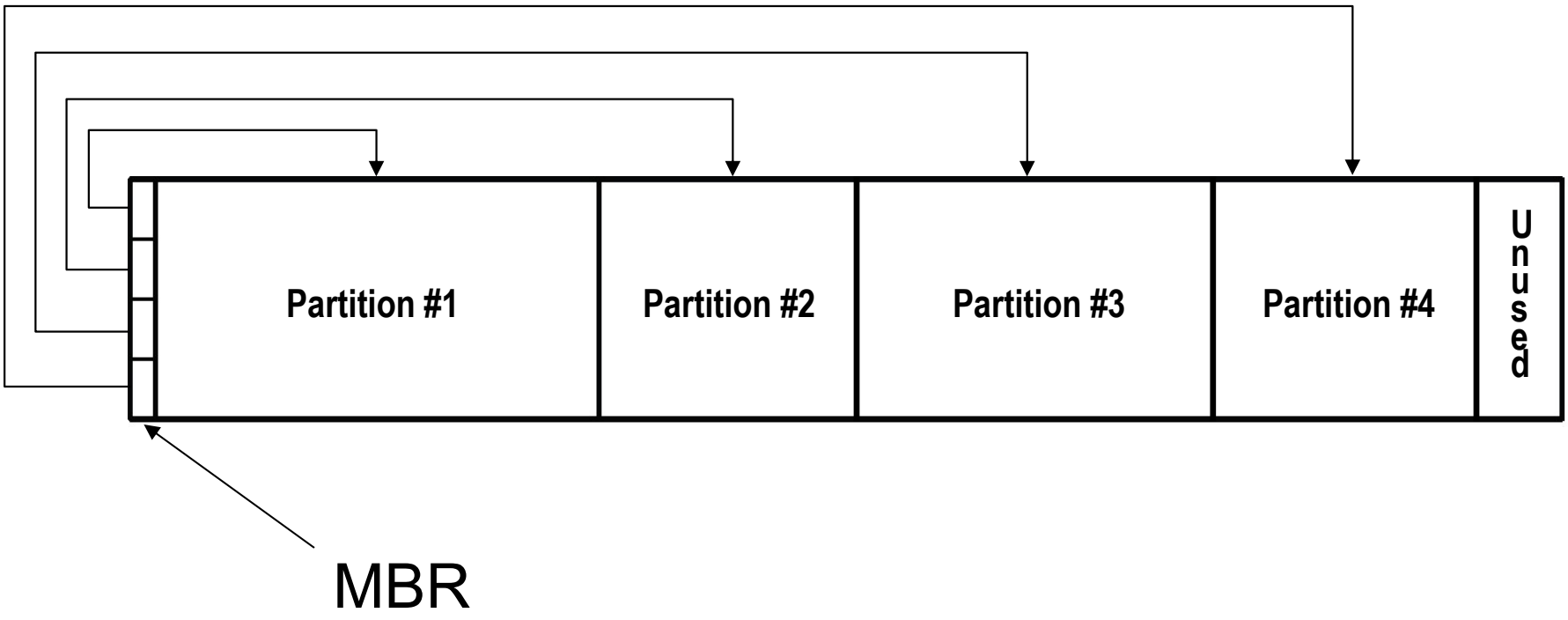
Notes on Partition Type

Other OSs do not rely on the type field

For instance, if a FAT32 partition has its type field set to MacOSX, Linux will still mount it as FAT32

But Windows will not

Maximum Number of Partitions



This limitation needed to be fixed

The concept of an ***Extended Partition*** was introduced

Extended Partitions

The solution to a limit of 4 partitions

Some Definitions

We're now going to discuss some definitions that are important

Primary Partition

A partition that has an entry in the MBR

There are two types of Primary Partitions

Primary File System Partition

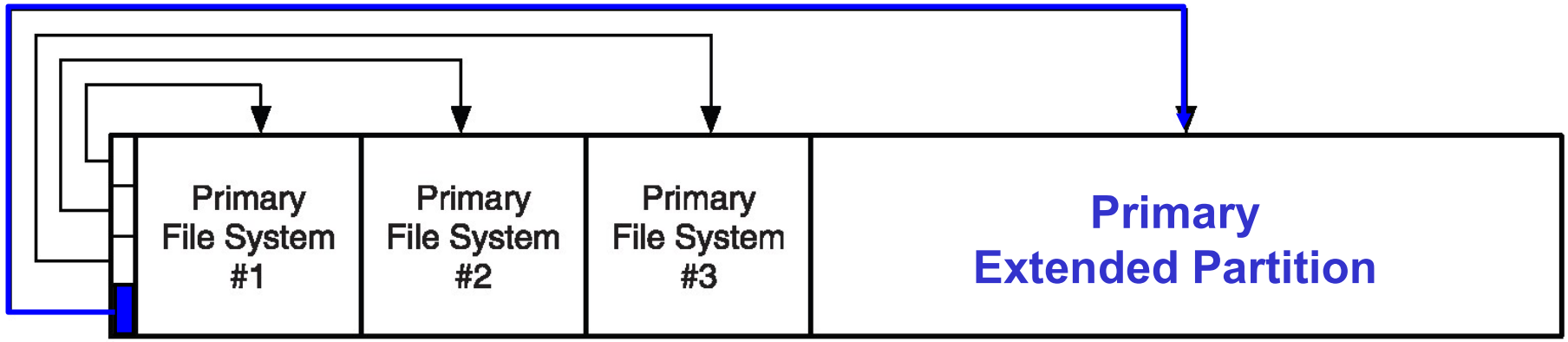
A Primary Partition that contains a file system

Primary Extended Partition

A Primary Partition that contains additional partitions

The word "*primary*" equates to an entry in MBR

Disk with a *Primary Extended Partition*



Extended Partition Overview

An ***Extended Partition*** can be thought of as a container or wrapper for an additional partition

Extended Partitions have ***Extended Partition Boot Records*** in their first sectors

An ***Extended Partition*** can contain two partitions

*A **File System Partition***

*An **Extended Partition***

Note:

Extended Partition Boot Records are similar to MBRs

But don't confuse MBRs with Extended Partition Boot Records

More Definitions

A ***Primary Extended Partition*** contains two partitions

Secondary File System Partition

Also called a ***Logical Partition***

Secondary Extended Partition

More Definitions

Secondary File System Partition

A partition within an Extended Partition that contains a file system

Secondary Extended Partition

*A partition within a **Primary Extended Partition** or a **Secondary Extended Partition** that contains additional partitions*

The word "*secondary*" equates to an entry in an *Extended Partition Boot Record*

Primary: Think ***MBR***
Secondary: Think ***Extended***

Example

Suppose that you have a 12 GB volume

You need six partitions each of 2 GB

You can only have 4 Primary Partitions

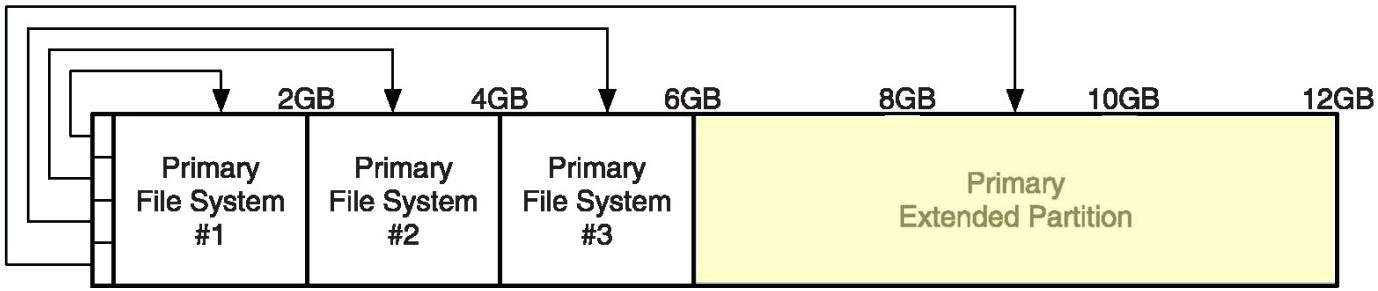
What to do?

You could create

3 Primary File System Partitions

1 Primary Extended Partition

3 Primary & 1 Extended



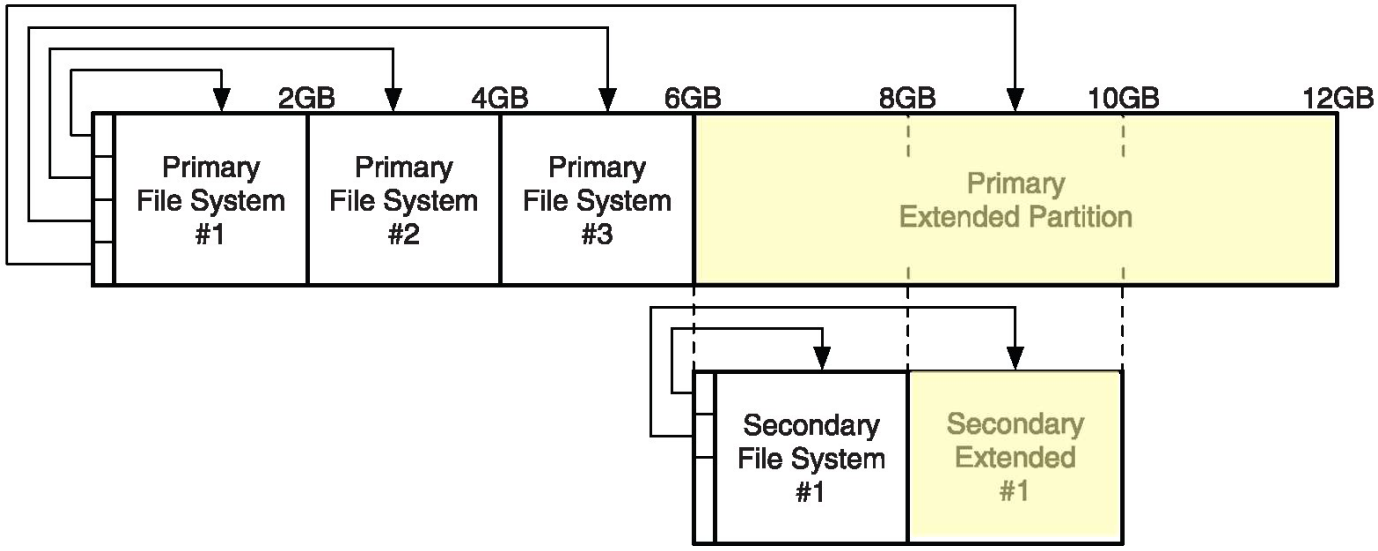
What to Do?

Within the Primary Extended Partition create

1 Secondary File System Partition

1 Secondary Extended Partition

Four File System Partitions



Now we have 4 partitions that can contain file systems that in turn contain OSs or other data

What next?

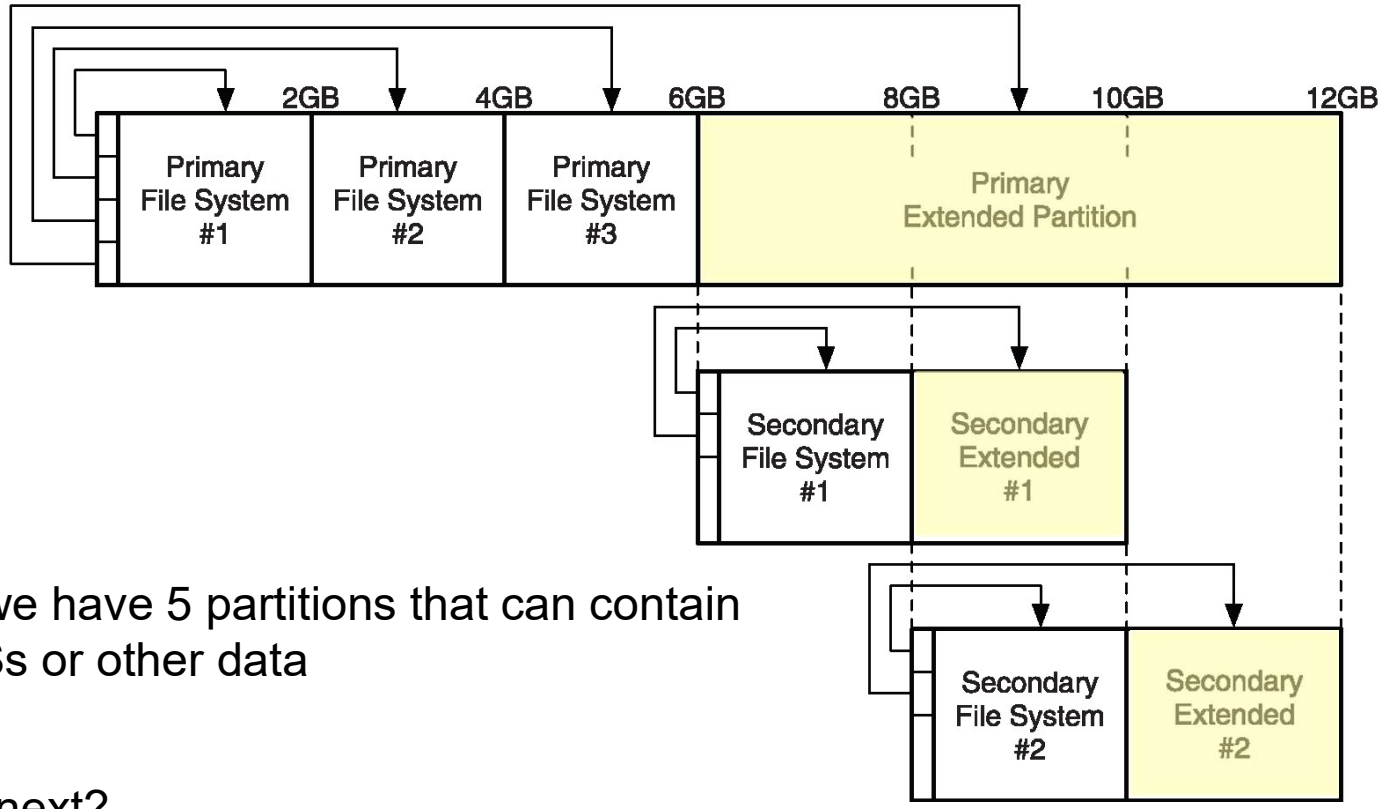
Create within *Secondary Extended #1*

One Secondary File System Partition (#2)

Also create

1 Secondary Extended Partition # 2

Five File System Partitions



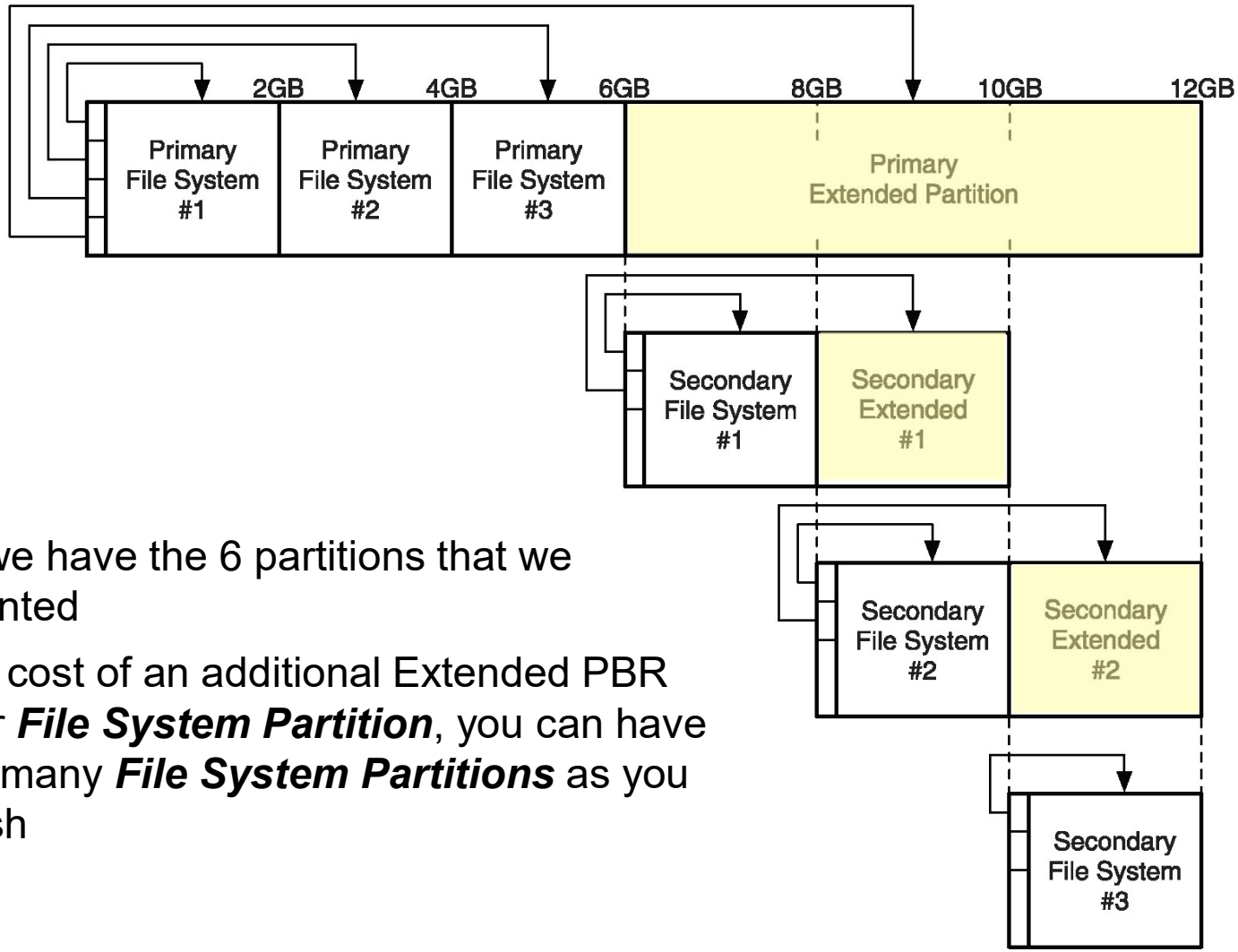
Now we have 5 partitions that can contain OSs or other data

What next?

Create within Secondary Extended # 2 partition

1 Secondary File System Partition (#3)

Six File System Partitions



Now we have the 6 partitions that we wanted

At the cost of an additional Extended PBR per ***File System Partition***, you can have as many ***File System Partitions*** as you wish

Note on Capacity of Secondary Extended Partitions

An extended partition table should have, at most

One entry for a secondary file system partition plus

One entry for a secondary extended partition

Note on Capacity of Secondary Extended Partitions

But Carrier writes that

Most operating systems will not generate an error if more than two entries are in the Secondary Extended Partition Table

He created a disk with a Primary Extended Partition that had in it

2 Secondary File System Partitions plus

1 Secondary Extended Partition

He reported that

Some forensic tools properly handled the third partition entry

Others ignored it or claimed that a 25 MB partition was a 1 TB partition

Note on Extended Partition Table Type Fields

Extended partitions have special types that are used in their partition table type field entries

e.g., Microsoft Extended CHS, Microsoft Extended LBA, Linux Extended…

There are several extended partition types that can appear in a type field

There is no type field differentiation between primary and secondary extended partitions

By just using the type fields it is hard to know what is really there

Boot Process

Boot Process*

Review

MBR boot code is in bytes 0-445 of sector 0 of the volume

But can extend into sectors #1- #62 as needed

Bytes 446-509 of sector 0 contain the partition table with room for 4 primary partitions (file system or extended)

The boot code examines the partition table to see which partition has the bootable flag set

What does the boot code look for?

Looks for the boot flag in the 1st byte in one of the primary partition table entries

It finds a bootable partition and determines the location of its 1st sector

The machine then "jumps" (i.e., transfers execution) to the first byte of the first sector of the bootable partition

Multiple OSs

Multiple operating systems are often installed on IA32 computers

So how is a computer organized so that

Multiple OSs can exist

At boot time a choice can be made as to which OS will run

There are currently two approaches

Microsoft's

Other

Multiple OSs
Microsoft Approach

MBR code is executed normally, identifying the bootable partition

Boot code exists in bytes 0-445 of sector 0 of the bootable partition

MBR code loads the code that is in this bootable partition

Computer executes this bootable partition code

User is prompted by the bootable partition code to choose a OS

e.g., boot.ini in some Windows OSs is used by the bootable partition code to display a choice for the user

User chooses an OS

Computer jumps to the 1st byte of the 1st sector (sector 0) of the selected operating system partition

Note: This approach allows OSs to boot from non-bootable partitions

Multiple OSs
Other Approach

MBR code in bytes 0-445 of sector 0 is modified

More MBR code is usually added in sectors 1, 2… through up to sector 62

New MBR code prompts the user to choose the OS to load

User chooses an OS

Computer jumps to the 1st byte of the 1st sector of the selected partition

Note: This approach makes the concept of a bootable partition irrelevant because it ignores the boot flag

Note on Boot Sector Virus

Suppose a virus contaminates the boot code

Called a boot sector virus

It will be executed each time you boot the computer

It may

Create a partition

Put stuff in it

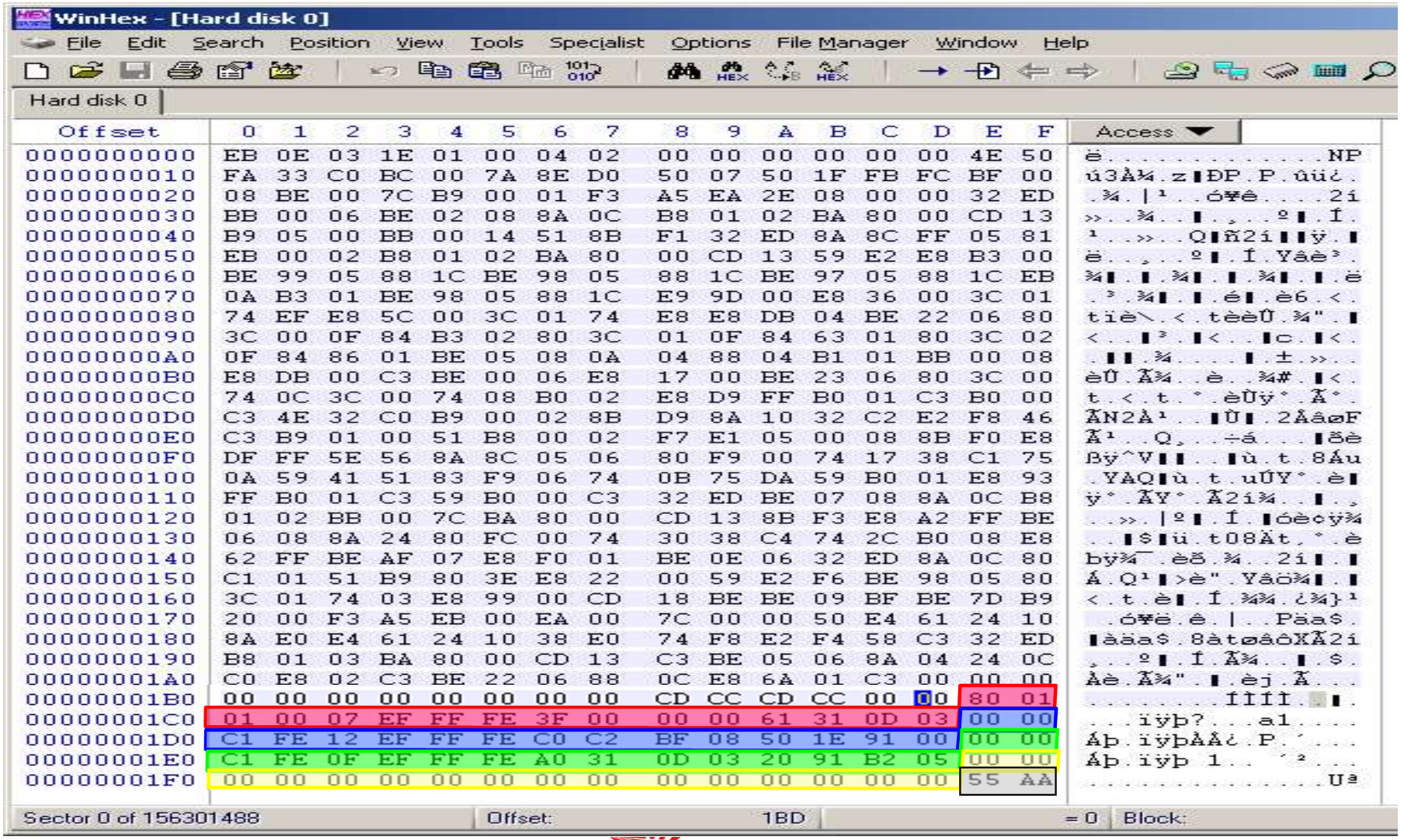
Then hide the partition by making it something that the OS can't mount

Basic MBR Partition Table Organization

Basic MBR Data Structure

Byte Range	Description	Essential
0–445 (0x01BD)	Boot Code	No
446–461 (01BE–01CD)	Partition Table Entry #1	Yes
462–477 (01CE–01DD)	Partition Table Entry #2	Yes
478–493 (01DE–01ED)	Partition Table Entry #3	Yes
494–509 (01EE–01FD)	Partition Table Entry #4	Yes
510–511 (01FE–01FF)	Signature value (0xAA55)	No

MBR Boot Code, Partition Table & Signature Value Indicated



Data Structure for Each MBR Partition Table Entry

Byte Range	Description	Essential	
0–0	Bootable Flag	No	0x80 if bootable Ignored for some multi-boot systems
1–3	Starting CHS Address	Yes	2^{24} x 512 = ~ 8.1 GB max H: 8 bits; C: 10 bits; S: 6 bits
4–4	Partition Type	No	Discussed next
5–7	Ending CHS Address	Yes	2^{24} x 512 = ~ 8.1 GB max H: 8 bits; C: 10 bits; S: 6 bits
8–11	Starting LBA Address	Yes	2^{32} x 512 = ~ 2.2 TB max
12–15	Size in Sectors	Yes	2^{32} x 512 = ~ 2.2 TB max

Operating system decides which sector addressing to use; CHS or LBA

Windows 95, 98 & ME used CHS (the last that I know of)
Windows 2000 and later used LBA
Linux and IA32 Unix use LBA

Comment on Updating MBR to Accommodate Larger Disks

If the legacy CHS addressing were eliminated from the partition table entries, the capacity of the LBA and partition size could be

$2^{64} \times 512 = \sim 9.5 \times 10^{21} = \sim 9.5$ *SB (Sextillion bytes)*

This plus Extended Partitions would handle very large disks

This was and is one of the arguments for continuing to use BIOS & MBR and not going to EFI & GPT

However EFI & GPT have gained acceptance

So both are used today

Some MBR Partition Types

Type	Description
0x00	Empty
0x01	FAT12, CHS
0x04	FAT16, 16–32 MB, CHS
0x05	Microsoft Extended, CHS
0x06	FAT16, 32 MB–2GB, CHS
0x07	NTFS
0x0b	FAT32, CHS
0x0c	FAT32, LBA
0x0e	FAT16, 32 MB–2GB, LBA
0x0f	Microsoft Extended, LBA
0x11	Hidden FAT12, CHS

Type	Description
0x14	Hidden FAT16, 16–32 MB, CHS
0x16	Hidden FAT16, 32 MB–2GB, CHS
0x1b	Hidden FAT32, CHS
0x1c	Hidden FAT32, LBA
0x1e	Hidden FAT16, 32 MB–2GB, LBA
0x42	Microsoft MBR. Dynamic Disk
0x82	Solaris x86
0x82	Linux Swap
0x83	Linux
0x84	Hibernation
0x85	Linux Extended

Type	Description
0x86	NTFS Volume Set
0x87	NTFS Volume Set
0xa0	Hibernation
0xa1	Hibernation
0xa5	FreeBSD
0xa6	OpenBSD
0xa8	Mac OSX
0xa9	NetBSD
0xab	Mac OSX Boot
0xb7	BSDI
0xb8	BSDI swap
0xee	EFI GPT Disk
0xef	EFI System Partition
0xfb	Vmware File System
0xfc	Vmware swap

Some Microsoft MBR Partition Types

Type	Description
0x00	Empty
0x01	FAT12, CHS
0x04	FAT16, 16–32 MB, CHS
0x05	Microsoft Extended, CHS
0x06	FAT16, 32 MB–2GB, CHS
0x07	NTFS
0x0b	FAT32, CHS
0x0c	FAT32, LBA
0x0e	FAT16, 32 MB–2GB, LBA
0x0f	Microsoft Extended, LBA

- Only Microsoft partition types are show here
- Remember: Windows uses the Type Field to determine the file system in the partition

Some Hidden Microsoft MBR Partition Types

Type	Description
0x00	Empty
0x01	FAT12, CHS
0x04	FAT16, 16–32 MB, CHS
0x05	Microsoft Extended, CHS
0x06	FAT16, 32 MB–2GB, CHS
0x07	NTFS
0x0b	FAT32, CHS
0x0c	FAT32, LBA
0x0e	FAT16, 32 MB–2GB, LBA
0x0f	Microsoft Extended, LBA
0x11	Hidden FAT12, CHS

Type	Description
0x14	Hidden FAT16, 16–32 MB, CHS
0x16	Hidden FAT16, 32 MB–2GB, CHS
0x1b	Hidden FAT32, CHS
0x1c	Hidden FAT32, LBA
0x1e	Hidden FAT16, 32 MB–2GB, LBA

FAT hidden partitions have the same last hex value as their non-hidden equivalents

The 1st hex value is a 0 for non-hidden and a 1 for hidden

MBR Partition Table Example

From Carrier

Overview
Carrier's Example

Dual boot

- *Microsoft Windows*
- *Linux*

Eight file system partitions

Tools used

- *dd*
  - Copies a file or disk
  - Converts and formats depending on chosen options
- *xxd*
  - Hex dump of **`xxd`**'s input
  - Input can be stdin or a file

Command Line Syntax

`# dd if=disk3.dd bs=512 skip=0 count=1 | xxd`

 # prompt means that Carrier is super user

Copy part of the image file `disk3.dd`

disk3.dd is a file containing a previous bit-by-bit copy of 80GB disk

`bs=512`

Block size = 512 bytes (1 sector)

`skip=0`

Don't skip any sectors before starting. Start with sector 0, the MBR

`count=1`

Copy 1 sector

`| xxd`

Pipe the output of dd to xxd. xxd outputs to stdout

Command Output
MBR

```
# dd if=disk3.dd bs=512 skip=0 count=1 | xxd
0000000: eb48 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0 .H..............
          _  _ null H  a  r  d  space D  i  s  k  null R  e  a  d  null
0000384: 0048 6172 6420 4469 736b 0052 6561 6400 .Hard Disk.Read.
0000400: 2045 7272 6f72 00bb 0100 b40e cd10 ac3c  Error........<
0000416: 0075 f4c3 0000 0000 0000 0000 0000 0000 .u..............
0000432: 0000 0000 0000 0000 0000 0000 0000 0001 ................
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000 ....?.?...A`....
0000464: 0180 83fe 3f8c 8060 1f00 cd2f 0300 0000 ....?..`.../....
0000480: 018d 83fe 3fcc 4d90 2200 40b0 0f00 0000 ....?.M.".@.....
0000496: 01cd 05fe ffff 8d40 3200 79eb 9604 55aa .......@2.y...U.
```

Command Output

MBR Boot Code

```
# dd if=disk3.dd bs=512 skip=0 count=1 | xxd
0000000: eb48 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0 .H..............
[REMOVED]
0000384: 0048 6172 6420 4469 736b 0052 6561 6400 .Hard Disk.Read.
0000400: 2045 7272 6f72 00bb 0100 b40e cd10 ac3c  Error.........<
0000416: 0075 f4c3 0000 0000 0000 0000 0000 0000 .u..............
0000432: 0000 0000 0000 0000 0000 0000 0000 0001 ................
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000 ....?.?...A`....
0000464: 0180 83fe 3f8c 8060 1f00 cd2f 0300 0000 ....?..`.../....
0000480: 018d 83fe 3fcc 4d90 2200 40b0 0f00 0000 ....?.M.".@.....
0000496: 01cd 05fe ffff 8d40 3200 79eb 9604 55aa .......@2.y...U.
```

Command Output

Partition Table Entry #1

```
# dd if=disk3.dd bs=512 skip=0 count=1 | xxd
0000000: eb48 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0 .H..............
[REMOVED]
0000384: 0048 6172 6420 4469 736b 0052 6561 6400 .Hard Disk.Read.
0000400: 2045 7272 6f72 00bb 0100 b40e cd10 ac3c  Error........<
0000416: 0075 f4c3 0000 0000 0000 0000 0000 0000 .u..............
0000432: 0000 0000 0000 0000 0000 0000 0000 0001 ................
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000 ....?.?...A`....
0000464: 0180 83fe 3f8c 8060 1f00 cd2f 0300 0000 ....?..`.../....
0000480: 018d 83fe 3fcc 4d90 2200 40b0 0f00 0000 ....?.M.".@.....
0000496: 01cd 05fe ffff 8d40 3200 79eb 9604 55aa .......@2.y...U.
```

Bootable flag

Starting CHA Address

Partition Type (NTFS)

Ending CHA Address

Starting LBA Address

Length in Sectors

Command Output

Partition Table Entry #2

```
# dd if=disk3.dd bs=512 skip=0 count=1 | xxd
0000000: eb48 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0 .H..............
[REMOVED]
0000384: 0048 6172 6420 4469 736b 0052 6561 6400 .Hard Disk.Read.
0000400: 2045 7272 6f72 00bb 0100 b40e cd10 ac3c  Error........<
0000416: 0075 f4c3 0000 0000 0000 0000 0000 0000 .u..............
0000432: 0000 0000 0000 0000 0000 0000 0000 0001 ................
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000 ....?.?...A`....
0000464: 0180 83fe 3f8c 8060 1f00 cd2f 0300 0000 ....?..`.../....
0000480: 018d 83fe 3fcc 4d90 2200 40b0 0f00 0000 ....?.M.".@.....
0000496: 01cd 05fe ffff 8d40 3200 79eb 9604 55aa .......@2.y...U.
```

Bootable flag

Partition Type (Linux)

Command Output

Partition Table Entry #3

```
# dd if=disk3.dd bs=512 skip=0 count=1 | xxd
0000000: eb48 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0 .H..............
[REMOVED]
0000384: 0048 6172 6420 4469 736b 0052 6561 6400 .Hard Disk.Read.
0000400: 2045 7272 6f72 00bb 0100 b40e cd10 ac3c  Error........<
0000416: 0075 f4c3 0000 0000 0000 0000 0000 0000 .u..............
0000432: 0000 0000 0000 0000 0000 0000 0000 0001 ................
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000 ....?.?...A`....
0000464: 0180 83fe 3f8c 8060 1f00 cd2f 0300 0000 ....?..`.../....
0000480: 018d 83fe 3fcc 4d90 2200 40b0 0f00 0000 ....?.M."."@.....
0000496: 01cd 05fe ffff 8d40 3200 79eb 9604 55aa .......@2.y...U.
```

Bootable flag

Partition Type (Linux)

Command Output

Partition Table Entry #4

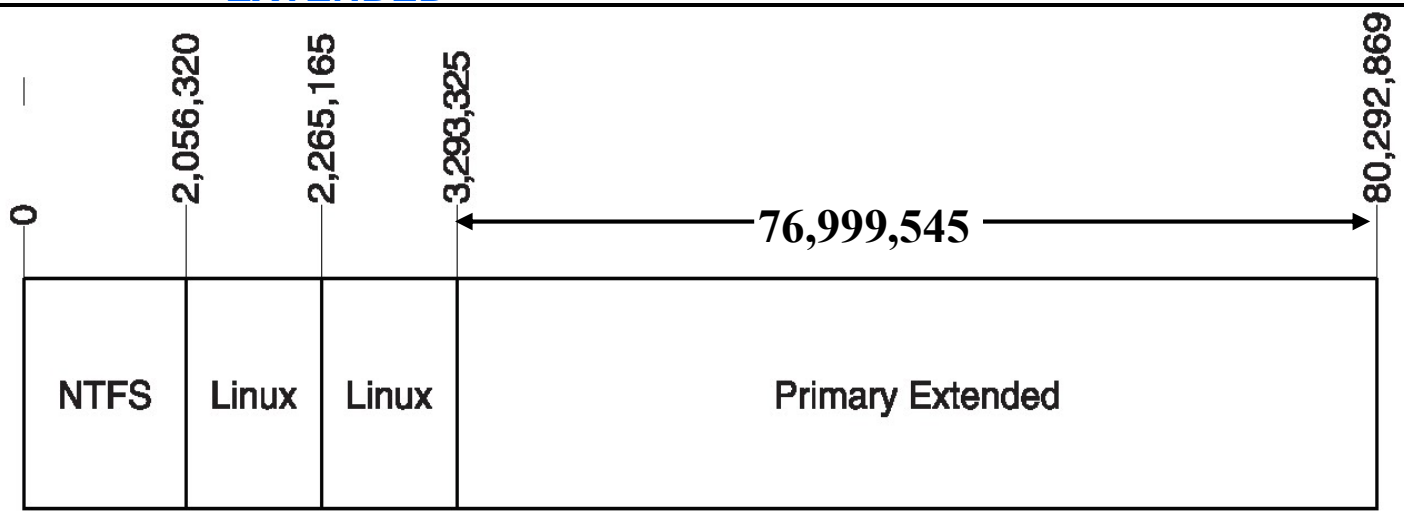
```
# dd if=disk3.dd bs=512 skip=0 count=1 | xxd
0000000: eb48 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0 .H..............
[REMOVED]
0000384: 0048 6172 6420 4469 736b 0052 6561 6400 .Hard Disk.Read.
0000400: 2045 7272 6f72 00bb 0100 b40e cd10 ac3c  Error........<
0000416: 0075 f4c3 0000 0000 0000 0000 0000 0000 .u..............
0000432: 0000 0000 0000 0000 0000 0000 0000 0001 ................
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000 ....?.?...A`....
0000464: 0180 83fe 3f8c 8060 1f00 cd2f 0300 0000 ....?..`.../....
0000480: 018d 83fe 3fcc 4d90 2200 40b0 0f00 0000 ....?.M.".@.....
0000496: 01cd 05fe ffff 8d40 3200 79eb 9604 55aa .......@2.y...U.
```

Bootable flag

Partition Type (MS extended CHS)

MBR Partition Table Contents

#	Flag	Type	Starting Sector	Size
1	0x00	0x07 NTFS	0x0000003f (63)	0x001f6041 (2,056,257)
2	0x80	0x83 LINUX	0x001f6080 (2,056,320)	0x00032fcd (208,845)
3	0x00	0x83 LINUX	0x0022904d (2,265,165)	0x000fb040 (1,028,160)
4	0x00	0x05 EXTENDED	0x0032408d (3,293,325)	0x0496eb79 (76,999,545)



Extended Partitions

An Aside

Extended Partitions
Confusion

The first sectors of extended partitions have same data structure as the MBR

But it is used differently than the MBR

It's used to create a linked list

Starting addresses for ***Secondary File System*** entries in the extended partition, partition table are relative to the beginning of the current *Extended Partition*

By contrast, starting addresses for ***Secondary Extended Partition*** entries are relative to the ***Primary Extended Partition***

This is confusing so let's read it again and discuss it

We will go over it again in a little while

Extended Partitions
More Confusion

A ***Primary Extended Partition*** must have a size capable of containing

The partition table record PLUS

All of the ***Secondary File System Partitions*** *PLUS*

All of the ***Secondary Extended Partitions***

A ***Secondary Extended Partition*** must have a size capable of containing

The partition table record PLUS

The next Secondary File System Partition

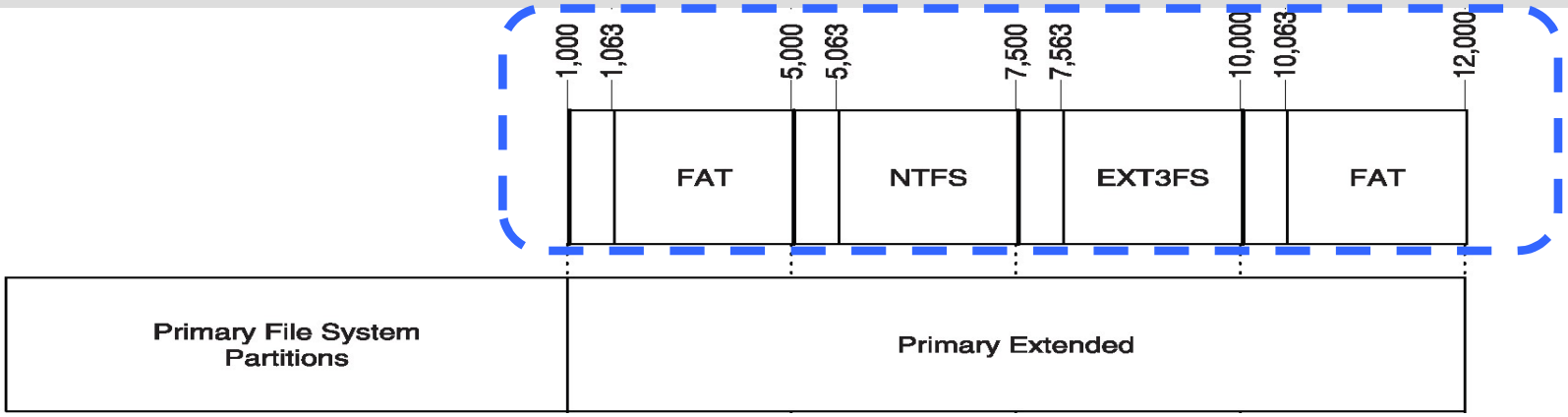
Does not need to contain further partitions

Extended Primary Partition Example

Initial Primary Partitioning

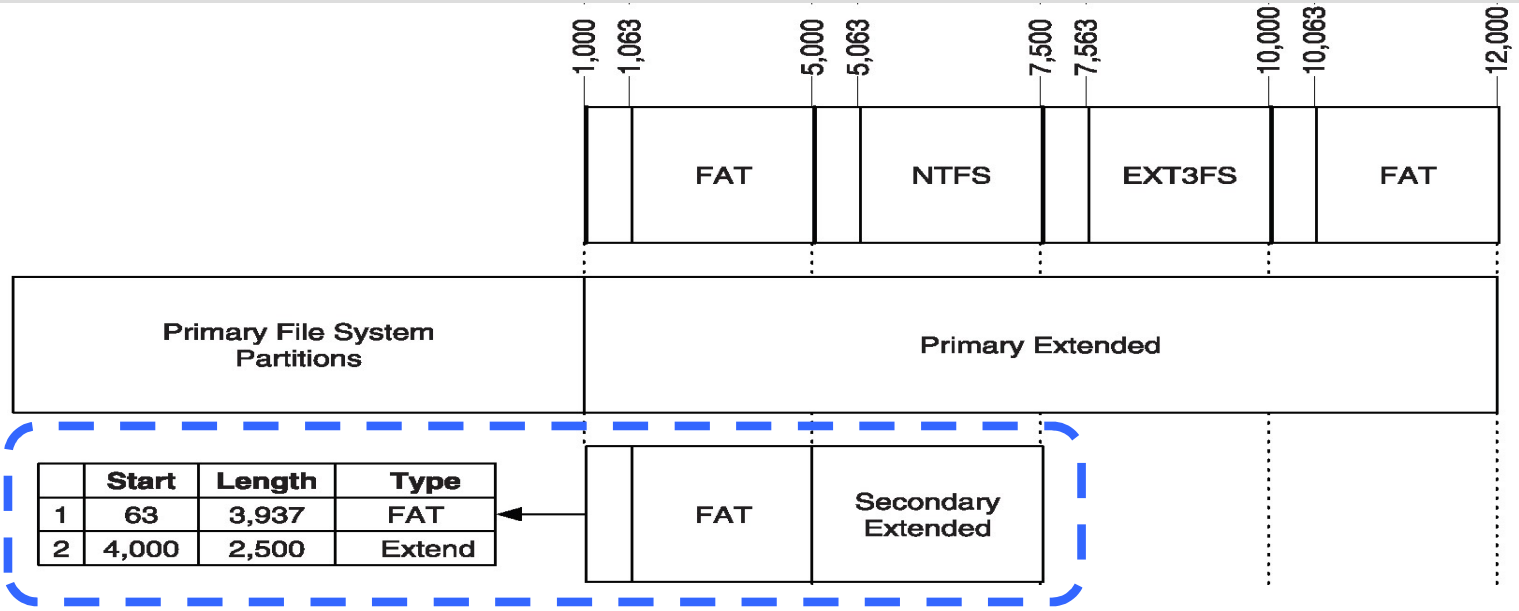
Extended Primary Partition Example

Desired Partitions in Primary Extended Partition



Extended Primary Partition Example

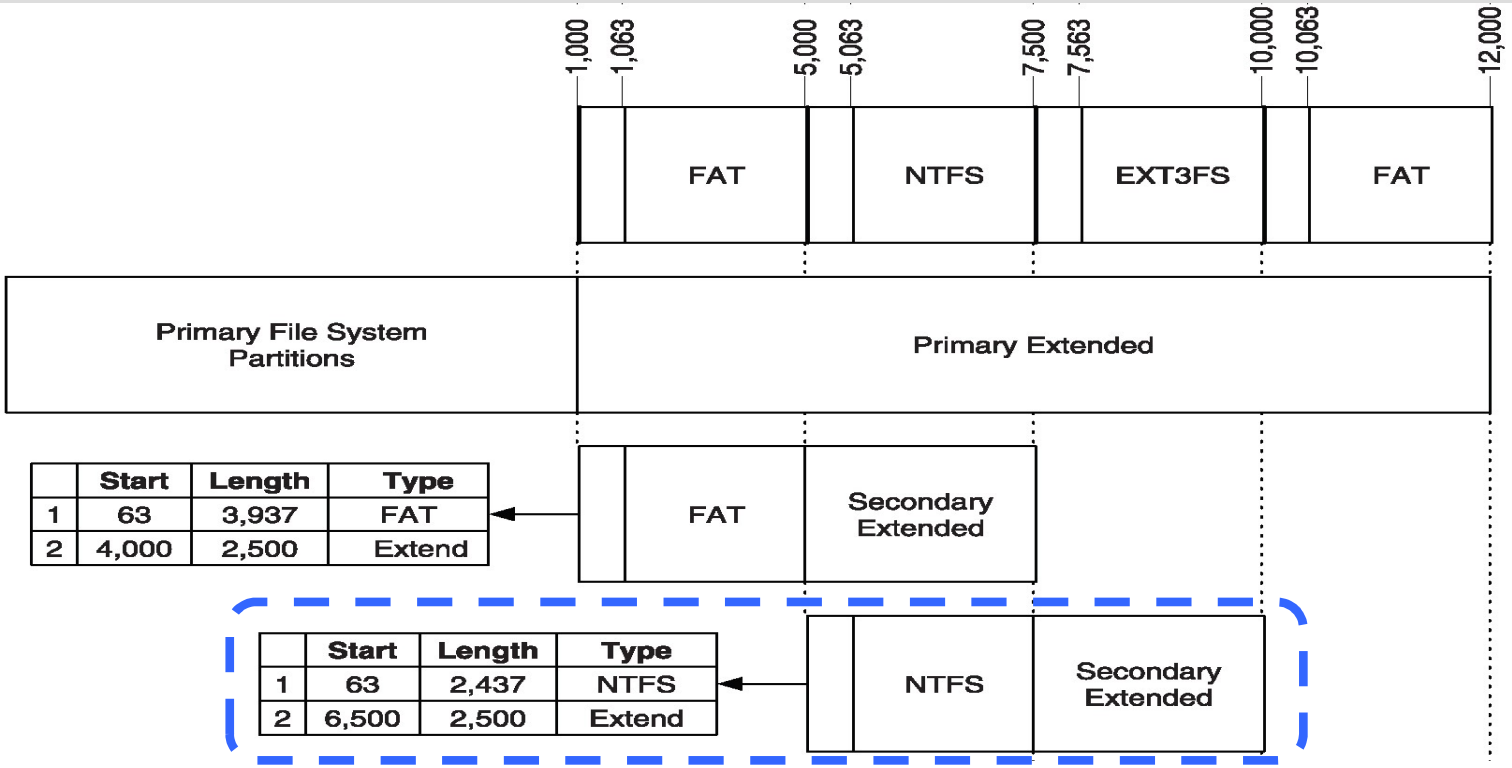
1st FAT Secondary File System Partition



	Start	Length	Type
1	63	3,937	FAT
2	4,000	2,500	Extend

Extended Primary Partition Example

2nd NTFS Secondary File System Partition

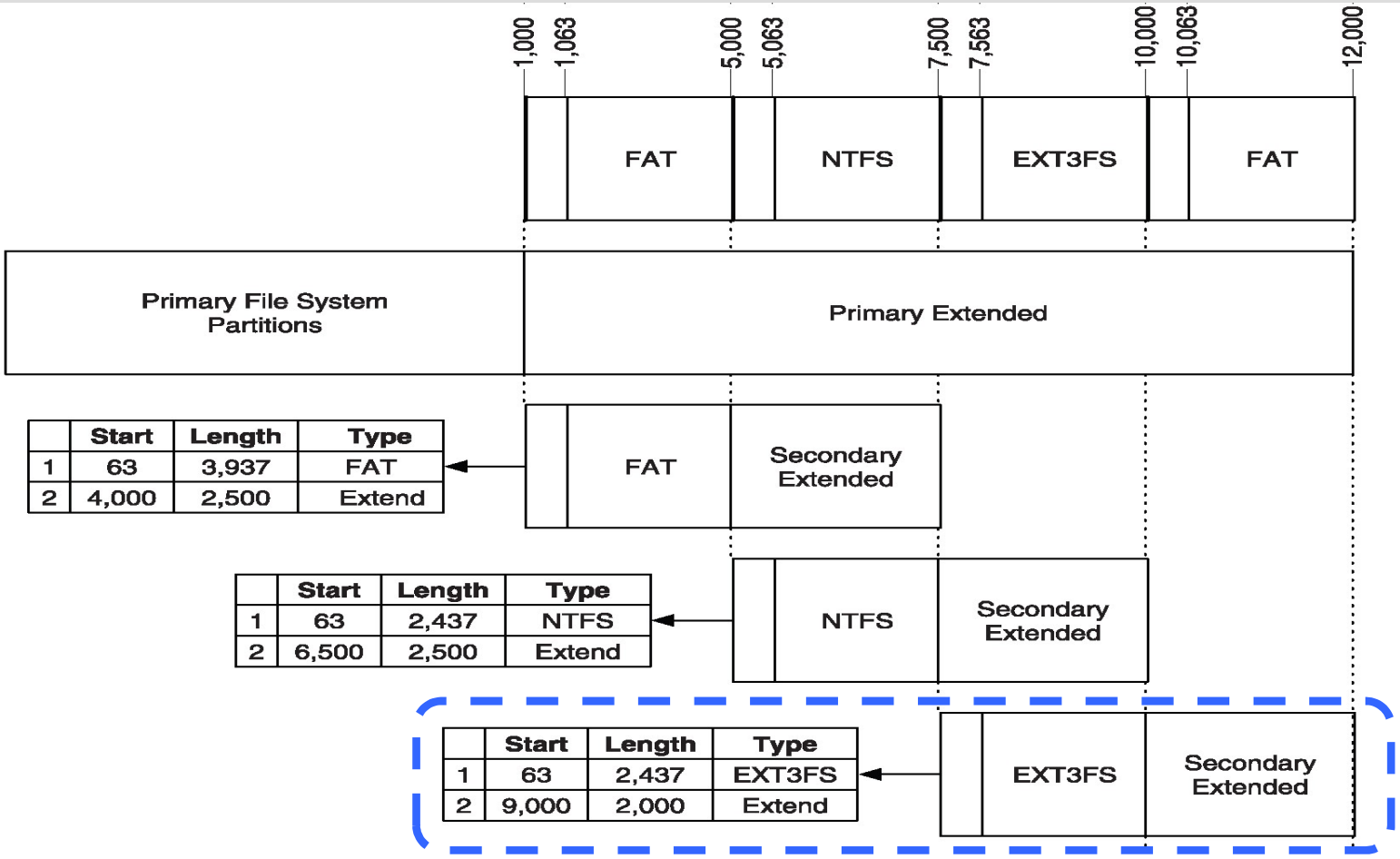


	Start	Length	Type
1	63	3,937	FAT
2	4,000	2,500	Extend

	Start	Length	Type
1	63	2,437	NTFS
2	6,500	2,500	Extend

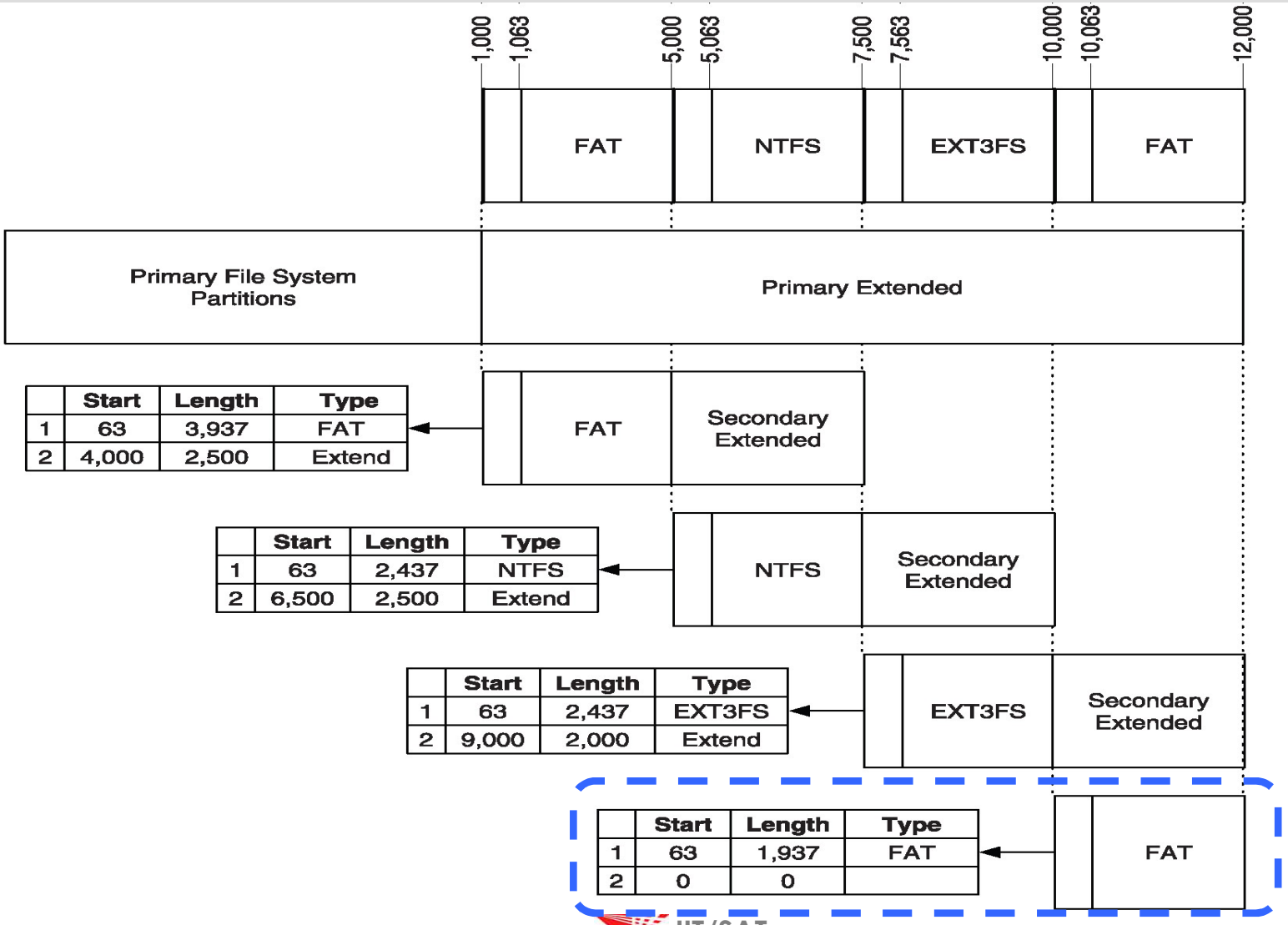
Extended Primary Partition Example

3rd EXT3 Secondary File System Partition



Extended Primary Partition Example

4th FAT Secondary File System Partition



	Start	Length	Type
1	63	3,937	FAT
2	4,000	2,500	Extend

	Start	Length	Type
1	63	2,437	NTFS
2	6,500	2,500	Extend

	Start	Length	Type
1	63	2,437	EXT3FS
2	9,000	2,000	Extend

	Start	Length	Type
1	63	1,937	FAT
2	0	0	

Extended Primary Partition

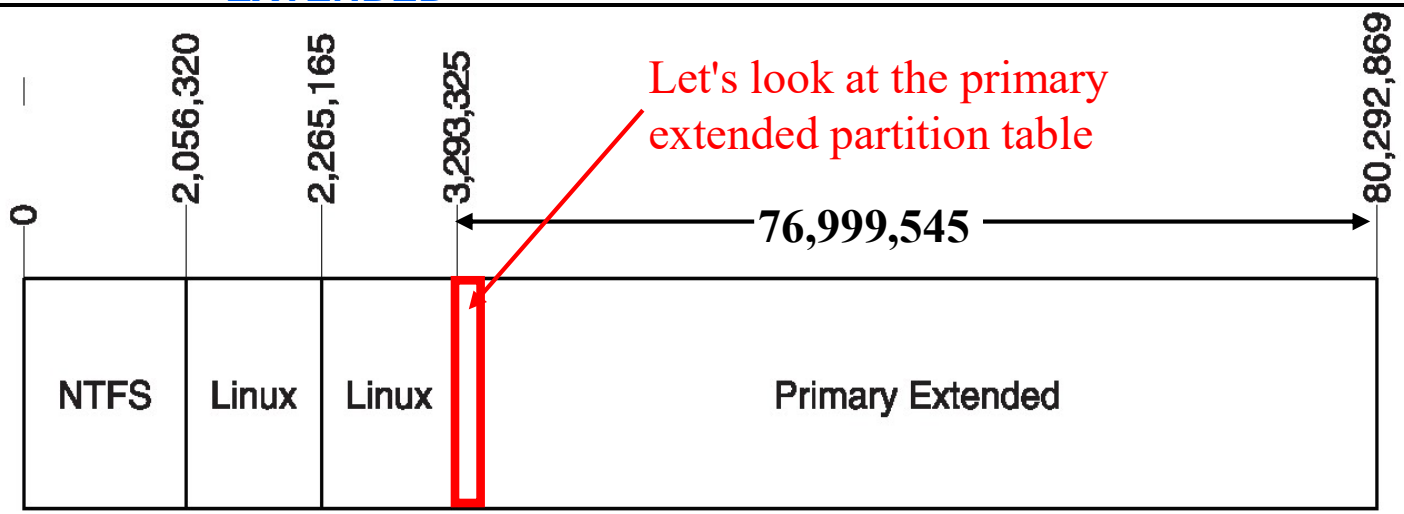
Return to Earlier Partition Table Example

Extended Primary Partition

Lets go back to the earlier MBR example and look at the primary extended partition

MBR Partition Table Contents

#	Flag	Type	Starting Sector	Size
1	0x00	0x07 NTFS	0x0000003f (63)	0x001f6041 (2,056,257)
2	0x80	0x83 LINUX	0x001f6080 (2,056,320)	0x00032fcd (208,845)
3	0x00	0x83 LINUX	0x0022904d (2,265,165)	0x000fb040 (1,028,160)
4	0x00	0x05 EXTENDED	0x0032408d (3,293,325)	0x0496eb79 (76,999,545)



Primary Extended Partition Boot Record

```
# dd if=disk3.dd bs=512 skip=3293325 count=1 | xxd
[REMOVED]
0000432: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000448: 01cd 83fe 7fcb 3f00 0000 0082 3e00 0000  ......?.....>...
0000464: 41cc 05fe bf0b 3f82 3e00 40b0 0f00 0000  A.....?.>.@.....
0000480: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000496: 0000 0000 0000 0000 0000 0000 0000 55aa  ..............U.
```

Relative to start of current partition table

#	Flag	Type	Starting Sector	Size
5	0x00	0x83 LINUX	0x0000003f (63)	0x003e8200 (4,096,512)
6	0x00	0x05 EXTENDED	0x003e823f (4,096,575)	0x000fb040 (1,028,160)

Relative to primary extended partition

Primary Extended Partition Boot Record

Remember:

A primary extended partition must have a size capable of containing:

All of the secondary file system partitions plus

All of the secondary extended partitions

A secondary extended partition has a size of

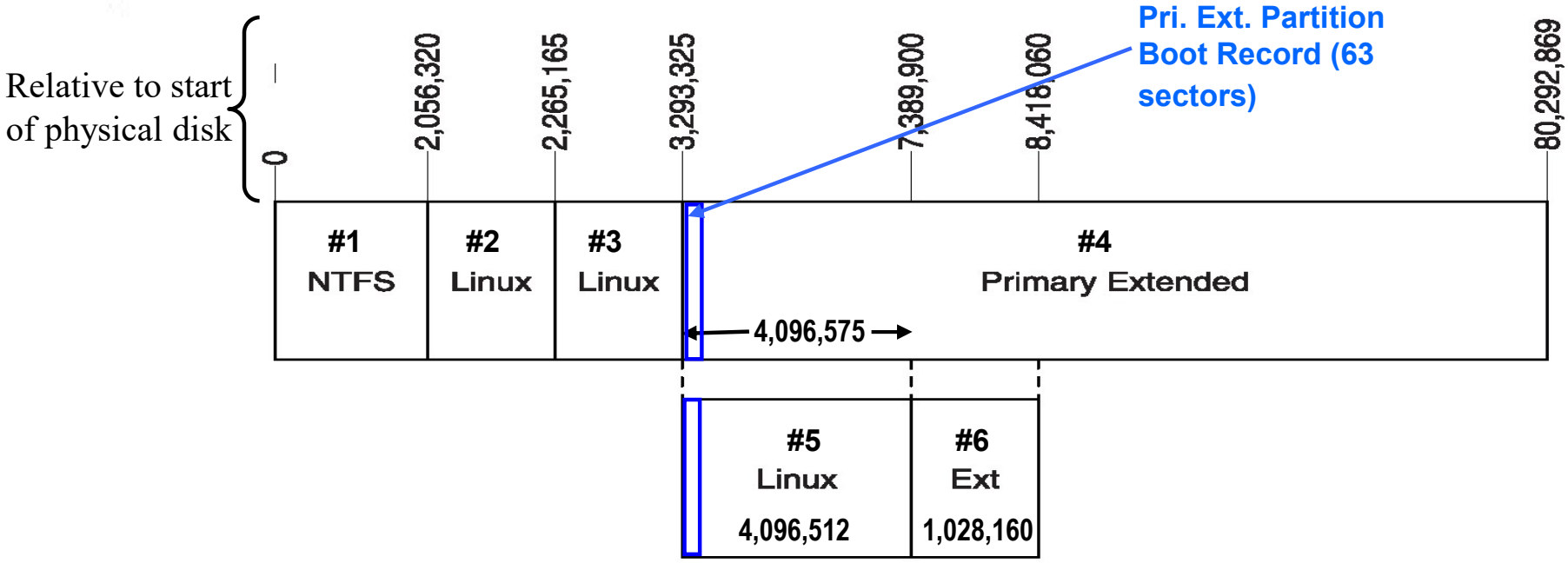
The next secondary file system partition plus

The partition table record

Primary Extended Partition Boot Record

#	Flag	Type	Starting Sector	Size
5	0x00	0x83	0x0000003f (63)	0x003e8200 (4,096,512)
6	0x00	0x05	0x003e823f (4,096,575)	0x000fb040 (1,028,160)

4,096,512 + 63 = 4,096,575

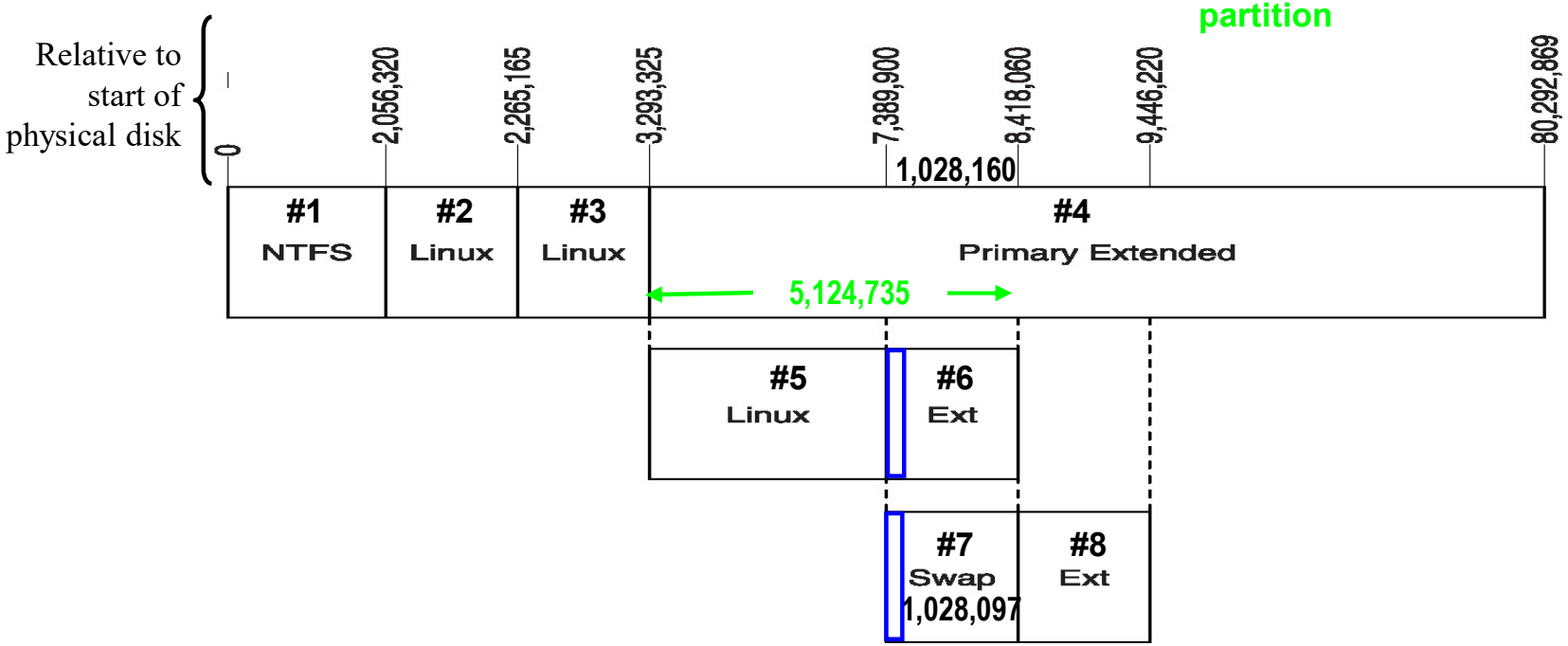


Secondary Extended Partition Boot Record

#	Flag	Type	Starting Sector	Size
7	0x00	0x82	0x0000003f (63)	0x000fb001 (1,028,097)
8	0x00	0x05	0x004e327f (5,124,735)	0x000fb040 (1,028,160)

Relative to start of current partition table

Relative to primary extended partition



Partition Viewing Tools

Overview

A number of tools can be used to determine and view disk partitions

Some tools can only view the partitions on a disk

Others can both view and modify

Some Tools

Carrier discusses two tools

- *fdisk for *nix (not Windows fdisk)*
- *mmls (from TSK)*

Some other tools

- *FTK*
- ***Hex Workshop***
- ***WinHex** or X-Ways Forensics*
- *Windows **Computer Management** | Disk Management snap-in*
- *Partition Commander*
- *Partition Magic*

Examine Disk Using fdisk

Here Carrier uses the disk that we just considered and uses **fdisk** to identify and locate the partitions

fdisk *can be used to get much (not all) partition information*

fdisk *come with all *nix distros (distributions)*

Note: fdisk in Windows is impotent for our purposes and seems not to exist for Windows 10

The fdisk command to retrieve volume partitions is

```
# fdisk -lu nameOfRawImageFile
```

-l List partitions; don't allow editing

-u Show LBA sectors instead of CHS

In the example disk3.dd is the raw image of a hard disk

Examine Disk Using diskpart

diskpart should be used instead of fdisk to examine volumes and partitions on Windows 10

Its use will be demonstrated in class, as time permits

*nix fdisk

```
# fdisk -lu disk3.dd
Disk disk3.dd: 255 heads, 63 sectors, 0 cylinders
Units = sectors of 1 * 512 bytes

   Device Boot      Start         End      Blocks   Id  System
#1 disk3.dd1           63     2056319     1028128+   7  HPFS/NTFS
#2 disk3.dd2   *    2056320     2265164      104422+  83  Linux
#3 disk3.dd3        2265165     3293324      514080   83  Linux
#4 disk3.dd4        3293325    80292869    38499772+   5  Extended
#5 disk3.dd5        3293388     7389899     2048256   83  Linux
#6 Missing
#8 Missing
#7 disk3.dd6        7389963     8418059      514048+  82  Linux swap
   disk3.dd7        8418123     9446219      514048+  83  Linux
   disk3.dd8        9446283    17639369     4096543+   7  HPFS/NTFS
   disk3.dd9       17639433    48371714    15366141   83  Linux
```

Block= Sectors/2

←Primary

fdisk does not show secondary extended partitions

Thus you are not seeing all the partition table entries

Examine Disk Using mmls

Here Carrier uses the disk that we just considered and uses **mmls** to identify and locate the partitions

mmls *yields all the partition information*

The mmls command to retrieve volume partitions is

`# mmls -t dos` ***nameOfRawImageFile***

`-t` *Must define the type of partitioning*

`dos` identifies a Basic MBR Partition scheme

Note: In Win10 these will work to see the 1st drive on a computer

```
mmls -t dos \\.\PhysicalDrive0
mmls \\.\PhysicalDrive0
```

mmls

Start, *End* & *Length* are in sectors

All partition tables are located

Slot syntax
PartitionTable:*Entry InTable*

Partition types are shown

Unallocated sectors are shown as "-----"

Doesn't show which partition table entry has bootable flag set. Why?

```
# mmls -t dos disk3.dd
Units are in 512-byte sectors
     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Table #0
01:  -----   0000000001   0000000062   0000000062   Unallocated
#1 02:  00:00   0000000063   0002056319   0002056257   NTFS (0x07)
#2 03:  00:01   0002056320   0002265164   0000208845   Linux (0x83)
#3 04:  00:02   0002265165   0003293324   0001028160   Linux (0x83)
#4 05:  00:03   0003293325   0080292869   0076999545   DOS Extended (0x05)
06:  -----   0003293325   0003293325   0000000001   Table #1
07:  -----   0003293326   0003293387   0000000062   Unallocated
#5 08:  01:00   0003293388   0007389899   0004096512   Linux (0x83)
#6 09:  01:01   0007389900   0008418059   0001028160   DOS Extended (0x05)
10:  -----   0007389900   0007389900   0000000001   Table #2
11:  -----   0007389901   0007389962   0000000062   Unallocated
#7 12:  02:00   0007389963   0008418059   0001028097   Linux Swap (0x82)
#8 13:  02:01   0008418060   0009446219   0001028160   DOS Extended (0x05)
14:  -----   0008418060   0008418060   0000000001   Table #3
15:  -----   0008418061   0008418122   0000000062   Unallocated
16:  03:00   0008418123   0009446219   0001028097   Linux (0x83)
17:  03:01   0009446220   0017639369   0008193150   DOS Extended (0x05)
18:  -----   0009446220   0009446220   0000000001   Table #4
19:  -----   0009446221   0009446282   0000000062   Unallocated
20:  04:00   0009446283   0017639369   0008193087   NTFS (0x07)
21:  04:01   0017639370   0048371714   0030732345   DOS Extended (0x05)
22:  -----   0017639370   0017639370   0000000001   Table #5
23:  -----   0017639371   0017639432   0000000062   Unallocated
24:  05:00   0017639433   0048371714   0030732282   Linux (0x83)
```