



# Drive Analysis in a Flash

Simson L. Garfinkel, Ph.D.

Associate Professor

Naval Postgraduate School

<http://simson.net/>

[slgarfin@nps.edu](mailto:slgarfin@nps.edu)



# This presentation is about research being performed at the Naval Postgraduate School



Location: Monterey, CA

Campus Size: 627 acres

1500 Students:

- ⇒ US Military (All 5 services)
- ⇒ US Civilian (SFS & SMART)
- ⇒ Foreign Military (30 countries)

4 Schools:

- ⇒ Business & Public Policy
- ⇒ Engineering & Applied Sciences
- ⇒ Operational & Information Sciences
- ⇒ International Graduate Studies



# Hard drive forensics is facing the I/O Barrier.

## Seagate Barracuda 7200.11 SATA 3Gb/s 1.5-TB Hard Drive

- ⇒ 7200 rpm
- ⇒ 120 MB/s sustained data rate
- ⇒ 32-MB cache
- ⇒ \$125.90 (including shipping)



$$\frac{1,500,301,910,016 \text{ bytes}}{120,000,000 \text{ bytes/s}} = 12,502 \text{ seconds} = 3 \text{ hours, } 28 \text{ min}$$

Disk performance is determined by data transfer rate and by seek time.

The Seagate Barracuda has a "random read seek time" of <8.5 msec.

Seek time = travel time + stabilization time.



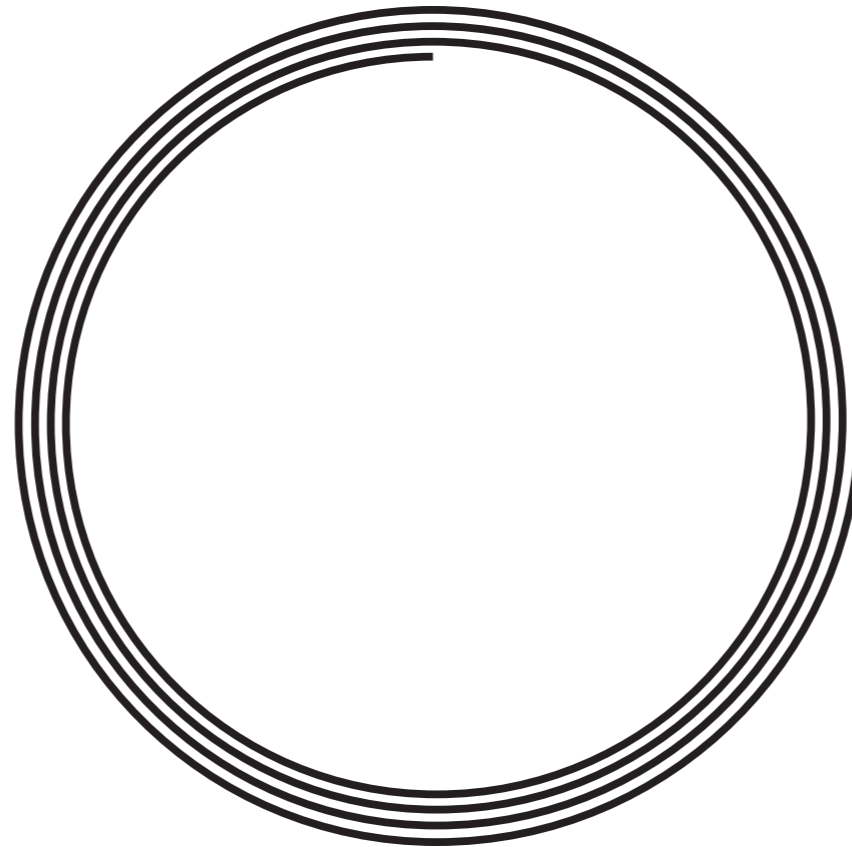
$1 \text{ sec} \div 8.5 \text{ msec/seek} \approx 1200 \text{ seeks/sec}$

Reading all of the data in 3.5 hours requires minimizing seeks.

Start at sector #0

End at sector #2,930,277,167

208 minutes x 7200 RPM = 1,500,300 revolutions

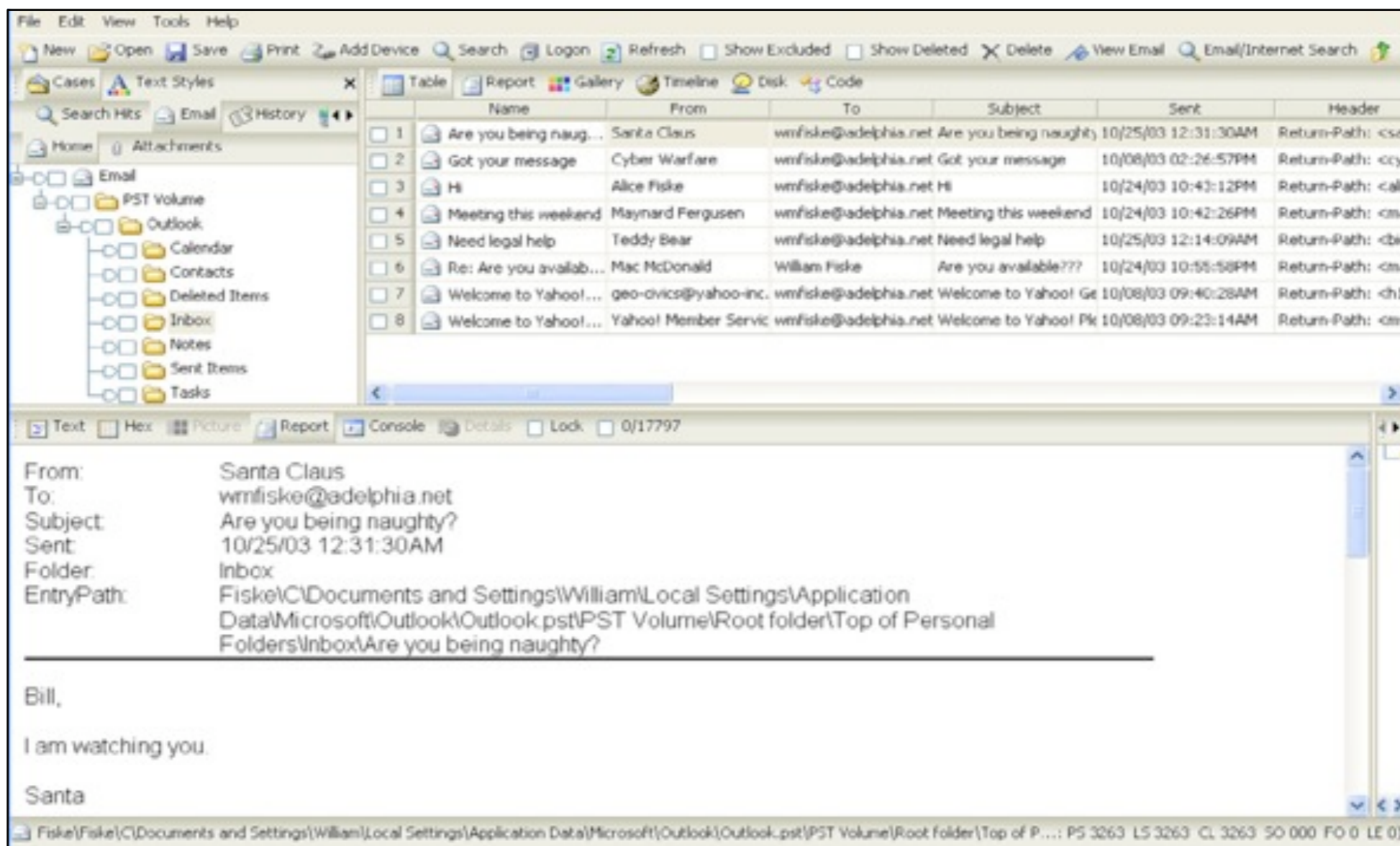


Unfortunately, data on hard drives is not laid out in consecutive sectors.

# Most forensic programs read file-by-file:

Guidance Software's EnCase is designed to:

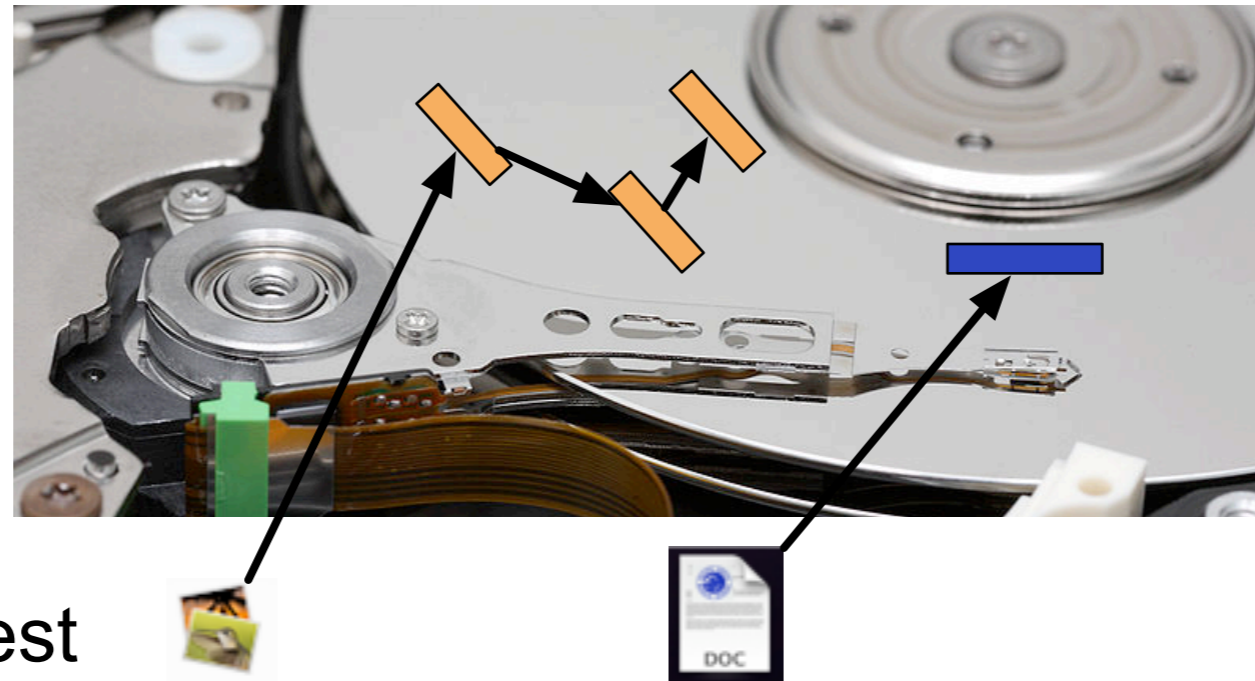
- ⇒ Recover deleted photos & documents.
- ⇒ Recover email messages



# Information is stored in different areas of the disk



# Information is stored in different areas of the disk

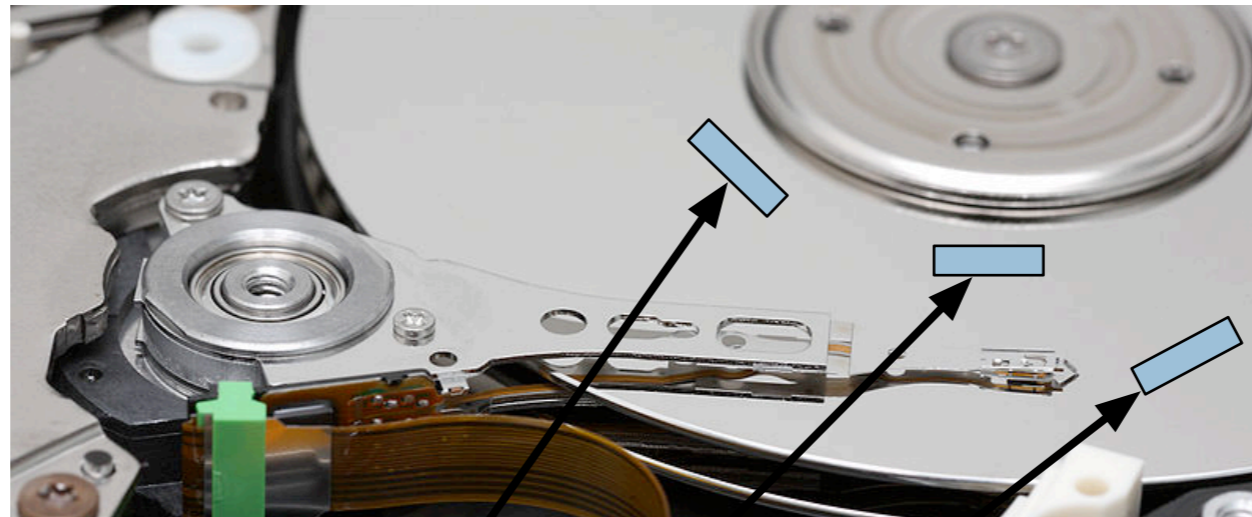


Documents of interest are stored in many areas.

Some documents are "fragmented" into multiple locations



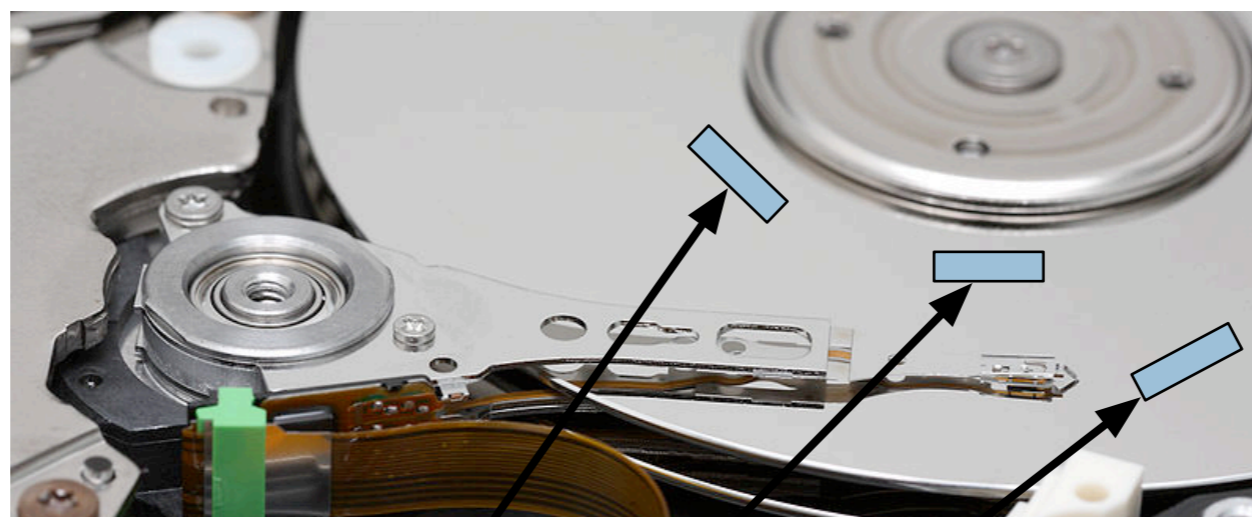
# Information is stored in different areas of the disk



▶ Desktop	Today, 7:15 PM
▶ Pictures	Today, 2:31 PM
unison.log	Today, 9:12 AM
▶ current	Yesterday, 9:10 AM
▶ Documents	May 22, 2009, 10:57 PM
userconftemplate.ppt	May 21, 2009, 10:41 PM
userconftemplate	May 21, 2009, 10:40 PM
ppt_barchart.pdf	May 21, 2009, 10:01 PM
▶ Downloads	May 21, 2009, 8:10 PM
▶ arch	May 18, 2009, 8:32 PM
▶ NetBeansProjects	May 17, 2009, 9:13 PM

Directories are stored in many locations.

# Information is stored in different areas of the disk



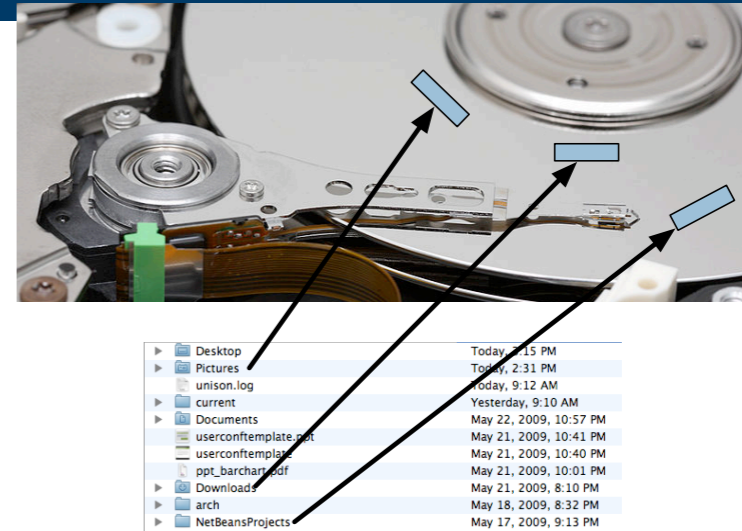
▶ Desktop	Today, 7:15 PM
▶ Pictures	Today, 2:31 PM
unison.log	Today, 9:12 AM
▶ current	Yesterday, 9:10 AM
▶ Documents	May 22, 2009, 10:57 PM
userconftemplate.ppt	May 21, 2009, 10:41 PM
userconftemplate	May 21, 2009, 10:40 PM
ppt_barchart.pdf	May 21, 2009, 10:01 PM
▶ Downloads	May 21, 2009, 8:10 PM
▶ arch	May 18, 2009, 8:32 PM
▶ NetBeansProjects	May 17, 2009, 9:13 PM

Directories are stored in many locations.

Because data is stored in different locations, it can take 10-30 hours to ingest a large disk.

# This talk presents two approaches for speeding disk forensics.

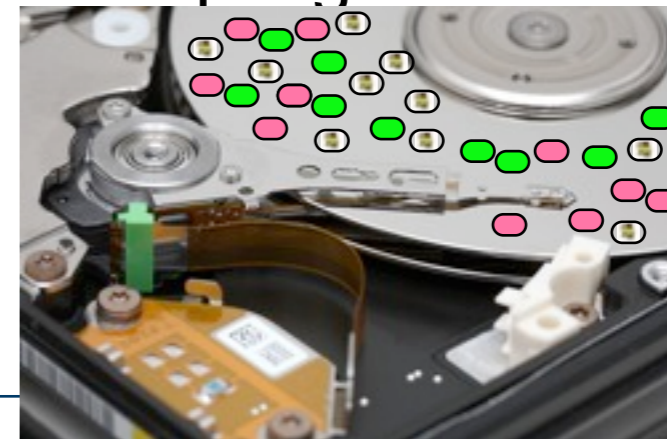
How is data arranged on a hard disk?



Approach #1: Speeding traditional forensics by considering disk order.



Approach #2: "Instant disk forensics" with drive sampling.

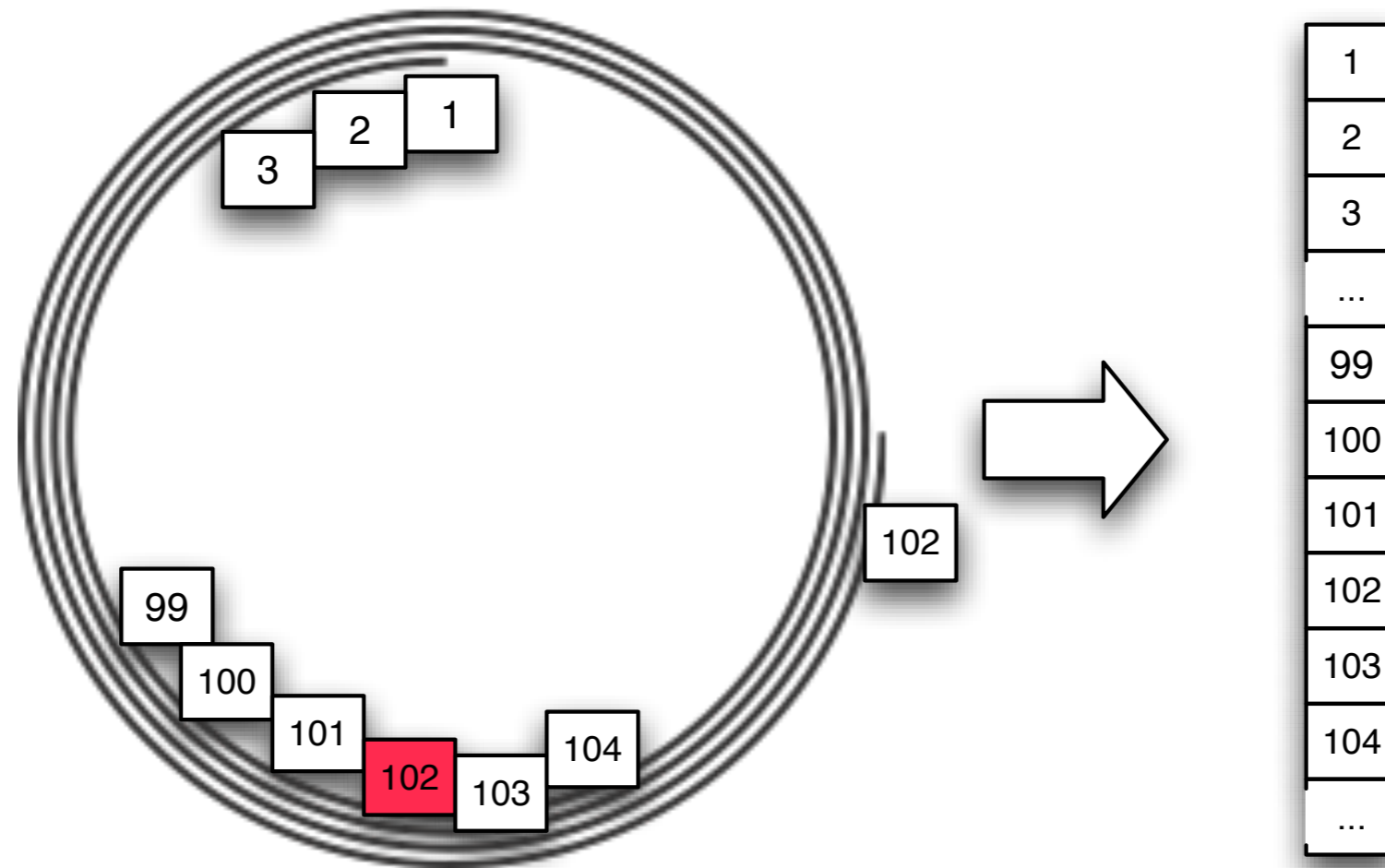




Speeding traditional disk  
forensics.

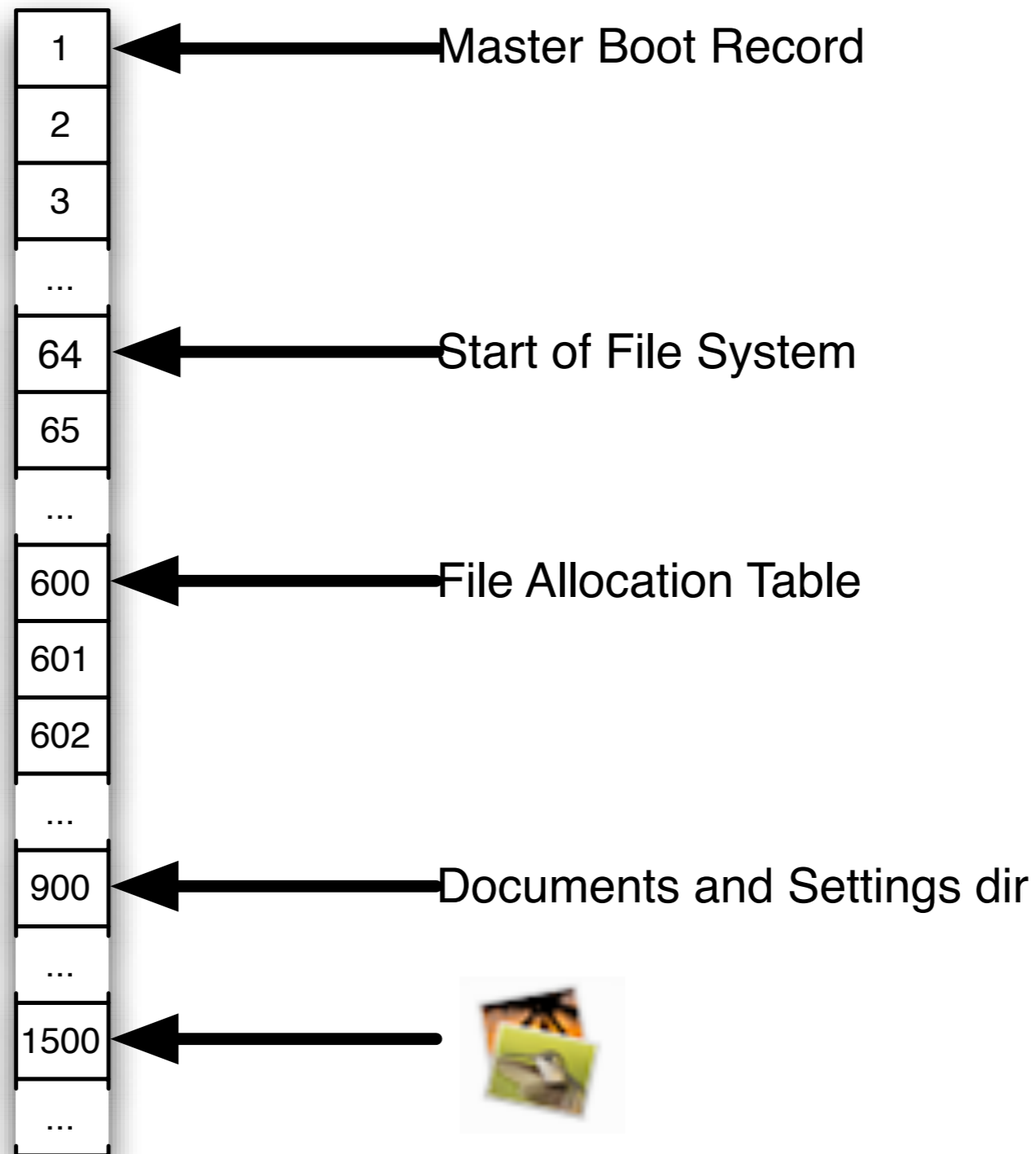


# Modern disk drives address sectors by *logical block number*.



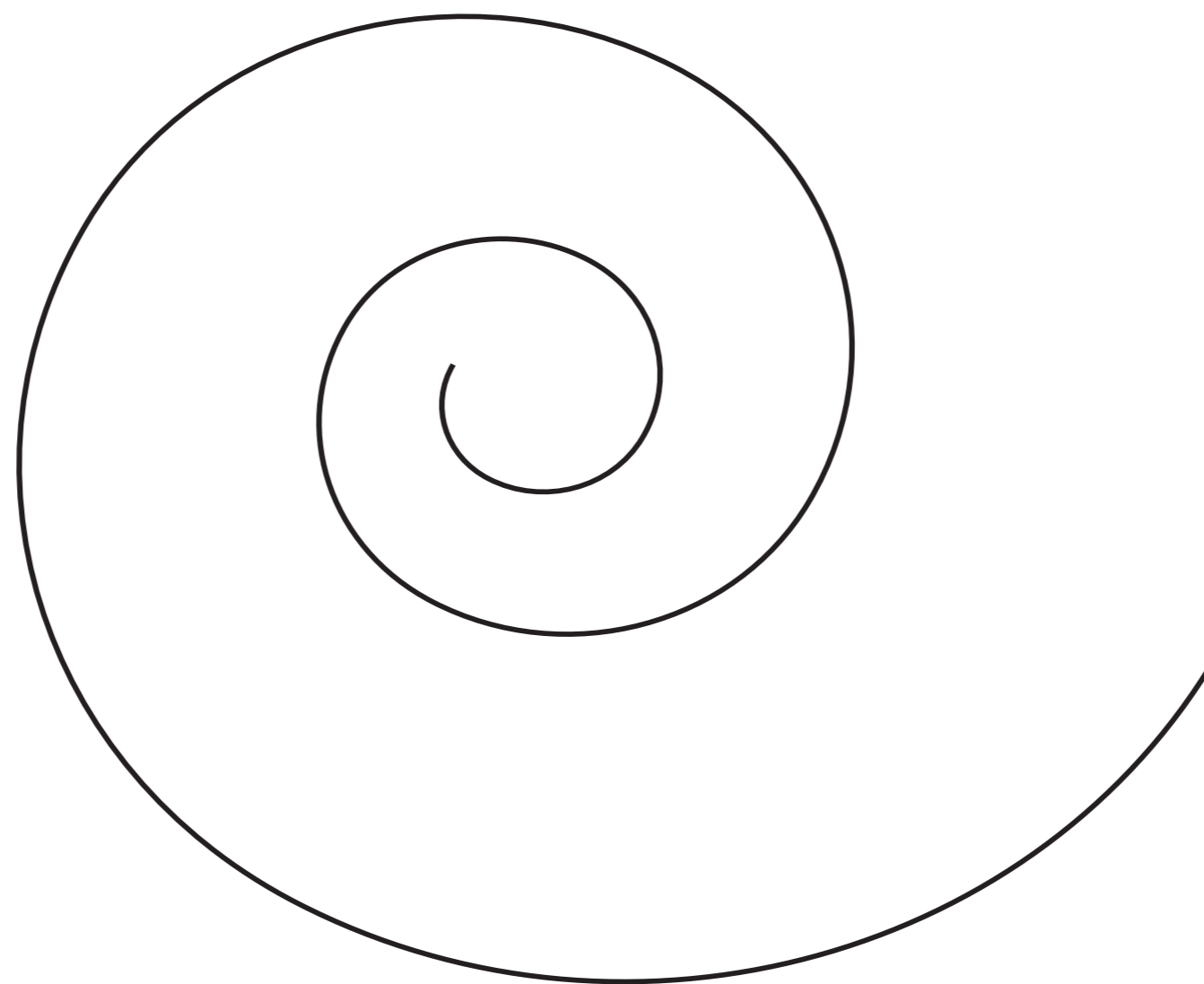
Bad sectors are transparently remapped.

# Different parts of the file system are stored at different blocks.



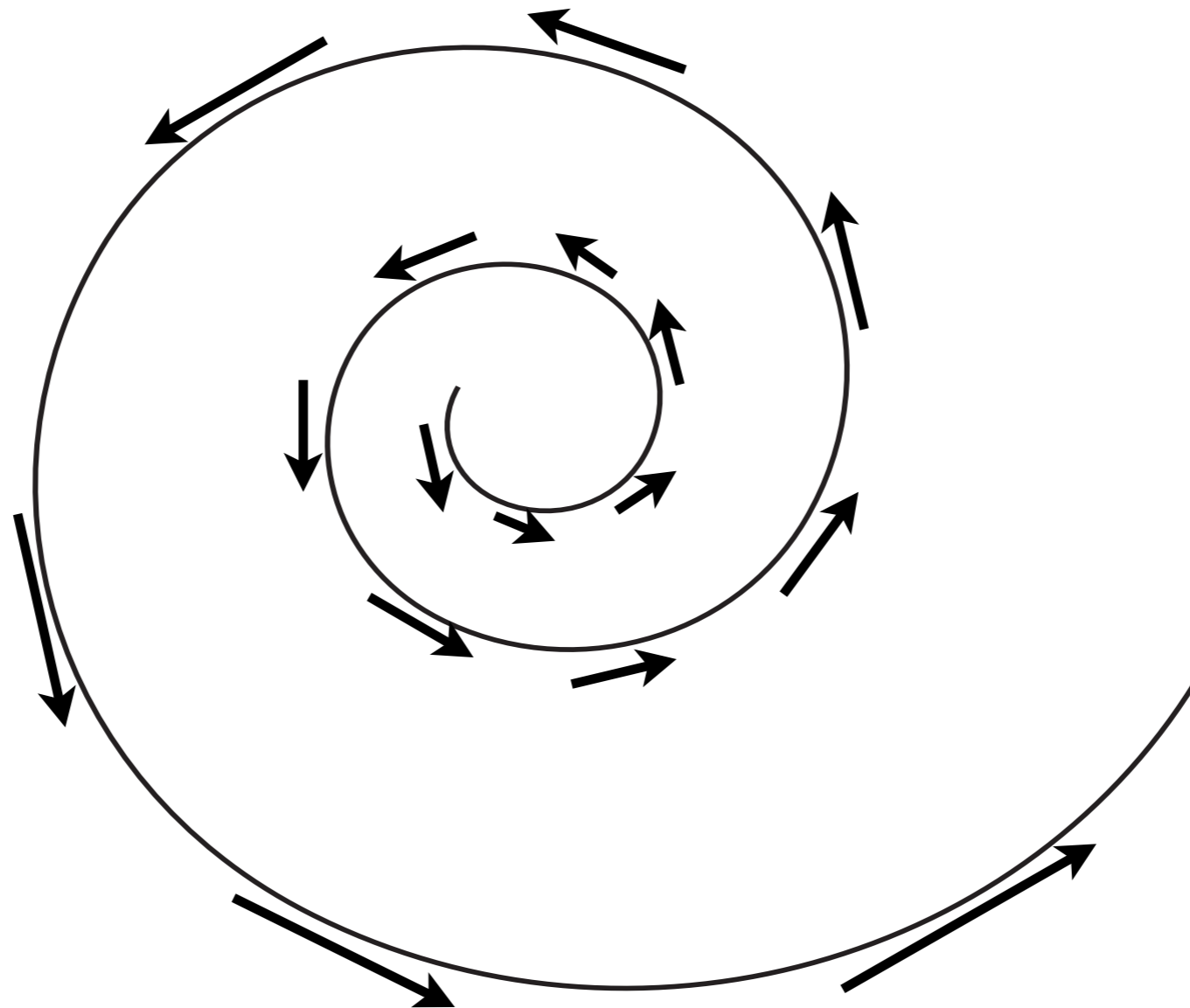
Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

Imaging is done at maximum disk transfer rate:



Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

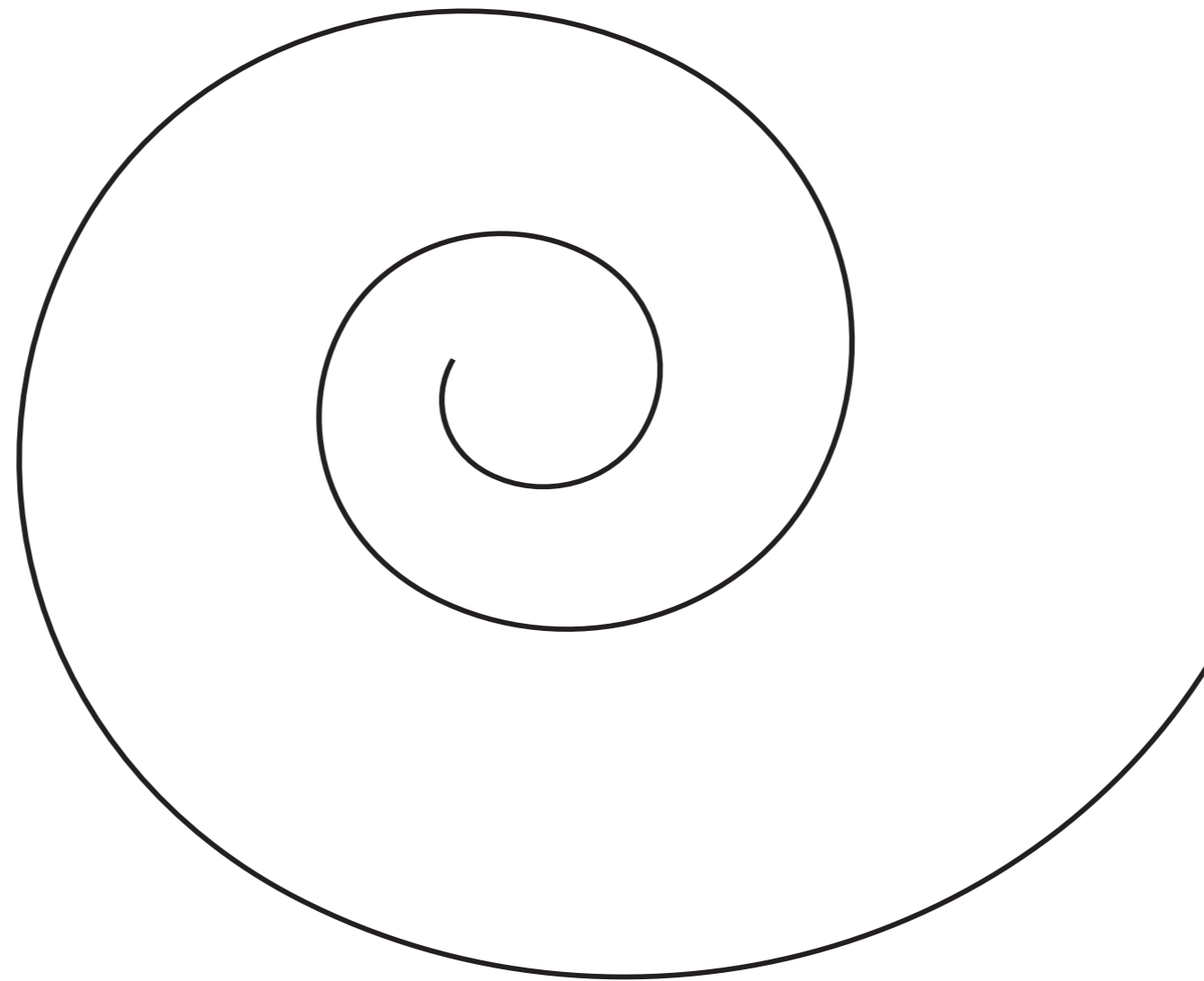
Imaging is done at maximum disk transfer rate:





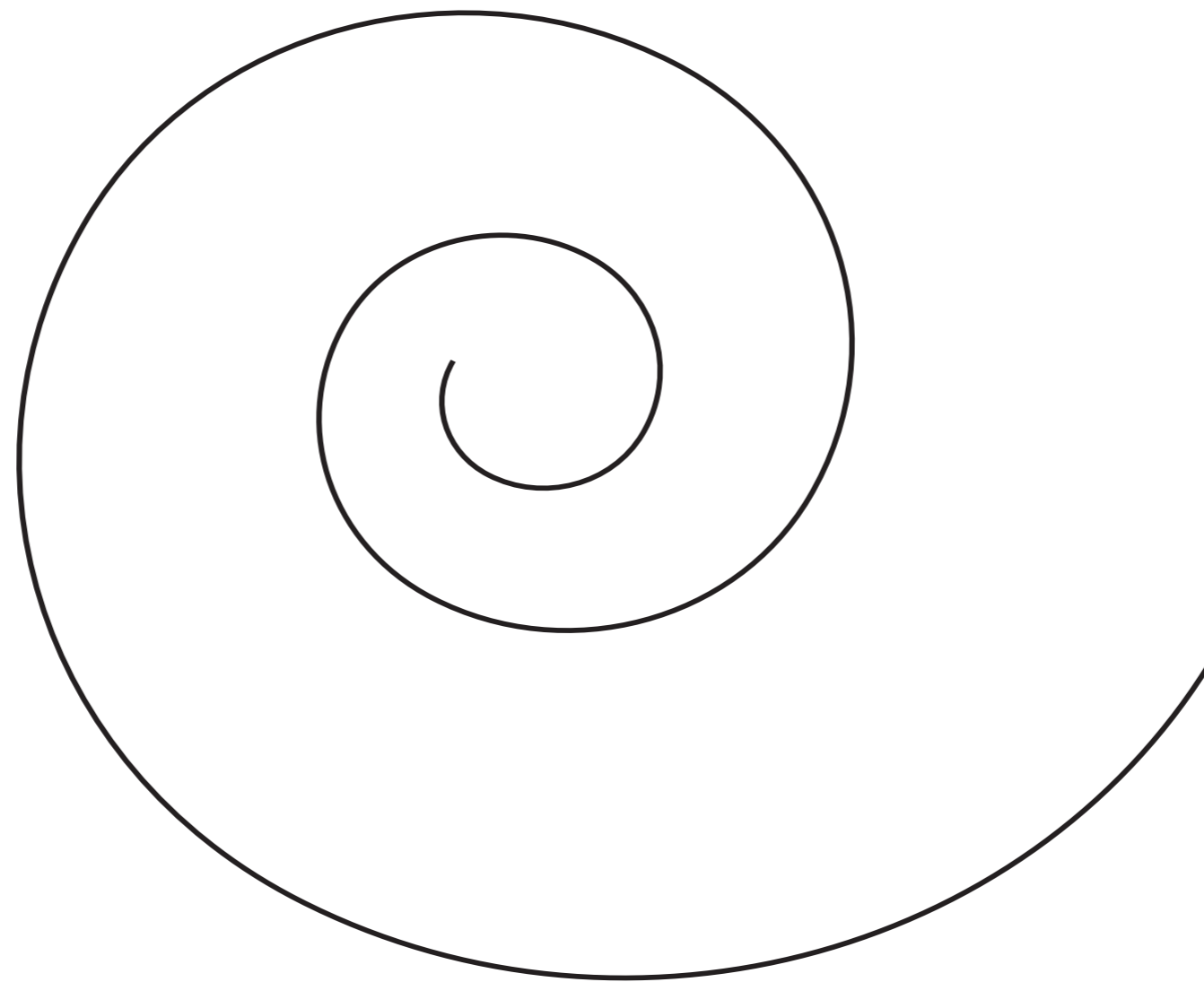
Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

Imaging is done at maximum disk transfer rate:



Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

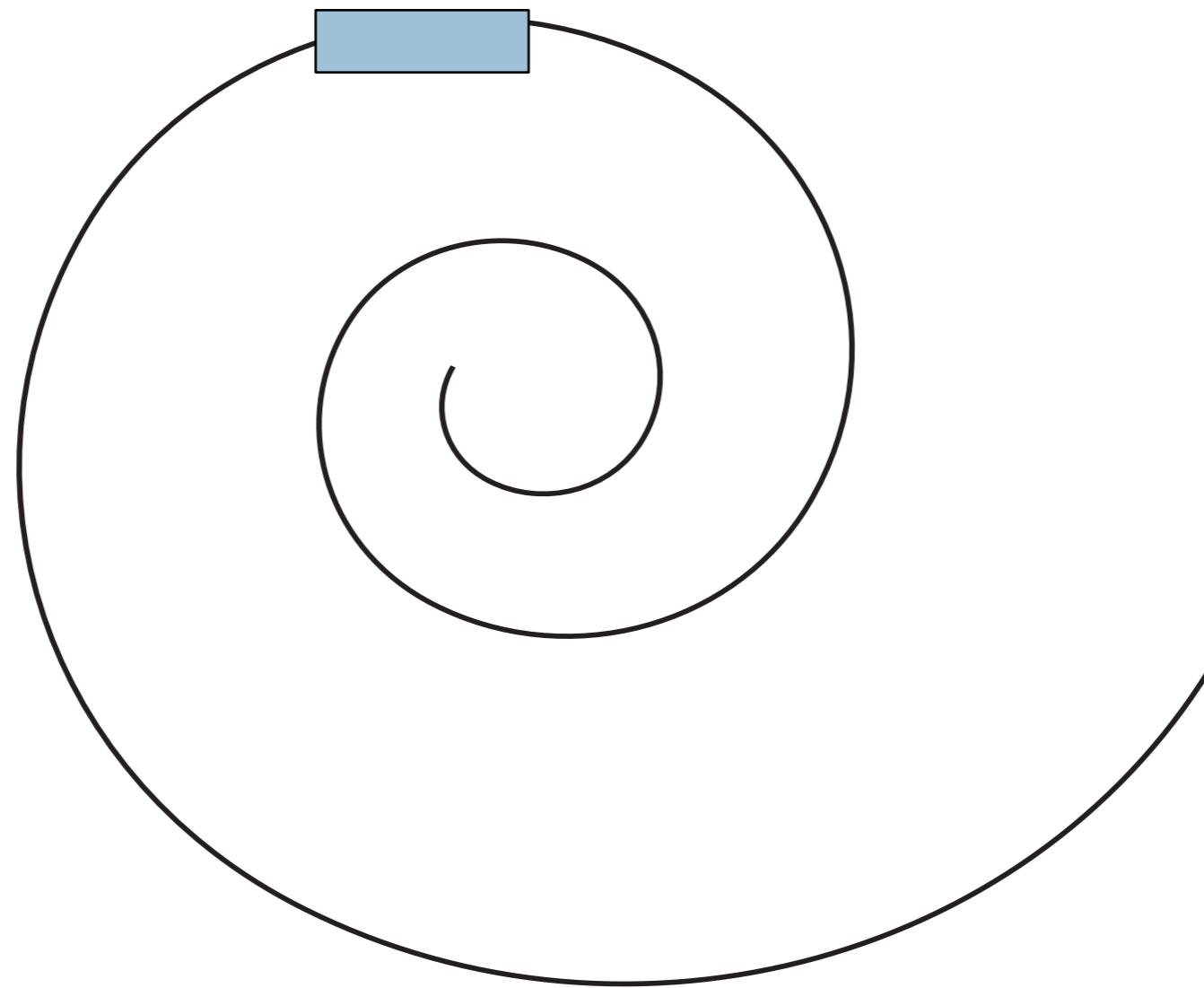
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

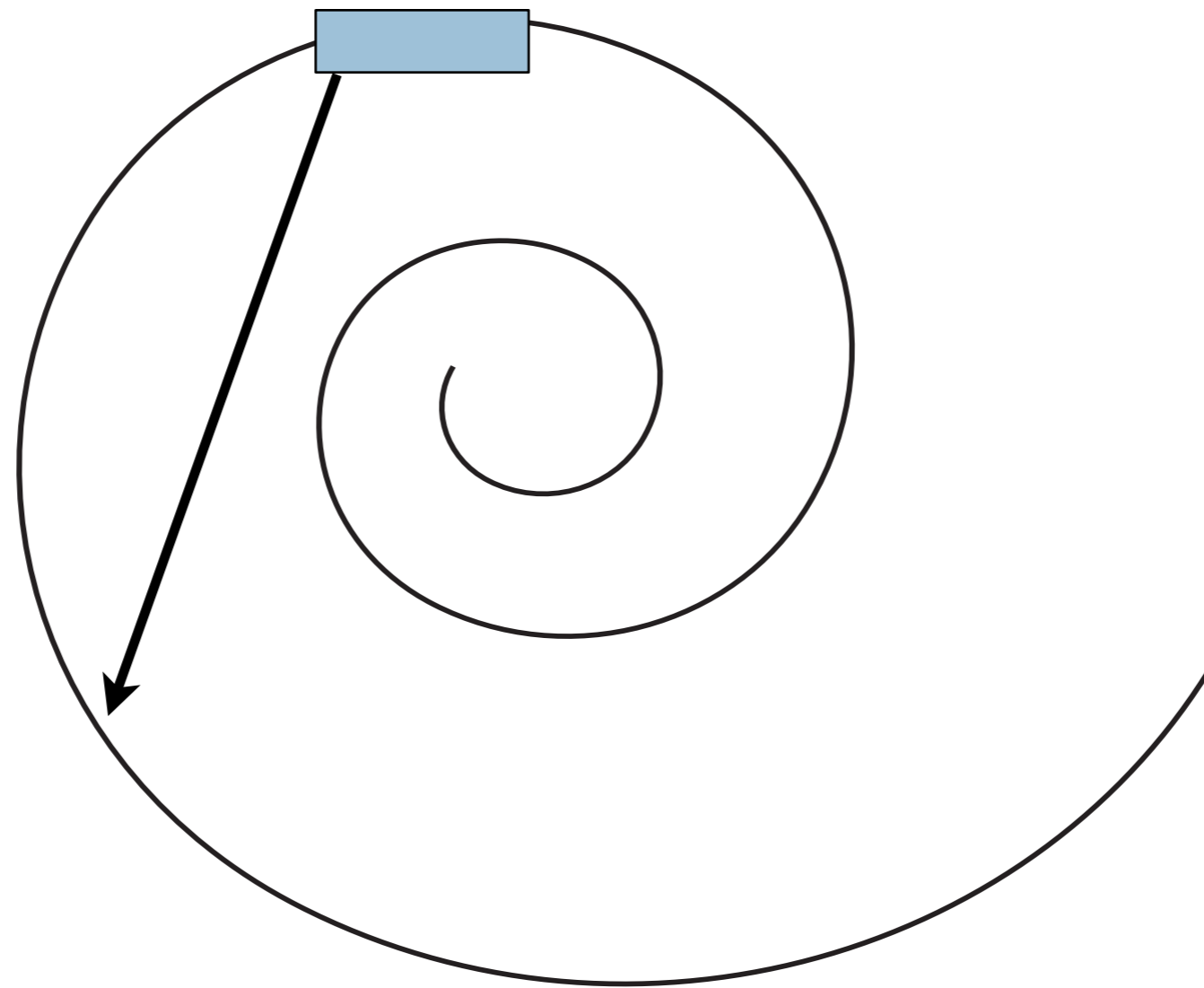
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

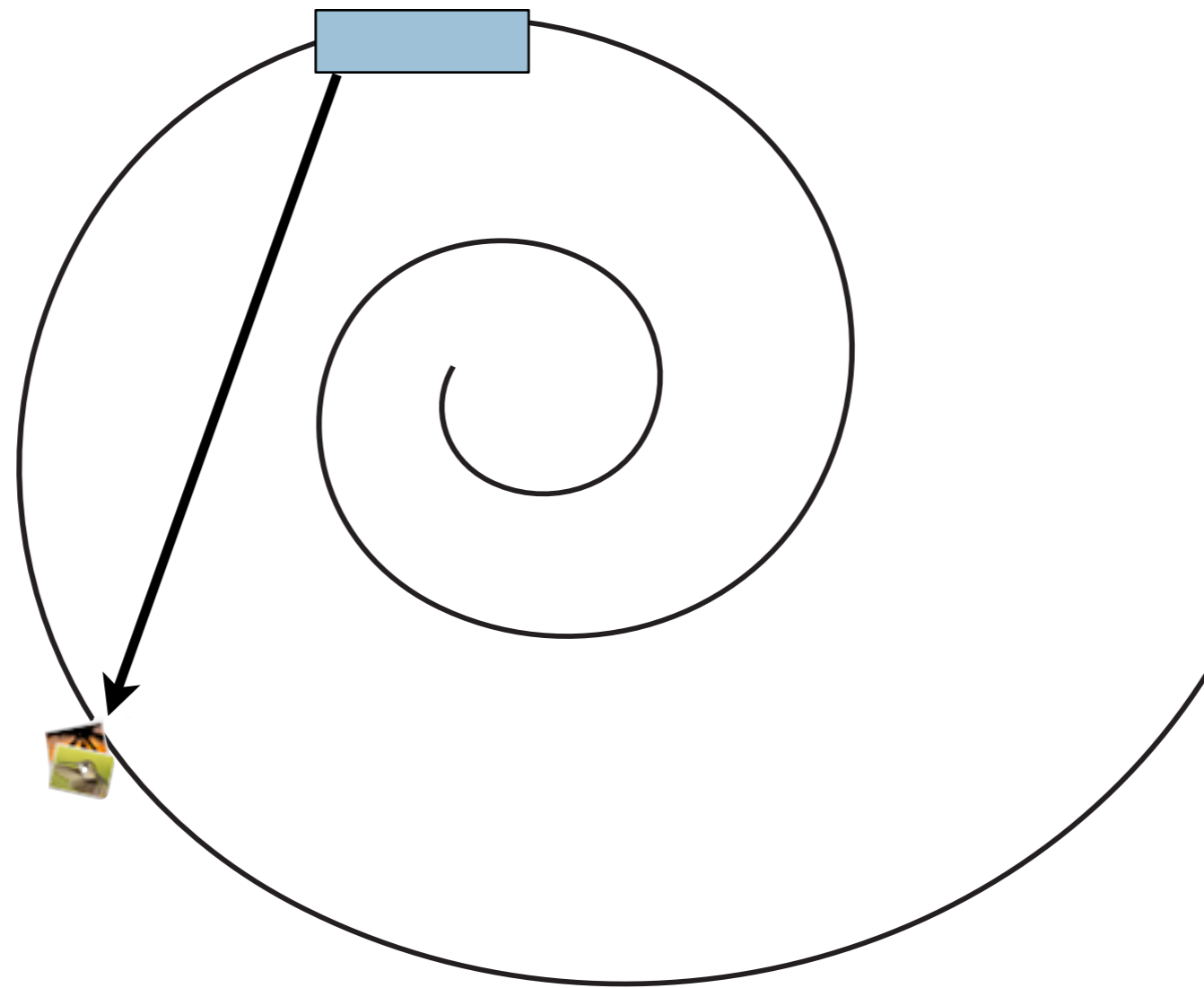
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

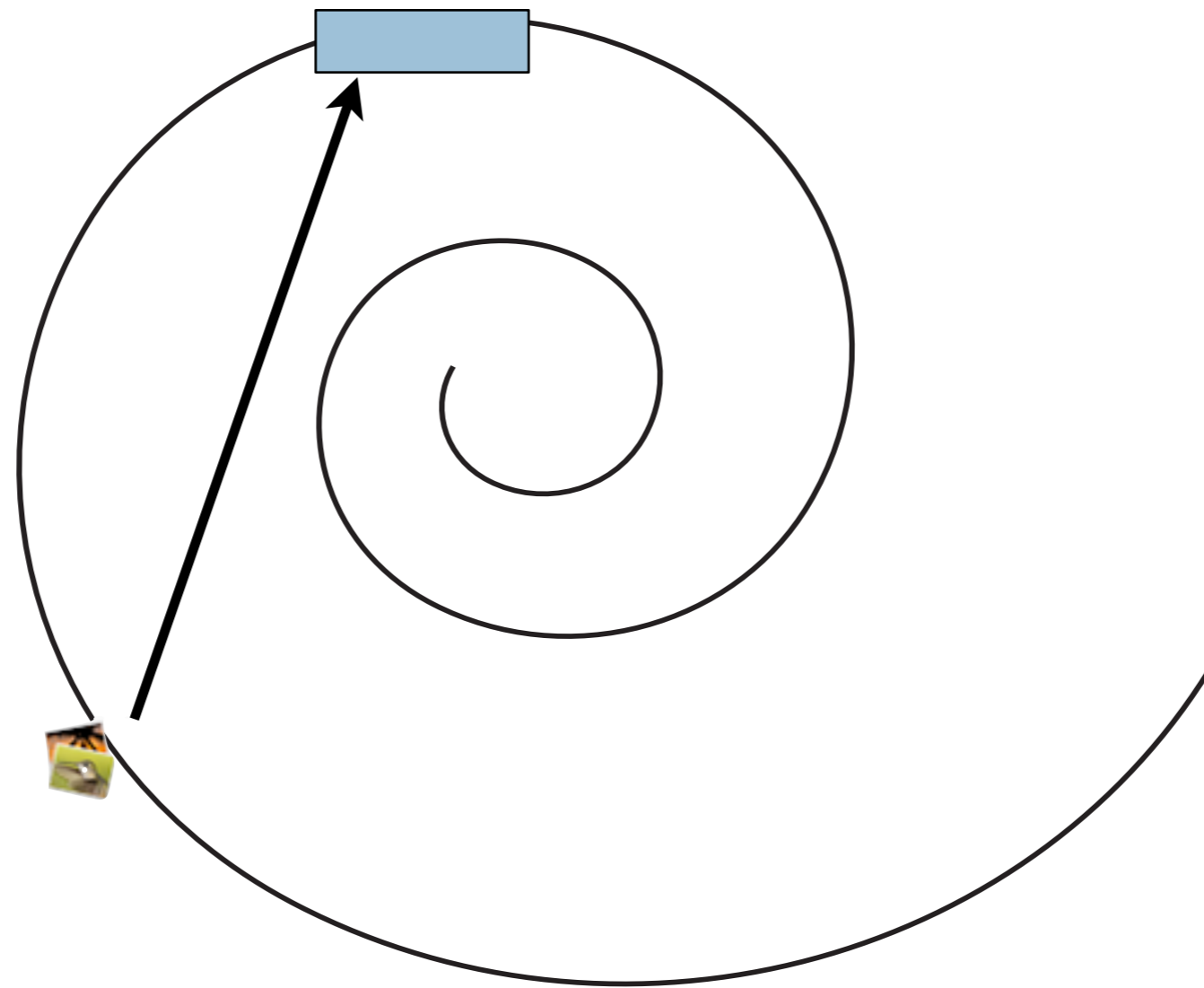
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

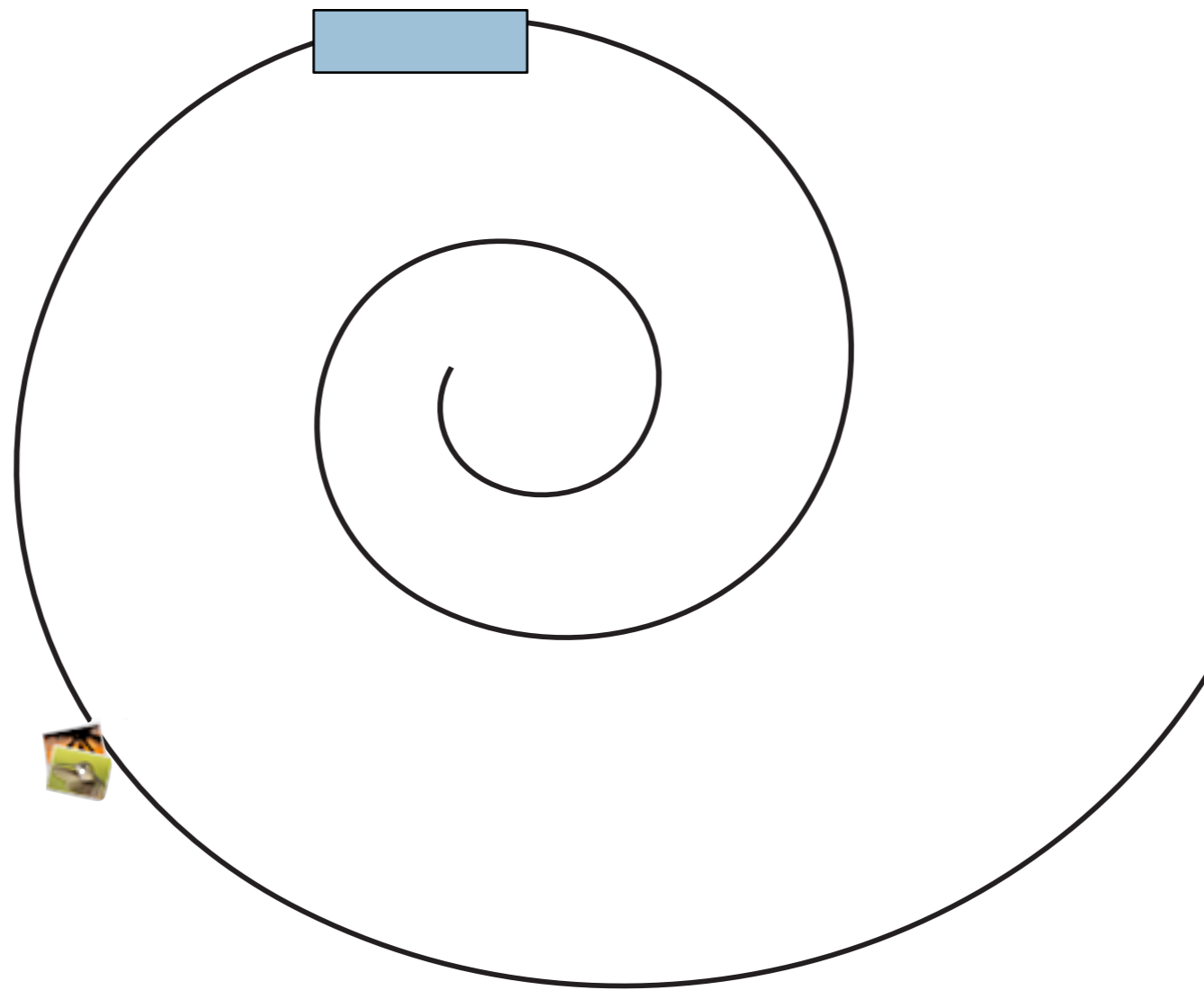
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

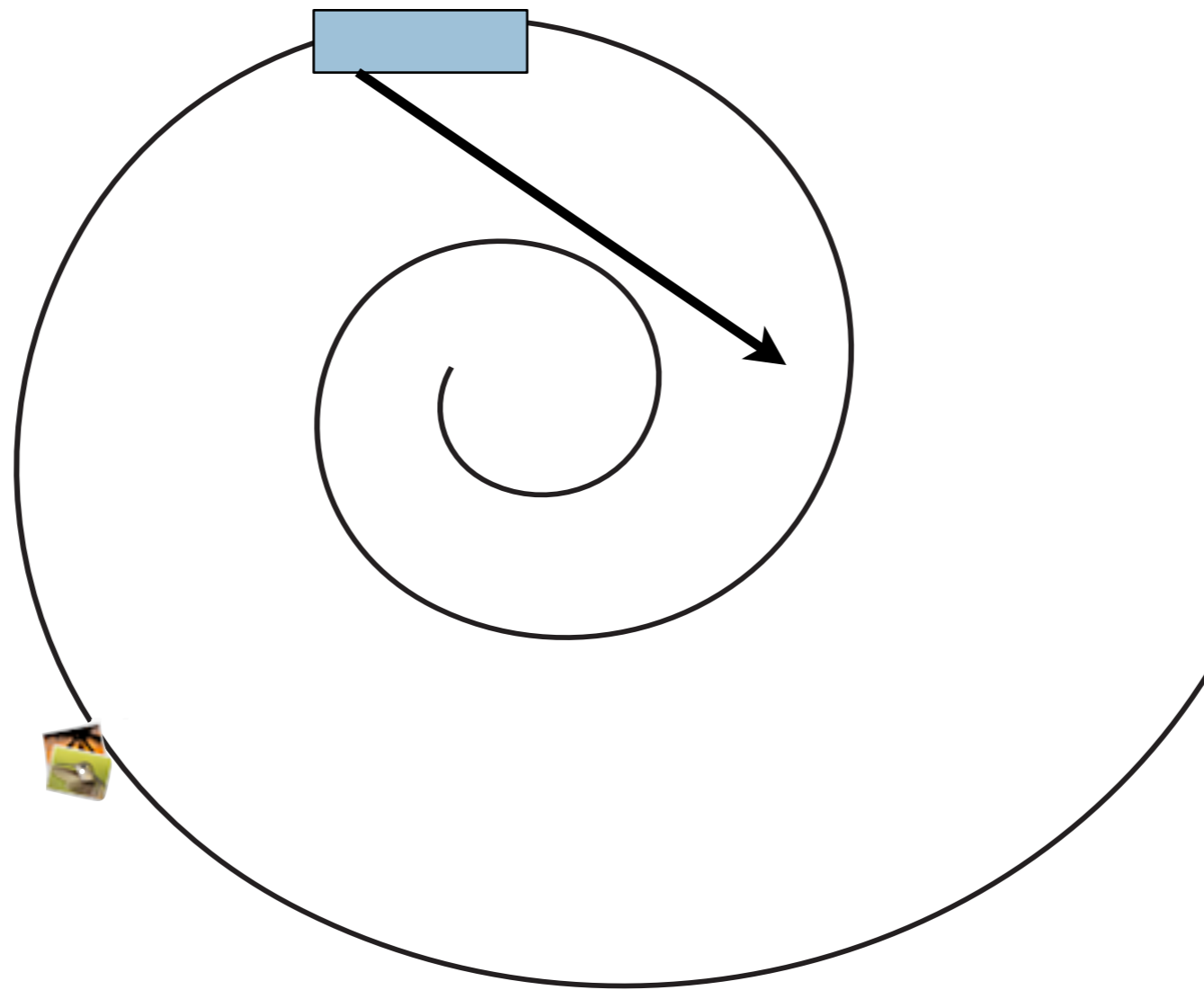
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

Imaging is done at maximum disk transfer rate:

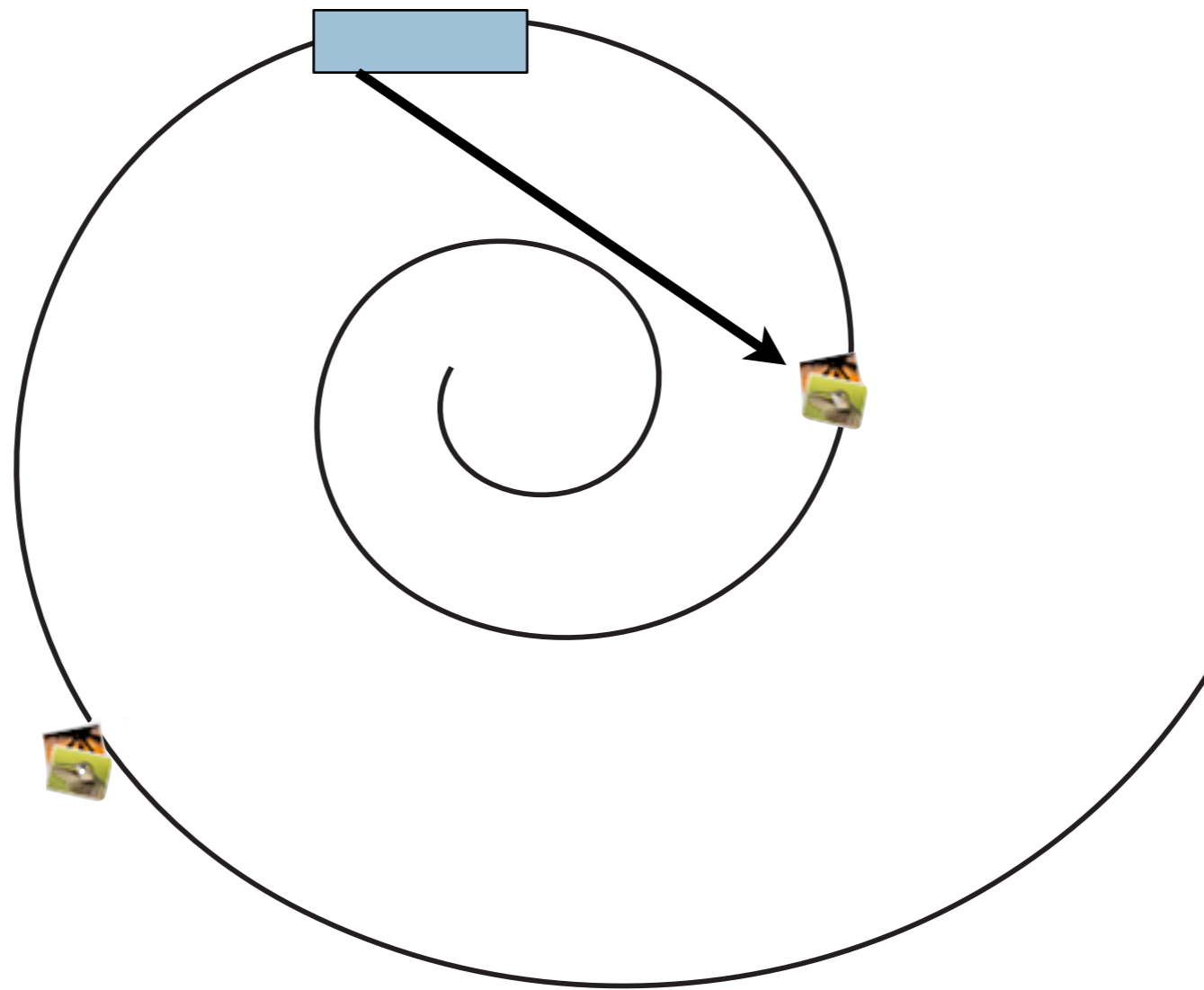


But ingesting is done directory-by-directory.



Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

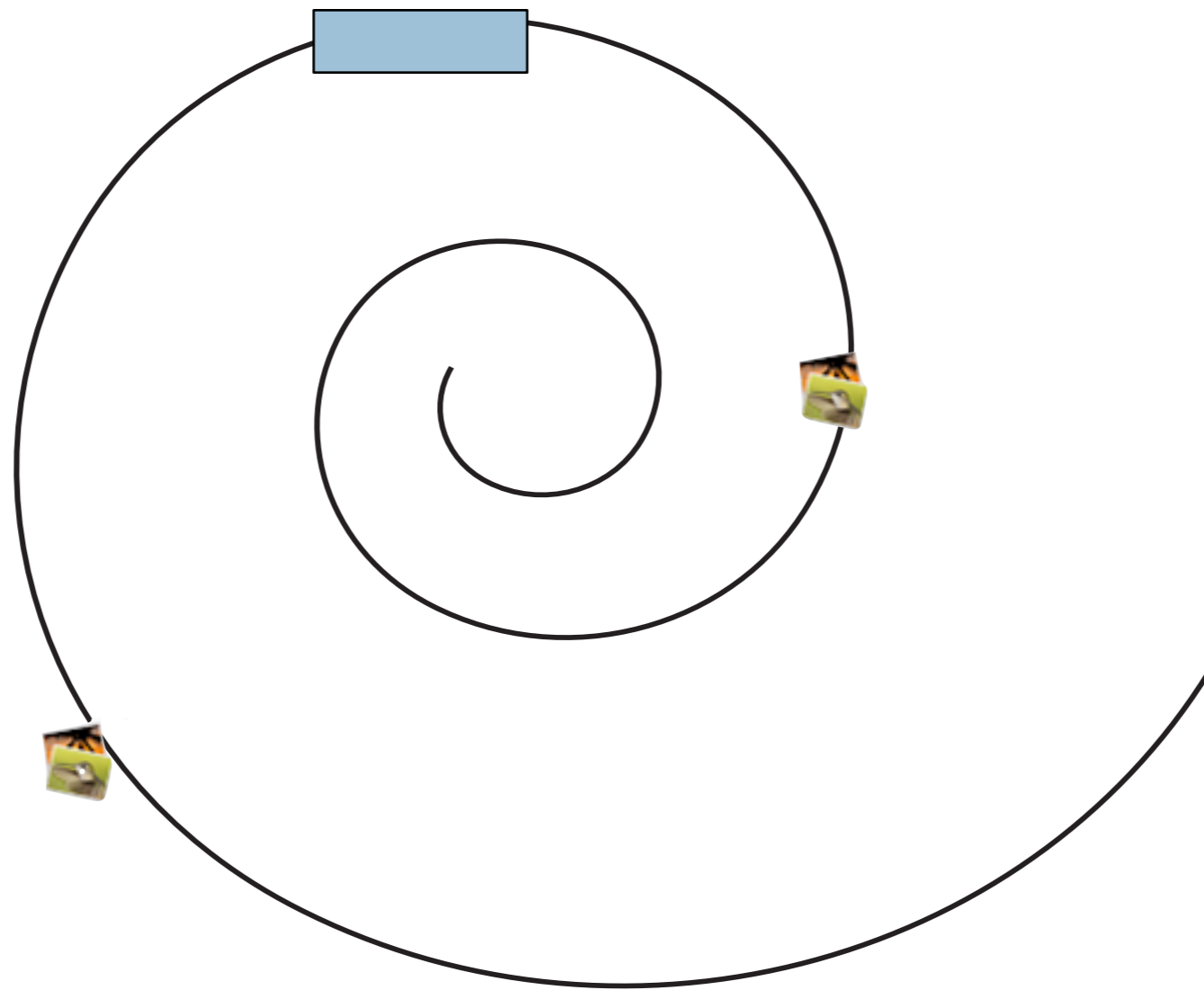
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

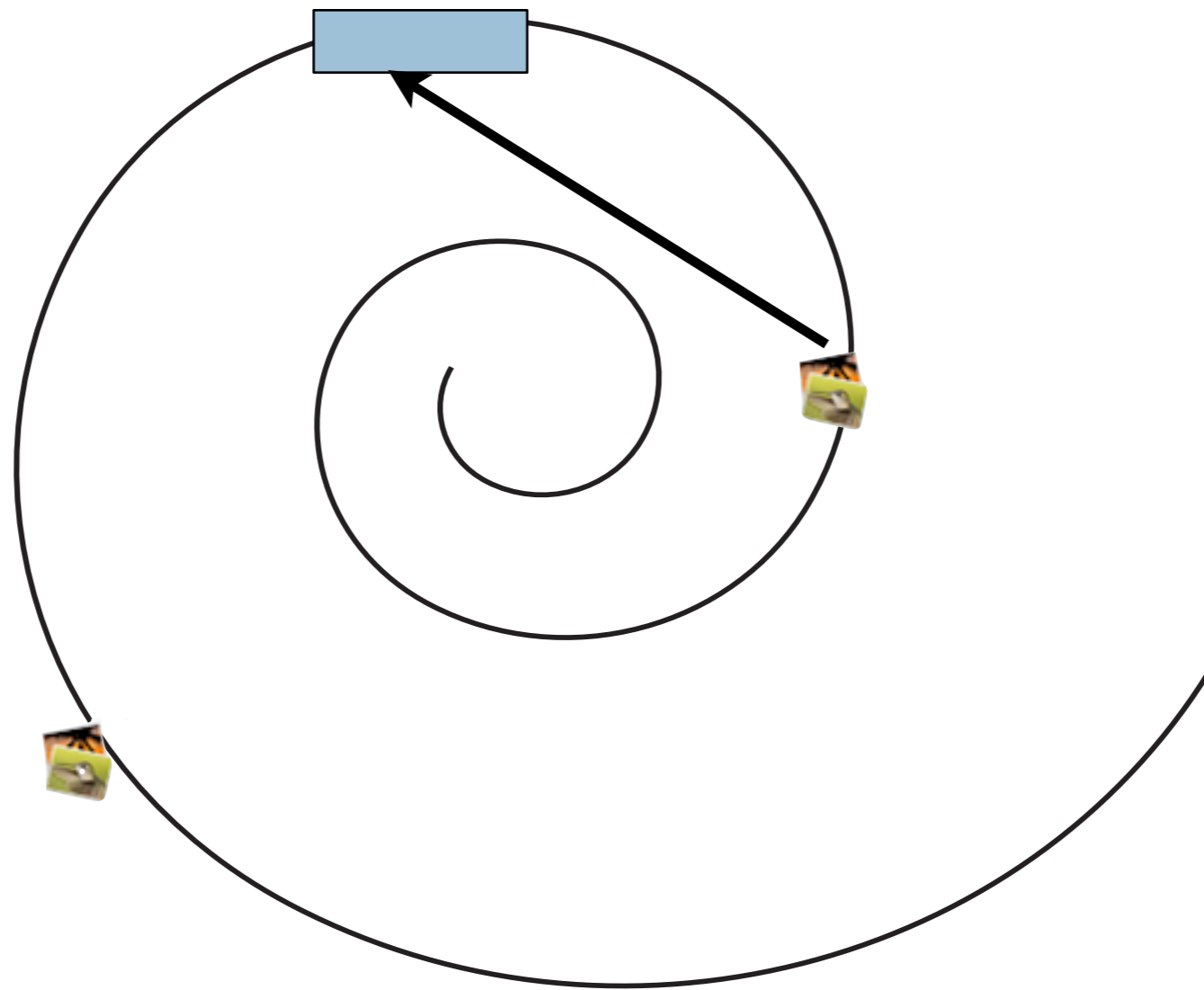
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

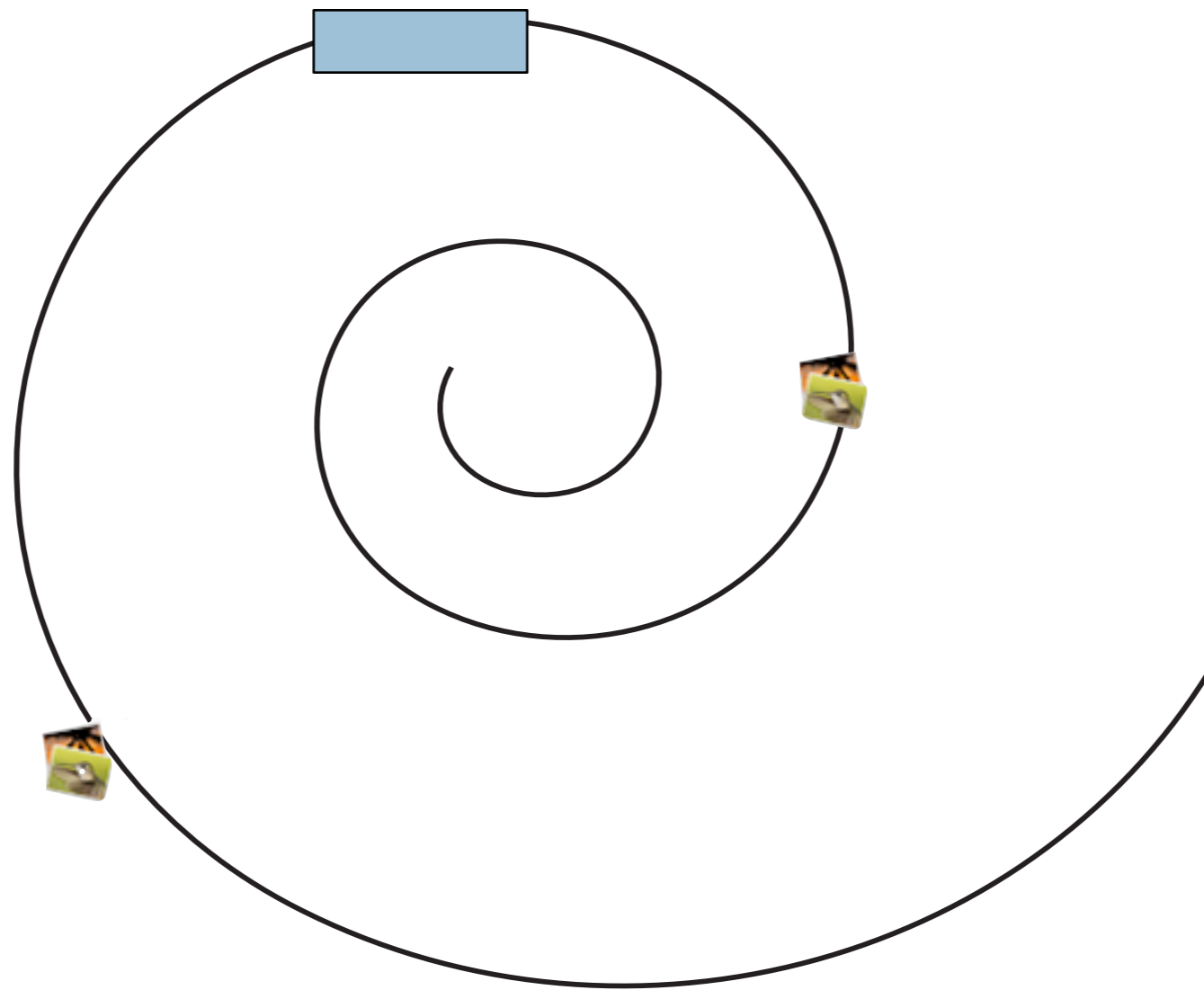
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

Most forensic programs *image* the disk,  
then *ingest* the image *directory-by-directory*.

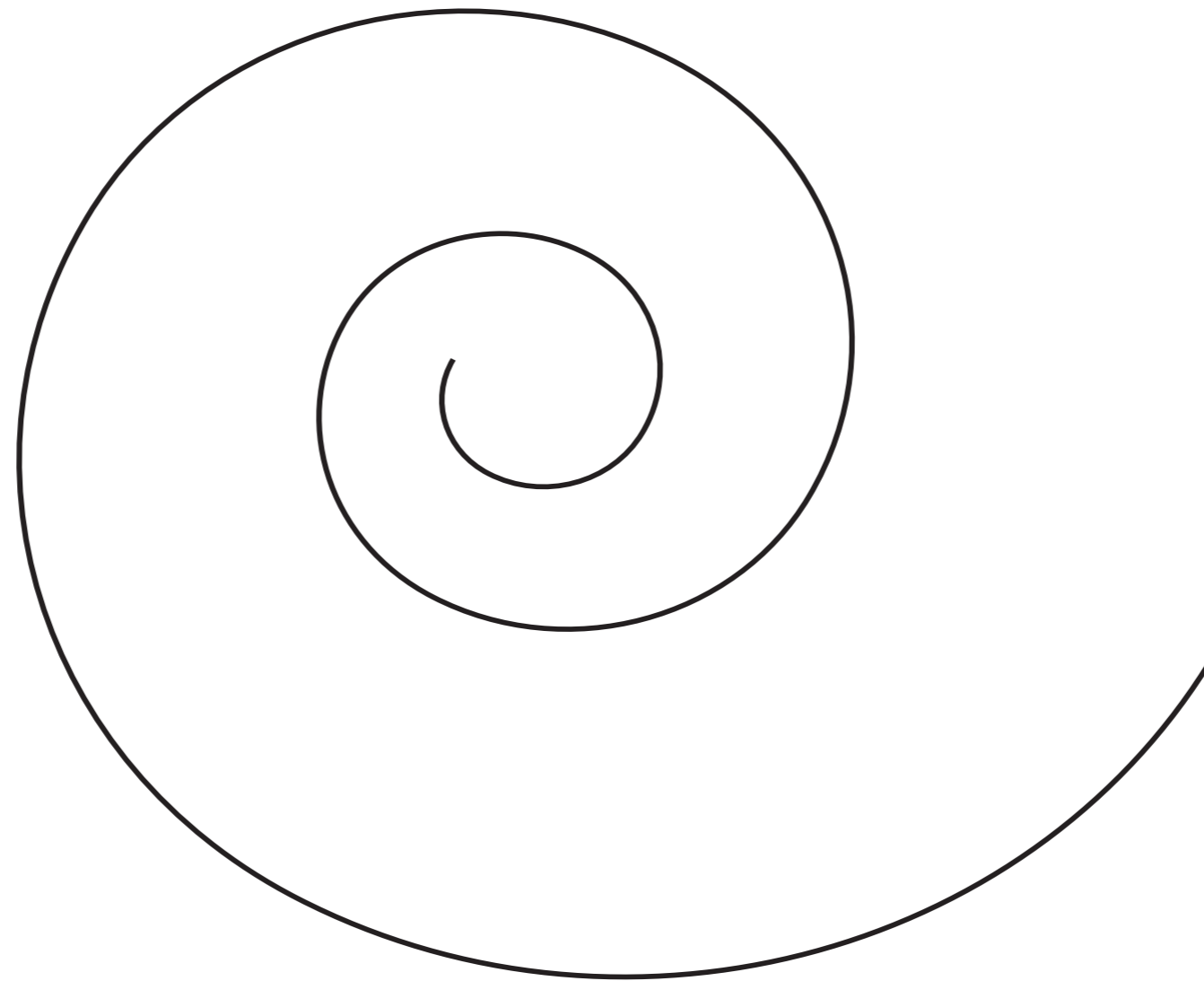
Imaging is done at maximum disk transfer rate:



But ingesting is done directory-by-directory.

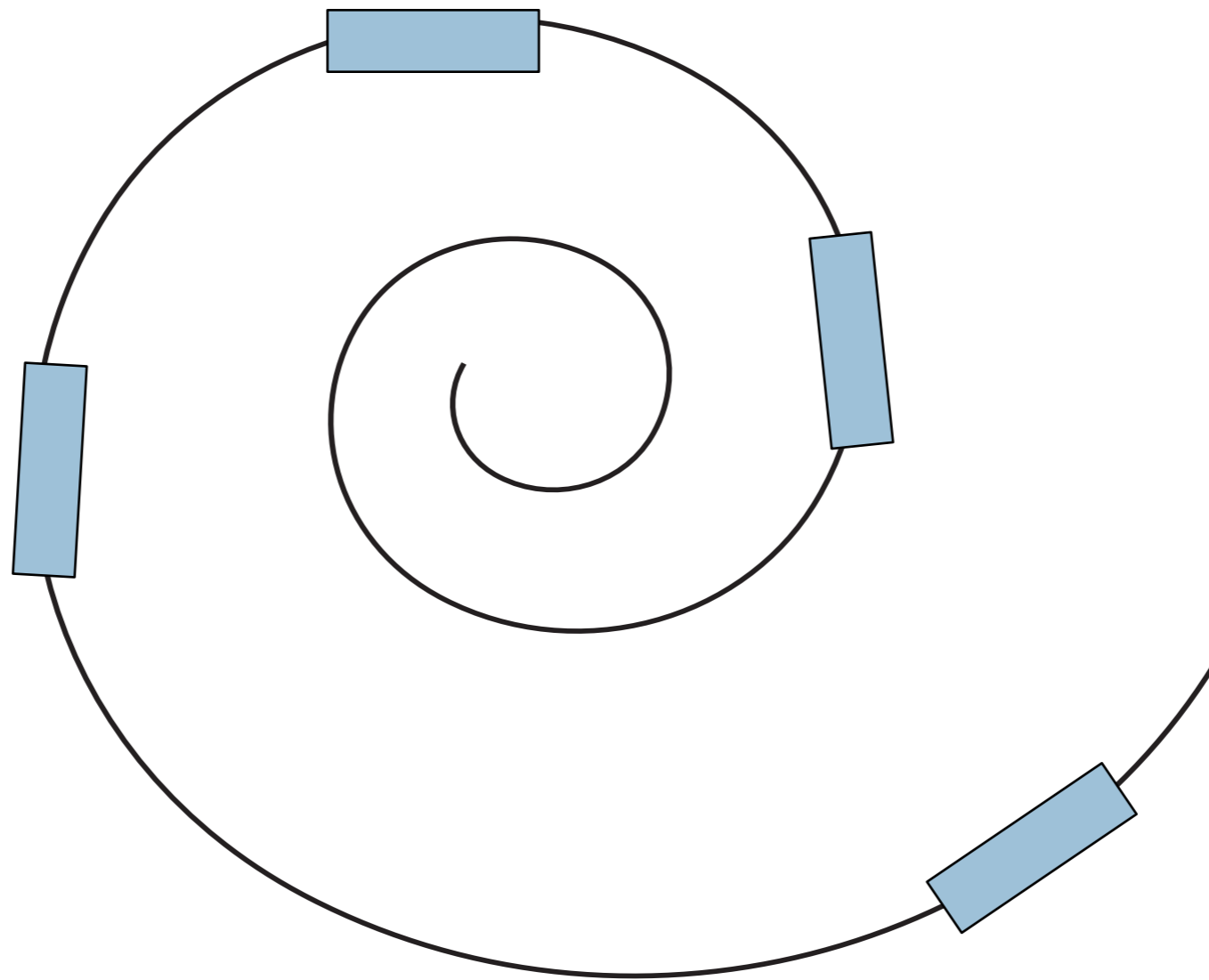
# There is room for improvement!

1 - Find all of the directories



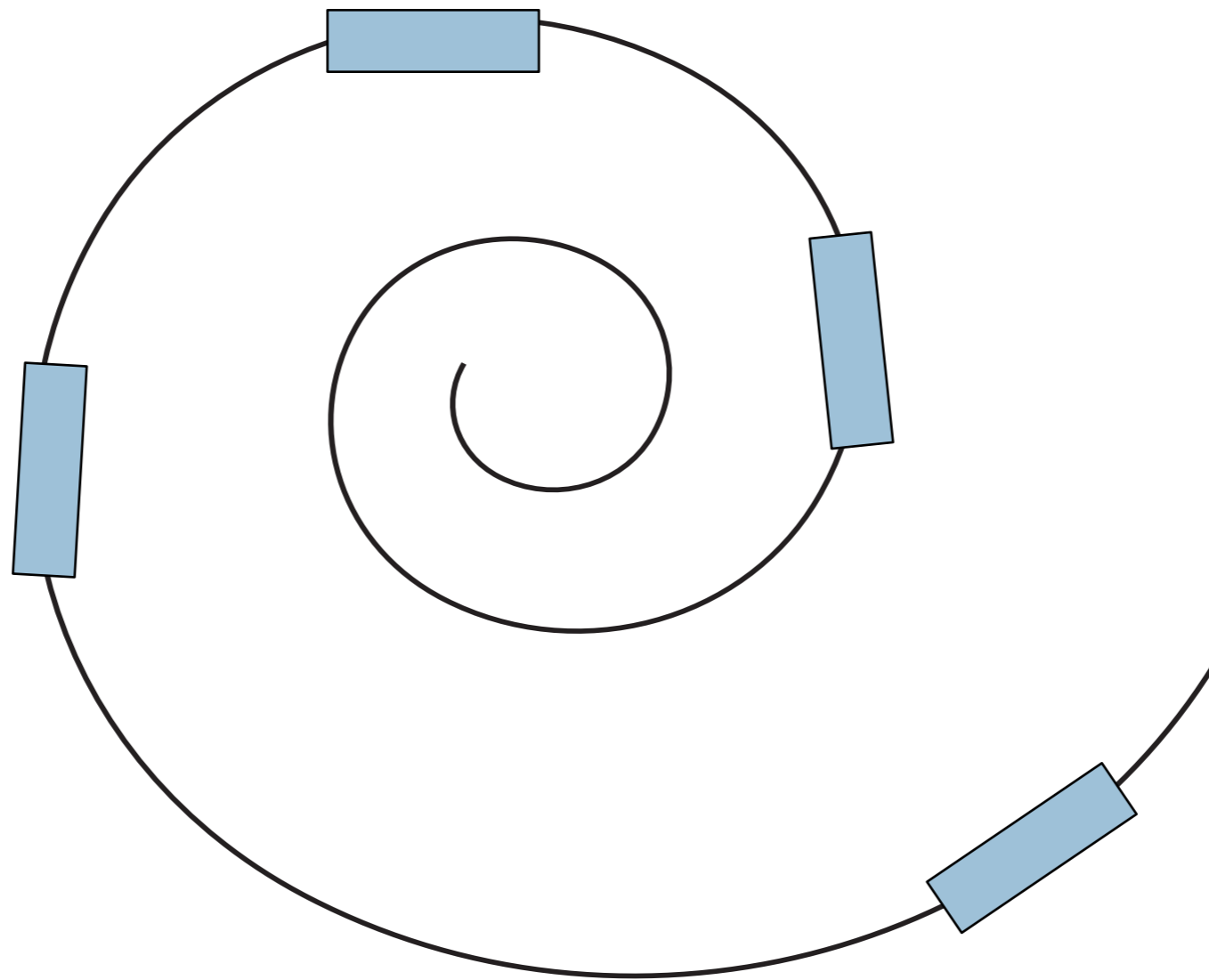
# There is room for improvement!

1 - Find all of the directories



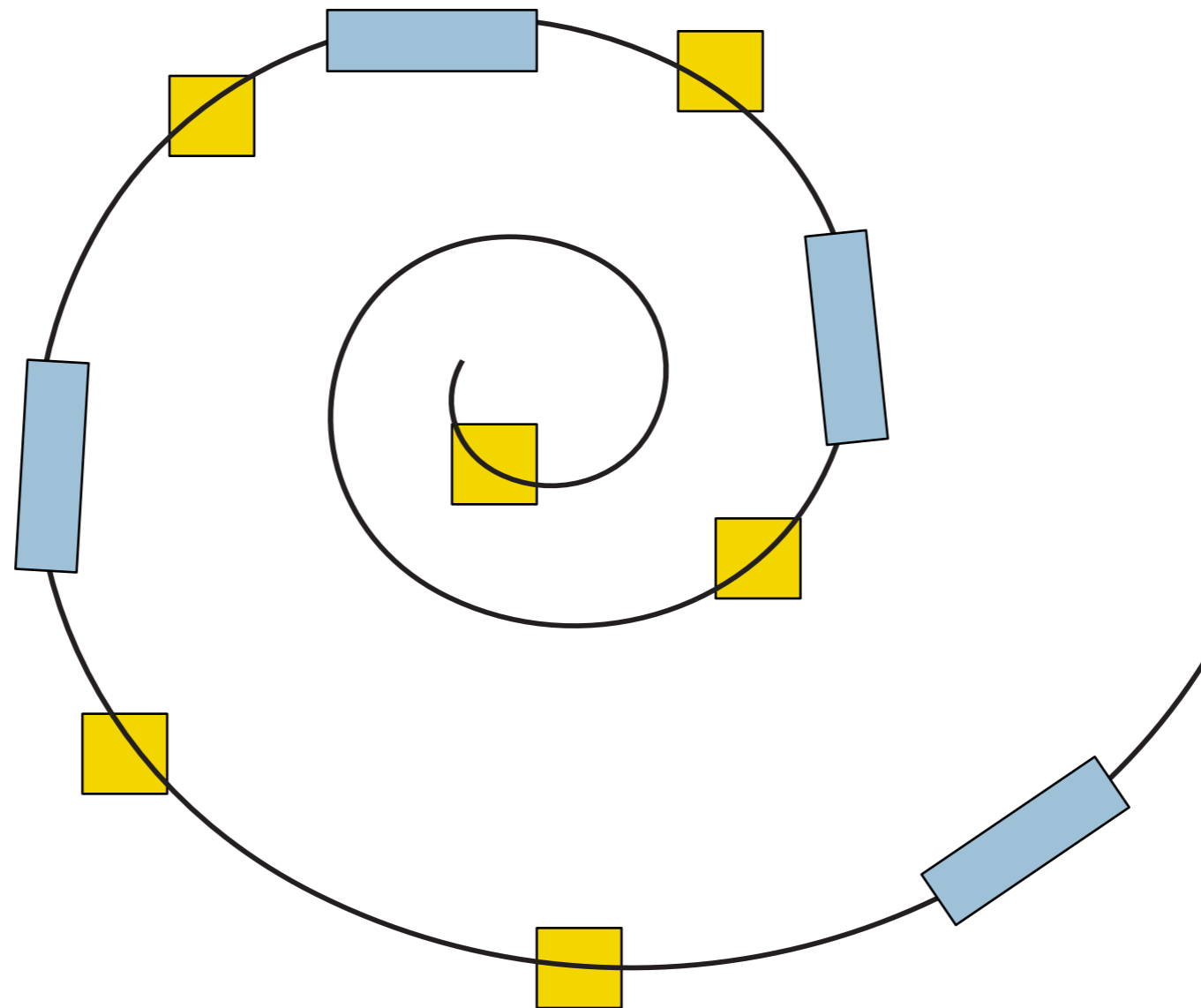
# There is room for improvement!

- 1 - Find all of the directories
- 2 - Locate all the files



# There is room for improvement!

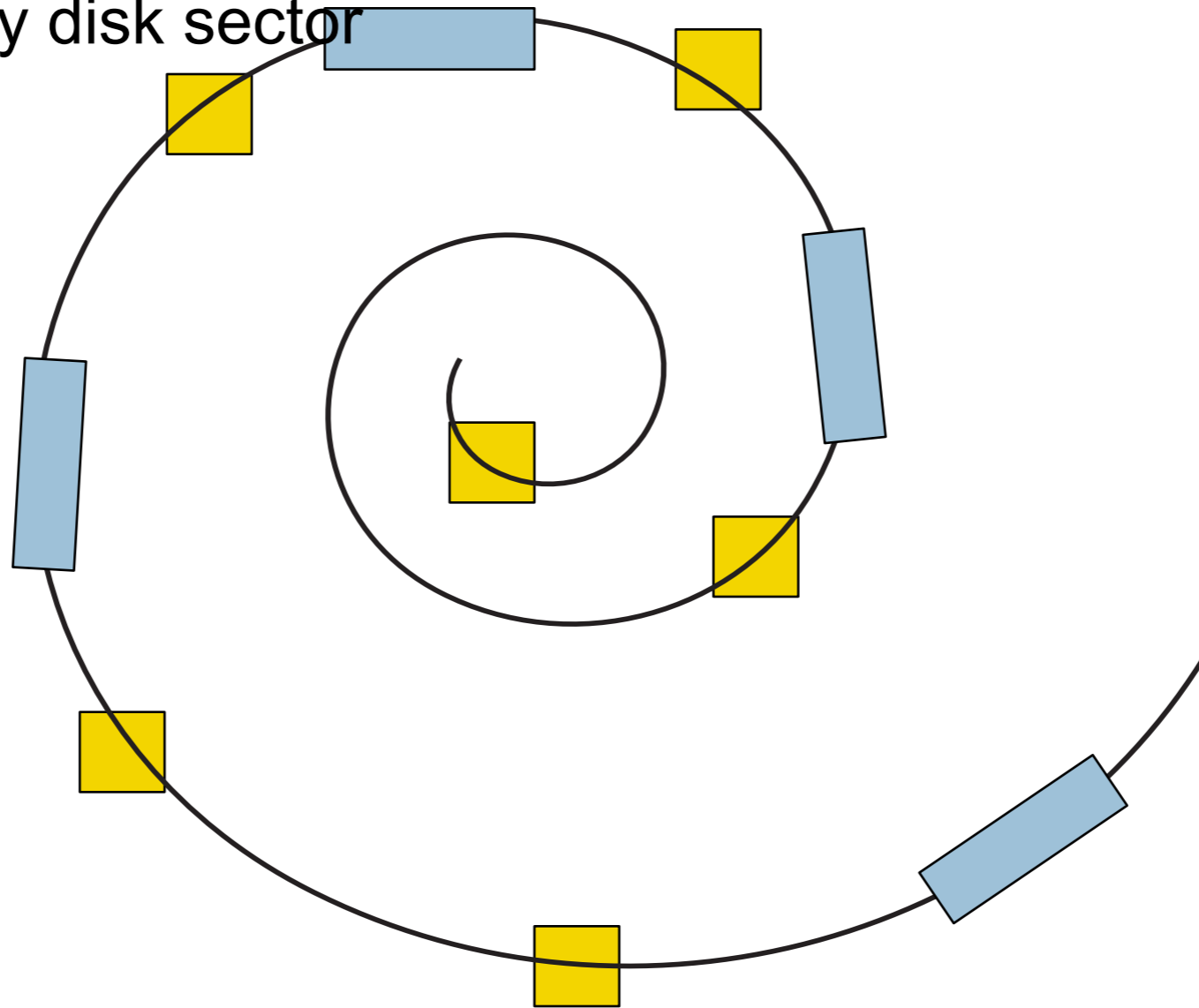
- 1 - Find all of the directories
- 2 - Locate all the files





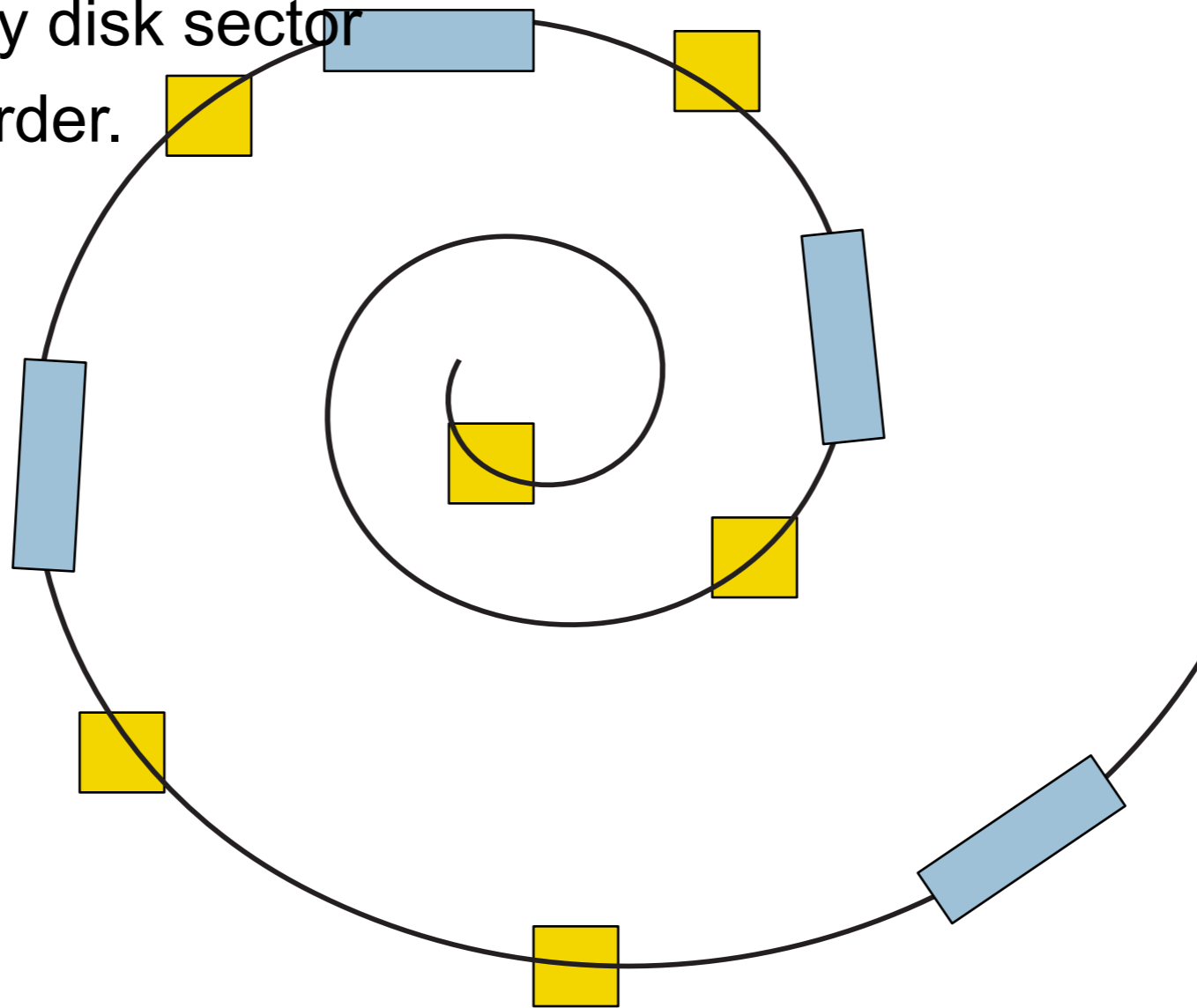
# There is room for improvement!

- 1 - Find all of the directories
- 2 - Locate all the files
- 3 - Sort the files by disk sector



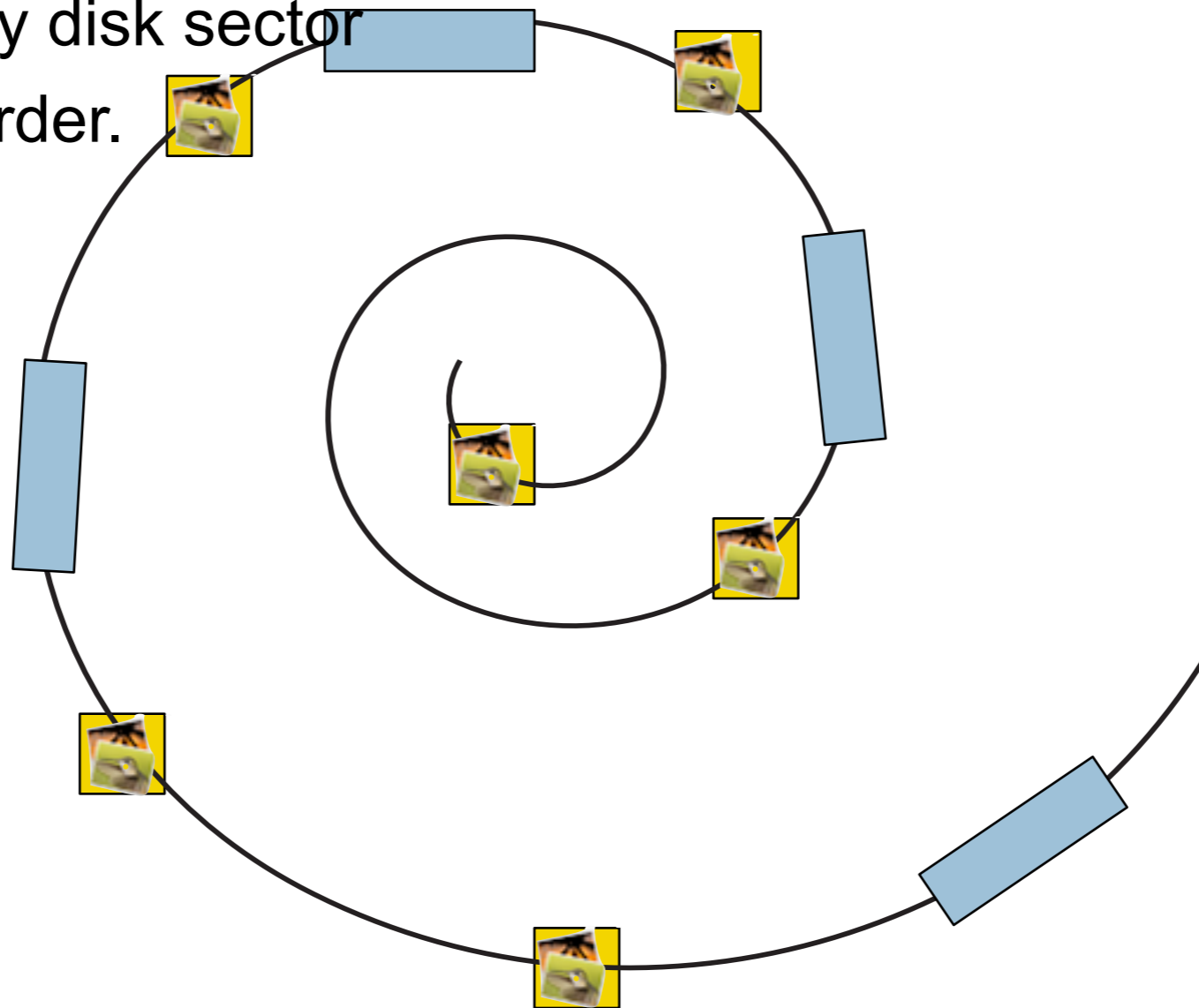
# There is room for improvement!

- 1 - Find all of the directories
- 2 - Locate all the files
- 3 - Sort the files by disk sector
- 4 - Read in disk order.



# There is room for improvement!

- 1 - Find all of the directories
- 2 - Locate all the files
- 3 - Sort the files by disk sector
- 4 - Read in disk order.



# Question: how much time can we save in forensic analysis by processing files in *sector order*?

Currently, forensic programs process in directory order.

```
for (dirpath,dirnames,filenames) in os.walk("/mnt"):  
    for filename in filenames:  
        process(dirpath+"/"+filename)
```



Advantages of processing by sector order:

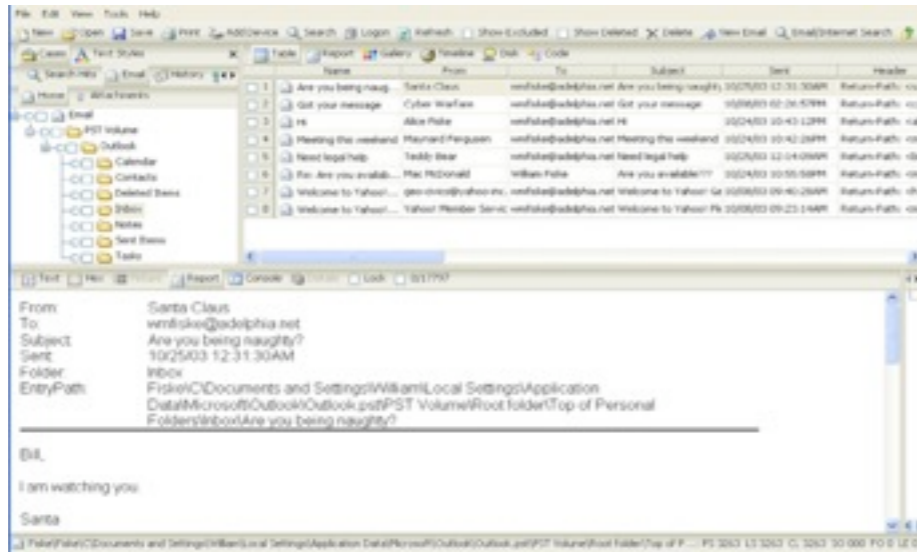
⇒ Minimizes head seeks.

Disadvantages:

⇒ Overhead to obtain file system metadata (but you only need to do it once).

⇒ File fragmentation means you can't do a perfect job:

# Unfortunately, today's forensic tools are designed for performing forensic investigations.



**Encase:**  
- GUI Closed Source



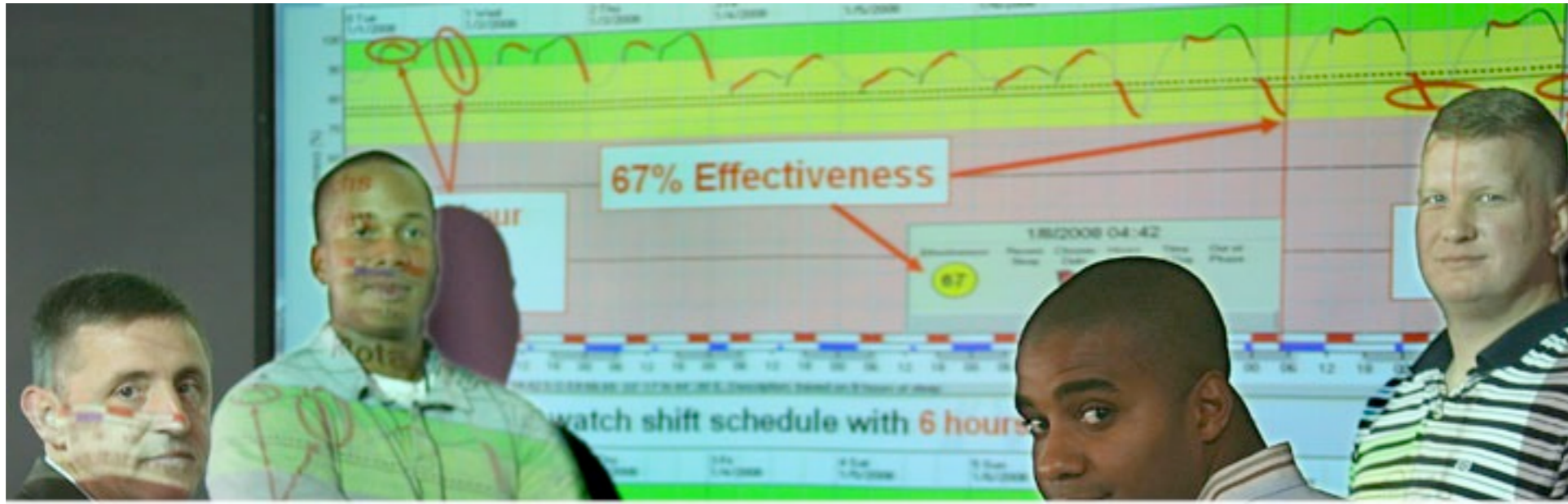
**SleuthKit:**  
- Command-line Open Source

These tools are great for:

- ⇒ File recovery
- ⇒ Search

***These tools were not created for research or automation.***

# Students (and researchers) need an easy-to-program environment for conducting forensic experiments.



It's *hard* to work with forensic data — All the details matter

- ⇒ Many different file systems.
- ⇒ Many different file types.

Good research requires working with large data sets.

- ⇒ Even small "pilot studies" should be tested on multiple data sources.
- ⇒ Otherwise, you aren't doing research on forensics — you are researching a particular object.

# But there is no good match between forensic tools and the needs of researchers.

Several of today's tools allow some degree of programmability:

- ⇒ EnCase – EScript
- ⇒ PyFlag – Flash Script & Python
- ⇒ Sleuth Kit – C/C++

But *writing programs* for these systems is hard:

- ⇒ Many of the forensic tools are not designed for easy automation.
- ⇒ Programming languages are *procedural* and *mechanism-oriented*
- ⇒ Data is separated from actions on the data.

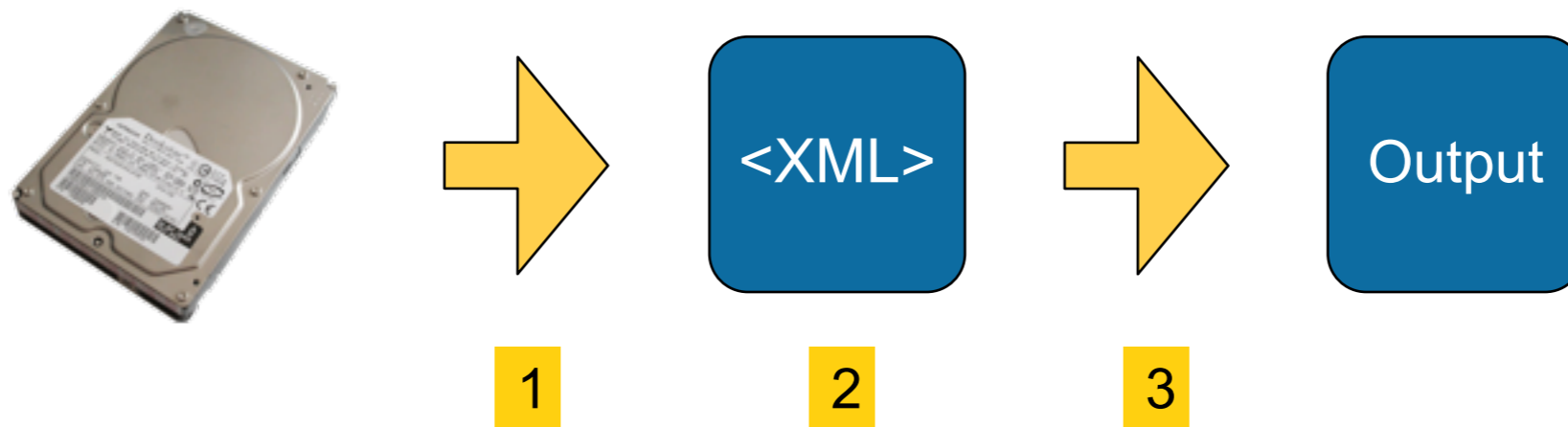
Faced with this, a standard approach is to leverage the database:

- ⇒ Extract everything into an SQL database.
- ⇒ Use multiple SELECT statements to generate reports.

# We have developed a new approach for automated forensic analysis and research

The approach breaks forensic processing into three key parts:

- 1.Extraction of forensic metadata.
- 2.Representation of the extracted metadata.
- 3.Processing.



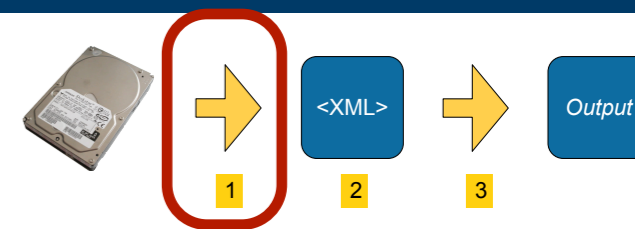
You can start using this framework today.

You can easily expand it.



# The framework is based on fiwalk, a tool that extracts metadata from disk images.

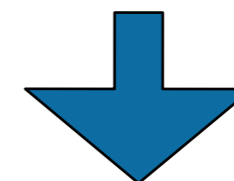
fiwalk is a C++ program built on top of SleuthKit



```
$ fiwalk [options] -X file.xml imagefile
```

## Features:

- ⇒ Finds all partitions & automatically processes each.
- ⇒ Handles file systems on raw device (partition-less).
- ⇒ Creates a *single output file* with forensic data data from all.

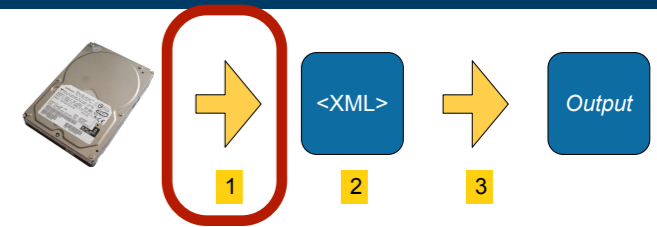


## Single program has multiple output formats:

- ⇒ XML (for automated processing)
- ⇒ ARFF (for data mining with Weka)
- ⇒ "walk" format (easy debugging)
- ⇒ SleuthKit Body File (for legacy timeline tools)
- ⇒ CSV (for spreadsheets)\*



# fiwalk provides limited control over extraction.



## Include/Exclude criteria:

- ⇒ Presence/Absence of file SHA1 in a Bloom Filter
- ⇒ File name matching.

```
fiwalk -n .jpeg /dev/sda # just extract the .jpeg files
```

## File System Metadata:

- ⇒ -g – Report position of all file fragments
- ⇒ -O – Do not report orphan or unallocated files

## Full Content Options:

- ⇒ -m – Report the MD5 of every file
- ⇒ -1 – Report the SHA1 of every file
- ⇒ -s *dir* – Save files to *dir*

# XML is ideally suited for representing forensic data.

Forensic data is tree-structured.

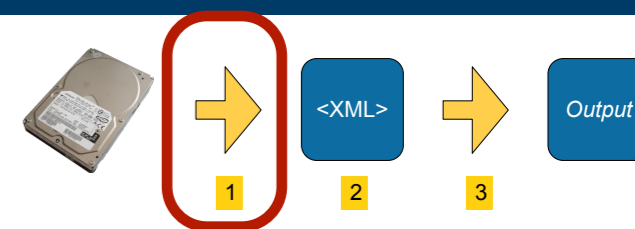
⇒ Case > Devices > Partitions > Directories > Files

⇒ Files

- *file system metadata*
- *file meta data*
- *file content*

⇒ Container Files (ZIP, tar, CAB)

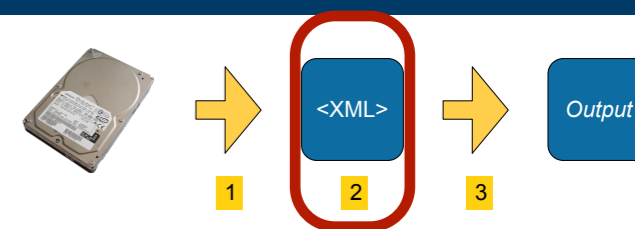
- *We can exactly represent the container structure*
- *PyFlag does this with “virtual files”*
- *No easy way to do this with the current TSK/EnCase/FTK structure*
- *(Note: Container files not currently implemented.)*



# fiwalk produces three kinds of XML tags.

## Per-Image tags

```
<fiwalk> – outer tag  
<fiwalk_version>0.4</fiwalk_version>  
<Start_time>Mon Oct 13 19:12:09 2008</Start_time>  
<Imagefile>dosfs.dmg</Imagefile>  
<volume startsector="512">
```



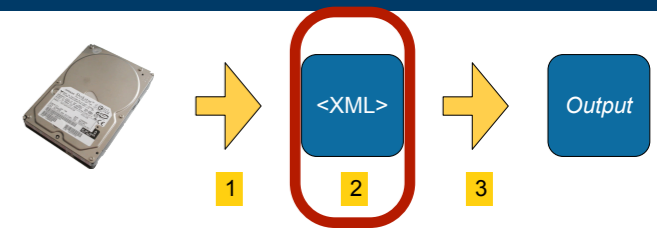
## Per <volume> tags:

```
<Partition_Offset>512</Partition_Offset>  
<block_size>512</block_size>  
<ftype>4</ftype>  
<ftype_str>fat16</ftype_str>  
<block_count>81982</block_count>
```

## Per <fileobject> tags:

```
<filesize>4096</filesize>  
<partition>1</partition>  
<filename>linedash.gif</filename>  
<libmagic>GIF image data, version 89a, 410 x 143</libmagic>
```

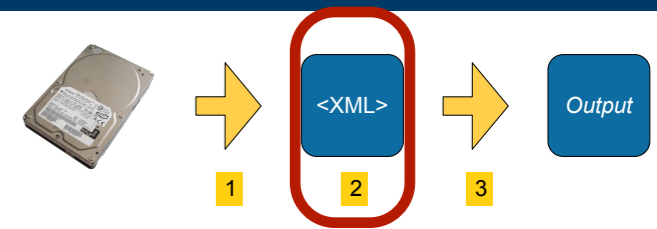
# fiwalk XML example



```
<fileobject>
<filename>WINDOWS/system32/config/systemprofile/「开始」菜单/程序/附件/_rf55.tmp</
filename>
<filesize>1391</filesize>
<unalloc>1</unalloc>
<used>1</used>
<mtime>1150873922</mtime>
<ctime>1160927826</ctime>
<atime>1160884800</atime>
<fragments>0</fragments>
<md5>d41d8cd98f00b204e9800998ecf8427e</md5>
<sha1>da39a3ee5e6b4b0d3255bfef95601890afd80709</sha1>
<partition>1</partition>
<byte_runs type='resident'>
  <run file_offset='0' len='65536'
    fs_offset='871588864' img_offset='871621120' />
  <run file_offset='65536' len='25920'
    fs_offset='871748608' img_offset='871780864' />
</byte_runs>
</fileobject>
```

# <byte\_runs> specifies data's physical location.

One or more <run> elements may be present:



```
<byte_runs type='resident'>
```

```
  <run file_offset='0' len='65536'  
      fs_offset='871588864' img_offset='871621120' />
```

```
  <run file_offset='65536' len='25920'  
      fs_offset='871748608' img_offset='871780864' />
```

```
</byte_runs>
```

This file has two fragments:

⇒ 64K starting at sector 1702385 ( $871621120 \div 512$ )

⇒ 25,920 bytes starting at sector 1702697 ( $871780864 \div 512$ )

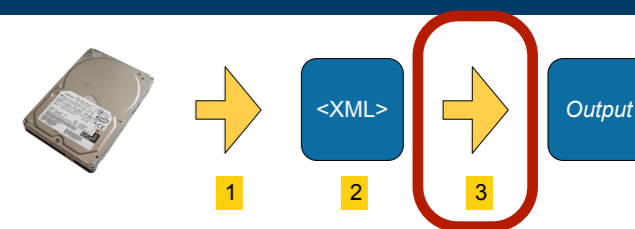
Additional XML attributes may specify compression or encryption.

⇒ Note: Currently <byte\_runs> not provided for compressed or MFT-resident files.

# fiwalk.py: a Python module for automated forensics.

## Key Features:

- ⇒ Automatically runs fiwalk with correct options if given a disk image
- ⇒ Reads XML file if present (faster than regenerating)
- ⇒ Creates **fileobject** objects.



## Multiple interfaces:

- ⇒ SAX callback interface

```
fiwalk_using_sax(imagefile, xmlfile, flags, callback)
```

– *Very fast and minimal memory footprint*

- ⇒ SAX procedural interface

```
objs = fileobjects_using_sax(imagefile, xmlfile, flags)
```

– *Reasonably fast; returns a list of all file objects with XML in dictionary*

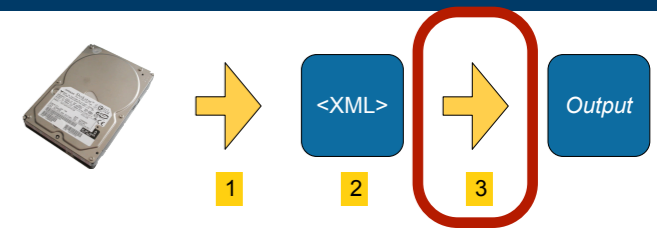
- ⇒ DOM procedural interface

```
(doc, objs) = fileobjects_using_dom(imagefile, xmlfile, flags)
```

– *Allows modification of XML that's returned.*

# The SAX and DOM interfaces both return fileobjects!

The Python `fileobject` class is an easy-to-use abstract class for working with file system data.



Objects belong to one of two subclasses:

<code>fileobject_sax(fileobject)</code>	– <i>for the SAX interface</i>
<code>fileobject_dom(fileobject)</code>	– <i>for the DOM interface</i>

Both classes support the same interface:

- `fi.partition()`
- `fi.filename()`, `fi.ext()`
- `fi.filesize()`
- `fi.ctime()`, `fi.atime()`, `fi.cmtime()`, `fi.mtime()`
- `fi.sha1()`, `fi.md5()`
- `fi.byteruns()`, `fi.fragments()`
- `fi.content()*`



# Using the framework, we performed the experiment.

Here's most of the program:

```
t0 = time.time()
fis = fiwalk.fileobjects_using_sax(imagefile)
t1 = time.time()
print "Time to get metadata: %g seconds" % (t1-t0)

print "Native order: "
calc_jumps(fis, "Native Order")
fis.sort(key=lambda(a):a.byteruns()[0].img_offset)
calc_jumps(fis, "Sorted Order")
```

With this framework, it took less than 10 minutes to write the program that conducted the experiment.

Answer: Processing files in sector order can improve performance *dramatically*.

	Unsorted	Sorted
Files processed:	23,222	23,222
backwards seeks	12,700	4,817
Time to extract metadata:	19 seconds	19 seconds
Time to read files:	441 seconds	38 seconds
Total time:	460 seconds	57 seconds

disk image: nps-2009-domexusers1



# Instant Drive Forensics with Statistical Sampling



# Research Question: Is it possible to analyze a hard drive in a minute?



What if US agents encounter a hard drive at a border crossing?



Or if a room filled with computers turns up on a search?



# If it takes 3.5 hours to read a 1TB hard drive, what can you learn in 1 minute?

		
Minutes	208	1
Max Data	1.5 TB	7.2 GB
Max Seeks	15 million	72,000

7.2 GB is a lot of data!

⇒ ≈ 0.48% of the disk

⇒ But it can be a statistically significant sample.

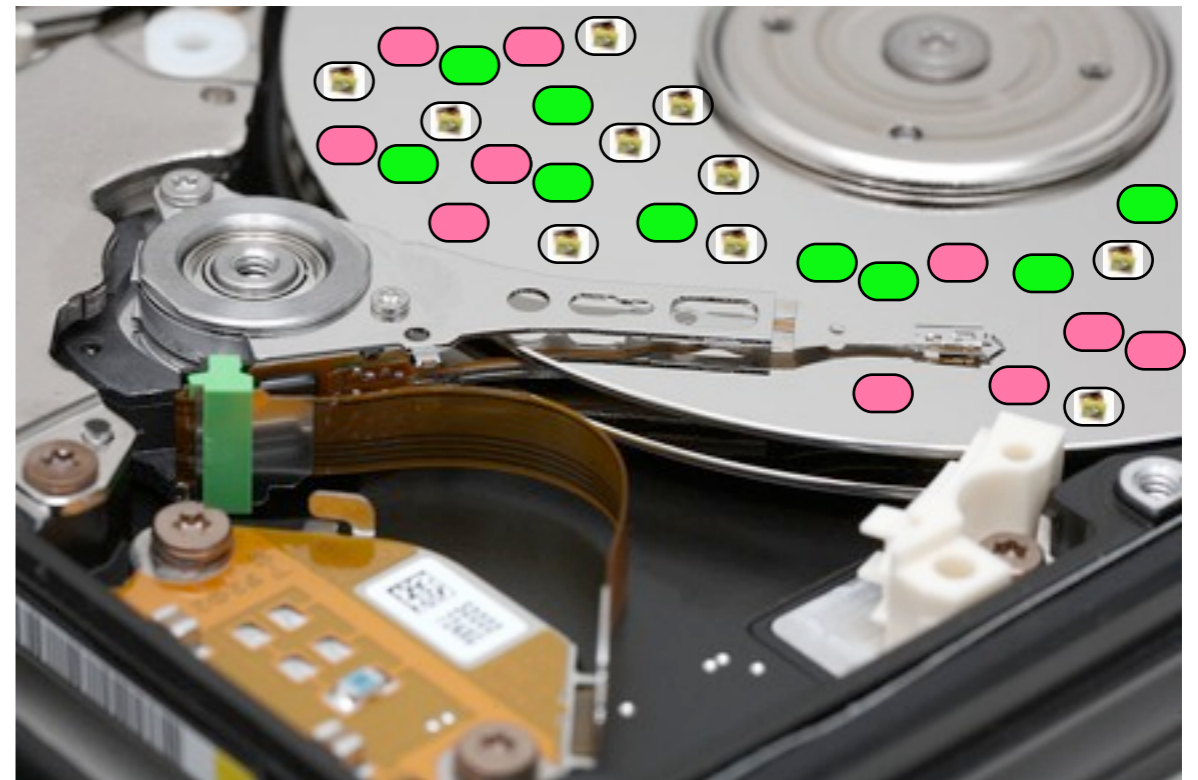
We can predict the statistics of a *population* by sampling a *randomly chosen sample*.

US elections can be accurately predicted by sampling a few thousand households:



The challenge is making sure that sample matches *likely voters*.

Hard drive contents can be predicted by sampling a few thousand sectors:



The challenge is *identifying the sectors* that are sampled.

The accuracy of a random sample can be computed from  $p$  (accuracy) and  $n$  (sample size), if  $n < 5\%$

$$\text{Standard error} = \sqrt{\frac{p(1-p)}{n}}$$

### Example:

- ⇒ Sample 10,000 sectors and find 30% blank, standard error  $\approx 4.5\%$
- ⇒ Sample 100,000 sectors and find 15% are video, standard error  $\approx 1.1\%$
- ⇒ Sample 1,000,000 sectors and find 5% encrypted, standard error  $\approx .02\%$

### Caveats:

- ⇒ The statistics are for the disk as a whole, not for the files.
- ⇒ We *must* be able to identify the content of sectors.

# Sectors on hard drives can be divided into three categories:

Resident Data

Deleted Data

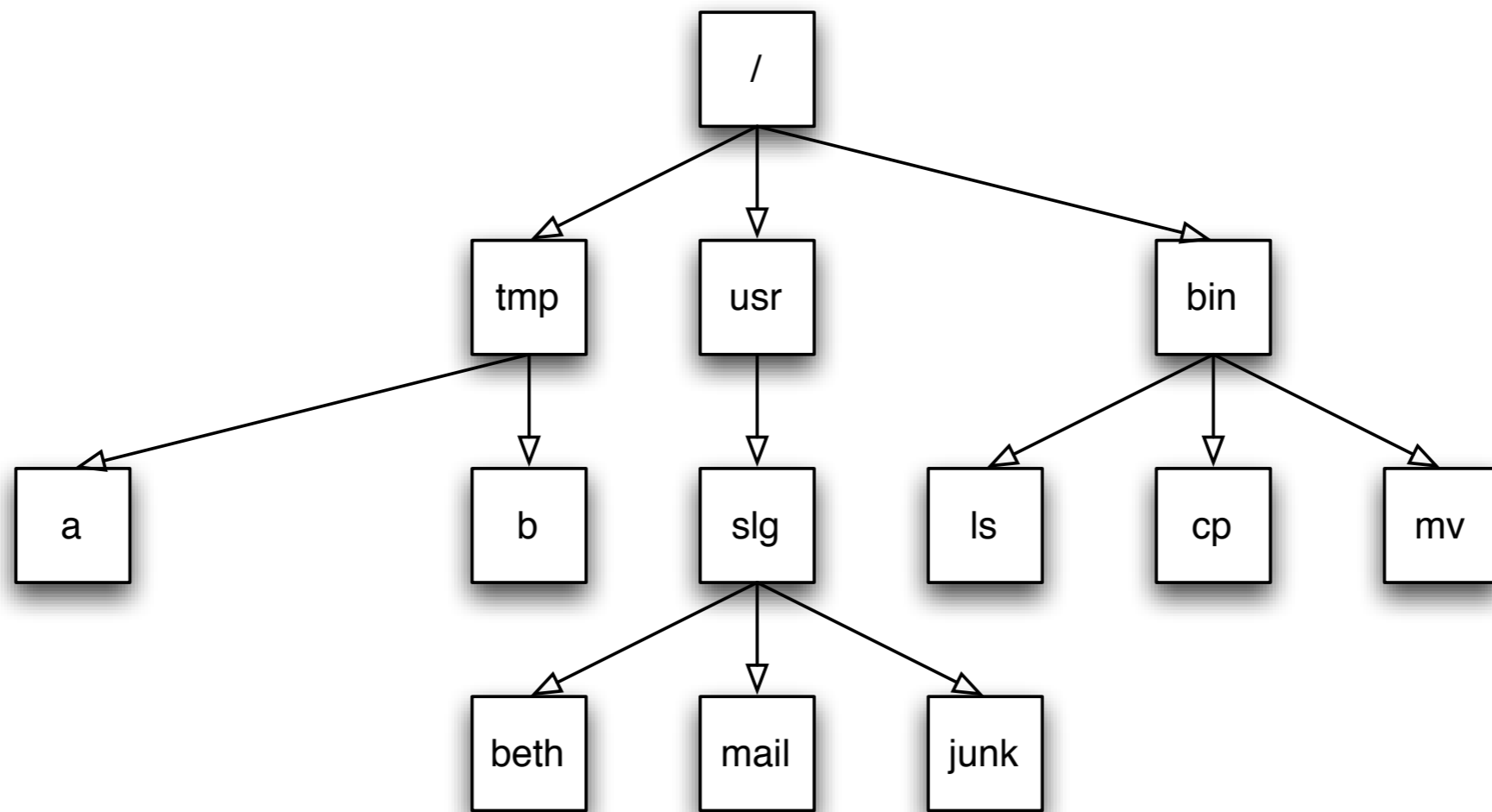
Uninteresting Data

user files  
email messages  
[temporary files]

blank sectors [OS files]



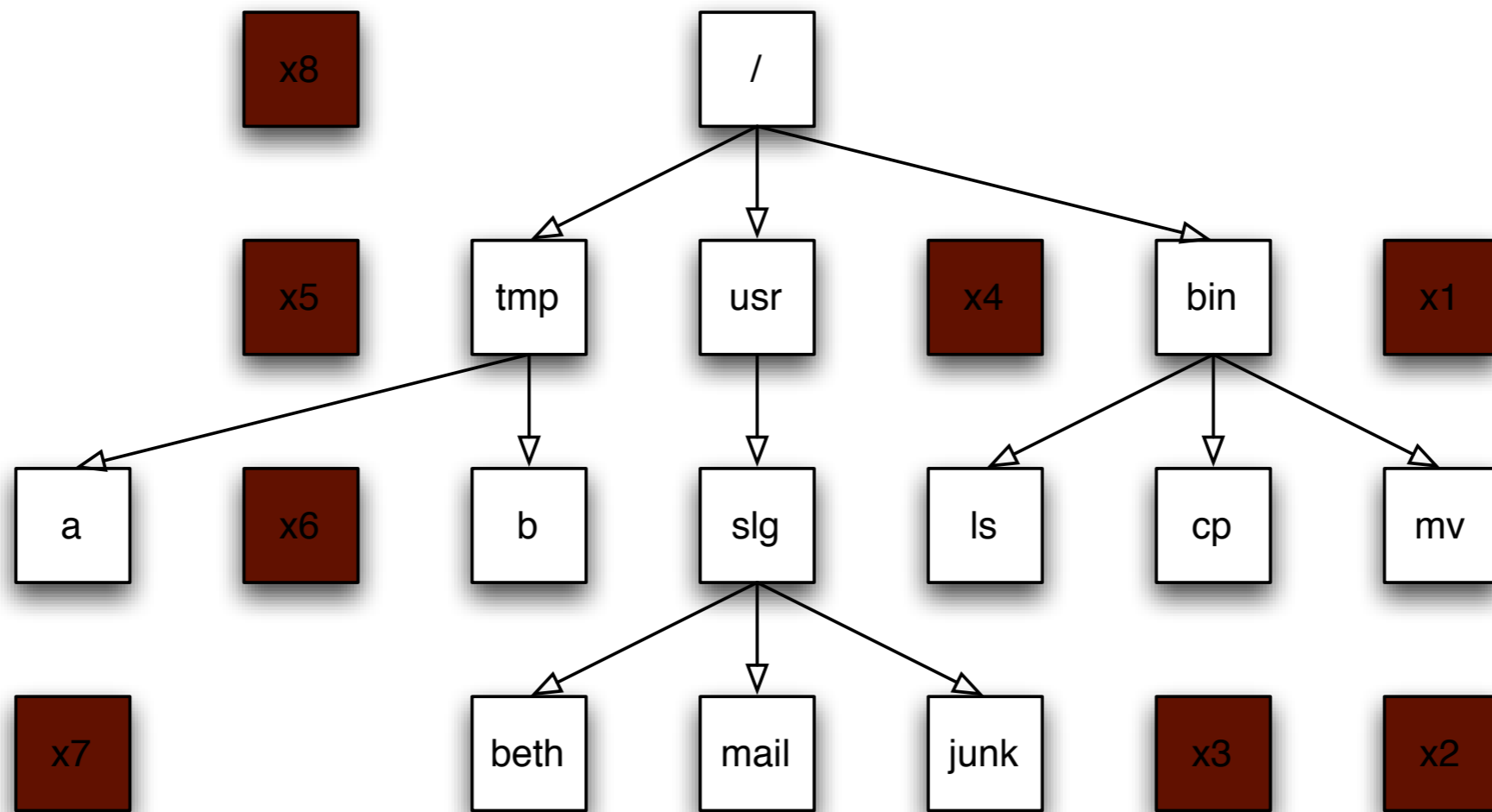
# Data on a hard drive is arranged in sectors



Resident Data

= data visible to the user

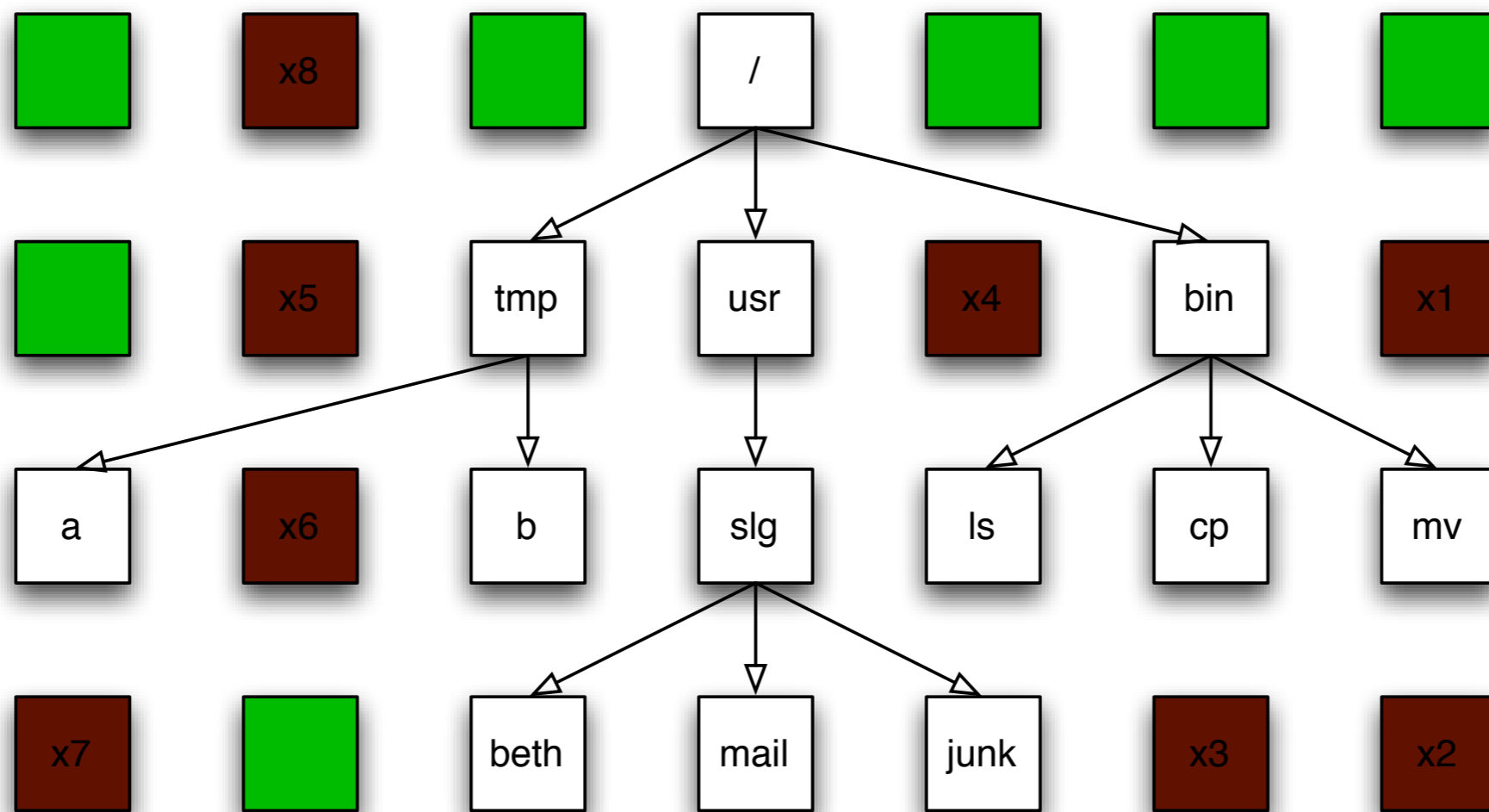
# Data on a hard drive is arranged in sectors



Deleted Data

= files that were deleted.

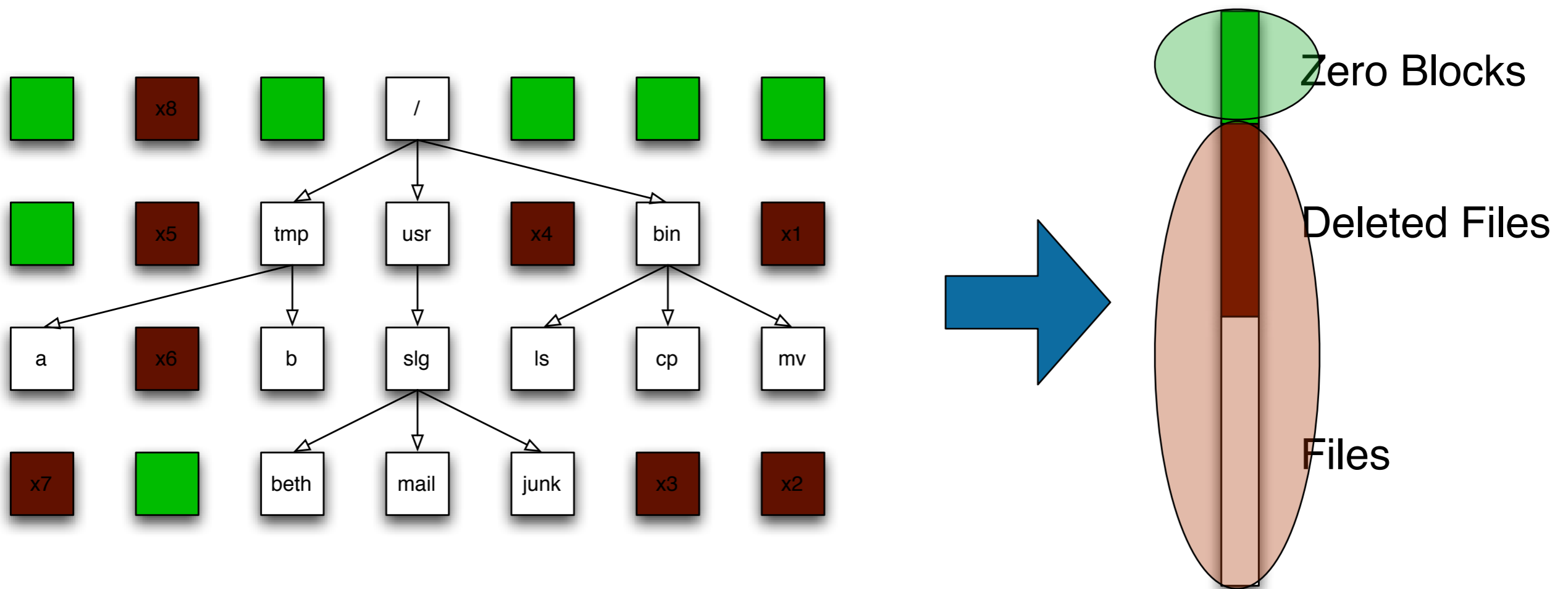
# Data on a hard drive is arranged in sectors



Uninteresting Data

= never written (or wiped clean)

Statistical can distinguish between "zero" and data.  
It's can't distinguish between resident and deleted.





# Many files are "container" files. Classifying sectors from these files will classify

The PDF file format consists of:

- ⇒ PDF header
- ⇒ PDF xref table (a directory of objects in the PDF file)
- ⇒ PDF objects (T/F; Numbers; Strings; Names; Arrays; Dicts; Streams; Null)

PDF header:

```
Terminal — emacs-i386 — 82x20
PDF-1.3
%PDF-1.3
4 0 obj
<< /Length 5 0 R /Filter /FlateDecode >>
stream
x^A\315W\313n\2030^P\274\363AUsl^N^P/\306\330\ \233\266Ro\215\204\324s\205\210\222\
\2524MH\245~^W\232WK\224^FKD+#\220\260^Y\217\355\231\335e\205)W \207\204`l
\2535\326W\236\361\216\361\244&^T5\250nu^A\262\221\341q|;^29k^A\^F\243-\264\311^P\
+^J\354^P\236\312uQ~l>_ \336\260^A\360-d\1\277\342\306c\260\375\244\2500~\254^HvK^F\3\
52p\331#\1\352\270\341\242\300<\230o\213^R^Rb\333\320\321^L\3078\2679\250AW^H\371^A\
Y\222K\271/\244,A\316\223<p\244^E8>\303H>_ \324\340\253^V%6\345\327^F#\344\257\270\
\317^?^H\^D\256\ \344\254\315Z\374` \207\317\304N\342v6<\377b^T\364e\263]\252\213\3\
67K\335RiVv~2\217^Q\336\374\254\222\315\3178\331\374\222T6?)\260\232H\375\305\302\
\375\301a\3530(\271\314\235\276\206t^]=\211\211^UV\266\224R\331J2\236B\352y\374\2\
76\272\323\262\343|\334\261\305e6\274\322\356\221lk\210v\206o\204\275\316\321\312\
\3160\247\263\273g^L\350\226Z=\267\370^?Kz2\223]\244\312.\361\217\376\265^D^V\200\
\244^F^K\354\236Z#\335I\324Rl@ \211\354@nDgAJ^G\223\332/\201\370\352\316^NV\201^J^A\
XM\277^A&\371<\314
-:---F1 file1.pdf Top L?? (Text)---11:32AM 0.43 Mail-----
```

PDF xref:

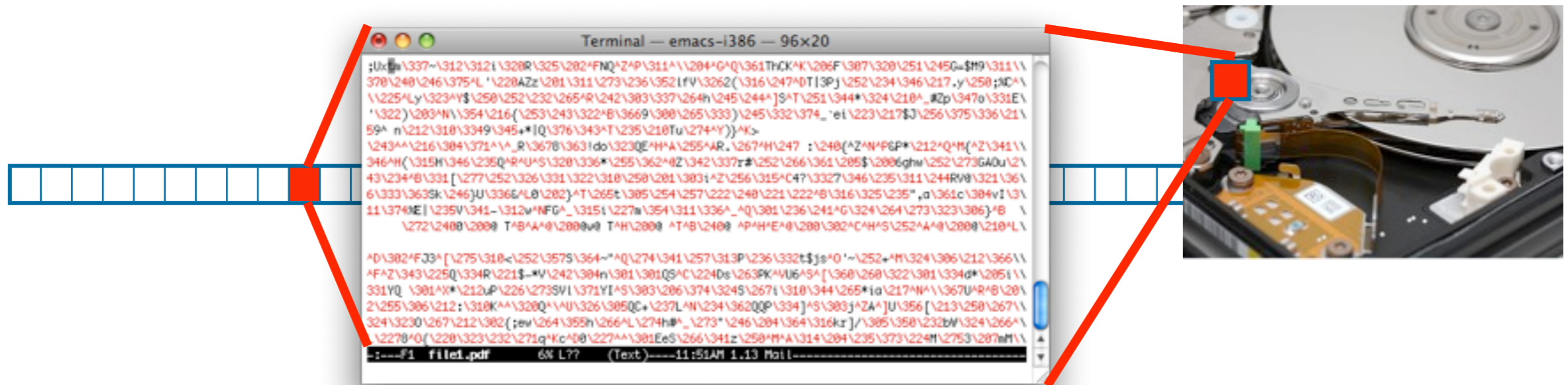
```
Terminal — emacs-i386 — 82x20
xref
0 21
0000000000 65535 f
0000073226 00000 n
0000000496 00000 n
00000057942 00000 n
0000000022 00000 n
0000000477 00000 n
0000000600 00000 n
0000057041 00000 n
0000000761 00000 n
0000066105 00000 n
0000067905 00000 n
0000073051 00000 n
0000066126 00000 n
0000067021 00000 n
0000067077 00000 n
0000067885 00000 n
-:---F1 file1.pdf 99% L?? (Text)---11:33AM 0.27 Mail-----
```







# Some sectors are characteristically PDF data, others are just JPEGs or compressed text.



# Most files on the hard drive are not fragmented. JPEGs in PDFs can be identified by scanning backwards.

In previous research, we found that only 15% of forensically interesting files are fragmented [Garfinkel 2007].

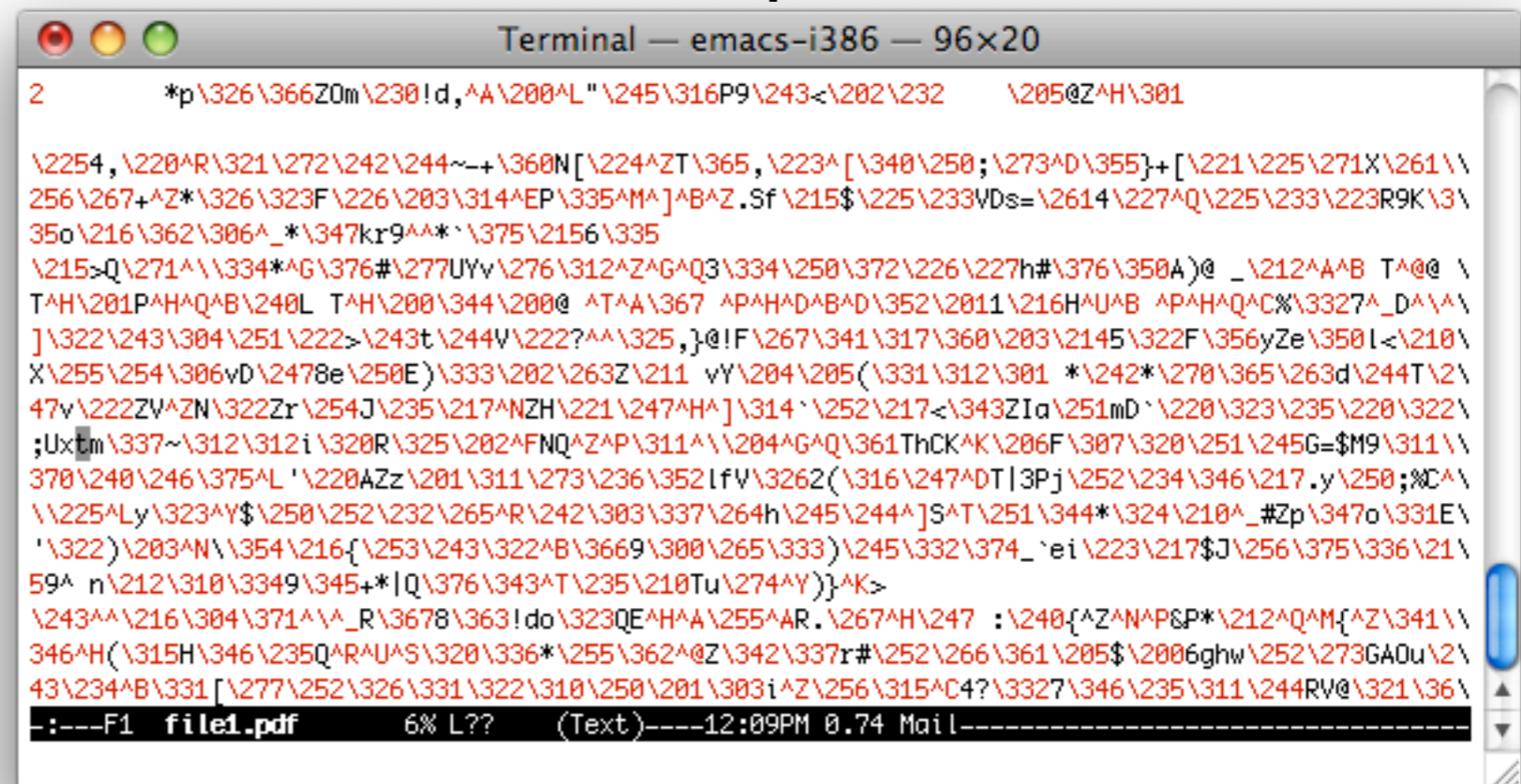
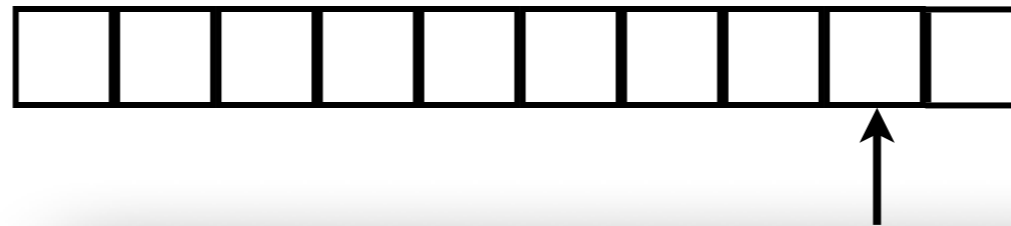
Therefore, we can use a sector's *context* to assist in identification:



# Most files on the hard drive are not fragmented. JPEGs in PDFs can be identified by scanning backwards.

In previous research, we found that only 15% of forensically interesting files are fragmented [Garfinkel 2007].

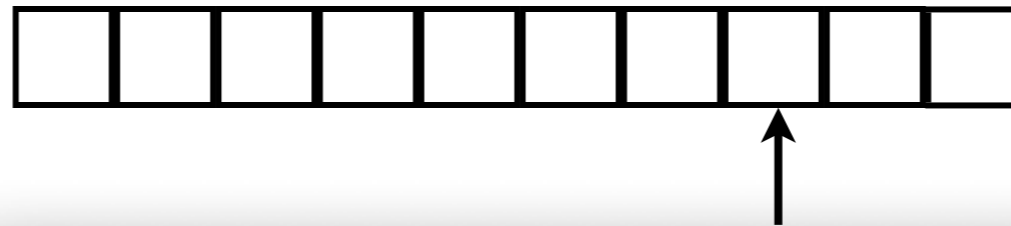
Therefore, we can use a sector's *context* to assist in identification:



# Most files on the hard drive are not fragmented. JPEGs in PDFs can be identified by scanning backwards.

In previous research, we found that only 15% of forensically interesting files are fragmented [Garfinkel 2007].

Therefore, we can use a sector's *context* to assist in identification:

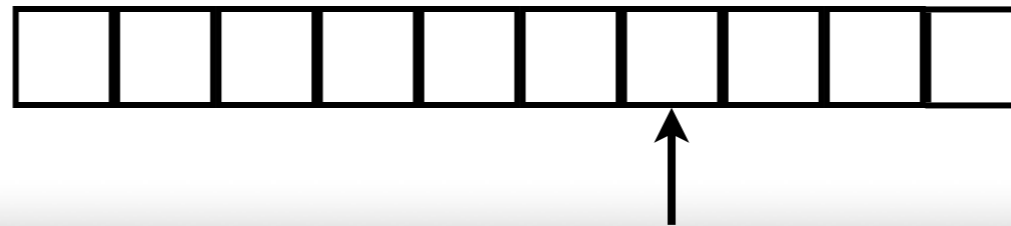


```
4
^K=-^[@\2166\347\334\265\244h\206\20000^B^H\346\204H^HTb\326QI^S\213\343^_ADD\226\352\343\235^R\
\267^HF\303Hsr\303\224S\242{\204\255 '\220(=^B\325\346\245a\364]1\256Y4^G%\266U\353"t\214:y\251f\
\311\375]5S\t\356\261\332\334\252\321\323\325\227c^XS\266\233\215\253}$_\215\363HV\361\307L\332\
\324\306^F^V\3311\312Ud\337\^[H\356\370\362\255\343^U\304\357\227\366d\340\347^[\2544\371\216\
\344^?\247Tg\373C\372\242\272\237e\221j\342X\311^\\202\250\372\262\211\270\246\214~\350A>^P^AA\
@\212^Q^B^A^B\240^P^_ \301^@\200\302^C^H^D^B^A^@\200P^H^D^B^A^C\364@\212\201@\210^W\252^D\302^A^A\
325^@\201^P^X\356\2011\262^Dw\331*\216;\210\216yz\254\325\214;\300.\262<\217\302\213\360\312\34\
0\x\217\367\255#\320^Y\263\226^ZYj\241&^Yj\224\221A\303^0\335a\245\256l^Ek\341\220\302\220\251^
G5E^Z\366\341\300\251b\303\3429\210(_i\240*\302\246'ltU
\336J\241\p A\301\337j@\2715\247\272\261^Z^T\222^G0^L\247\245hD0\321\216j\242\245}7\214|\30\
3!e\250\265g\245\202^]\334^A\357\262\324Jm\322\232^Y\336t^F\267^\\222\222\251\322\323\3702^d\3\
74^VU\240\346\345\232\212\261\225Y^?\253vy"\304^Vb[V\3409eA\324^H"r\212^A\330\240\202C\346Qa\2\
62^L\261*\212C\271 \2135<\201ZDd\212B|\250D \341\3531W[\366B\333^Dp\310ET\231\232\203\2069\242\
\271Wf\232\352^G \356h\225^A\356\343M_ ^D\315\344N
\322;\373^L\3765+^Ns\262#\227\270N\306^W^B\341\266\330\332Ik\271x1\342(\213\236z\341Y^Szm\36\
5\322 *p\326\366Z0m\230!d,^A\200^L "\245\316P9\243<\202\232 \205@Z^H\301
```

# Most files on the hard drive are not fragmented. JPEGs in PDFs can be identified by scanning backwards.

In previous research, we found that only 15% of forensically interesting files are fragmented [Garfinkel 2007].

Therefore, we can use a sector's *context* to assist in identification:

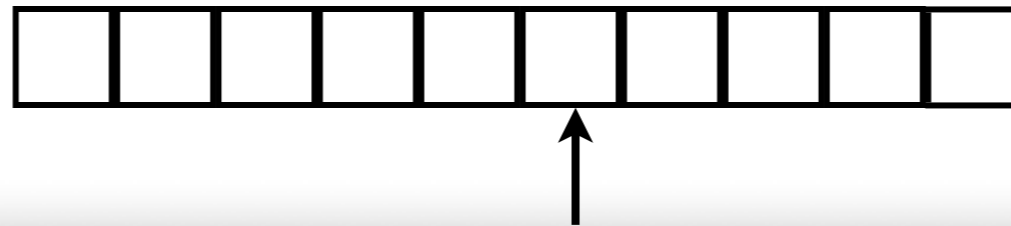


```
4H\361^F\312\310m\322\322Y\241\210^M\202\251^Z0S\2627\267KG0\250\364 [Q^_Ef{. \230\271U\374^B^WFt \
\212W\206^E-I^T\344\254^M<\326{\232\3226\327o\315N\345\322\3245! \373ejVlY^ [\205\2445\341\270R\2\
65^X\267\310\201\201\307^] ^W<\243q\300q; \364Ze \364+\233o\225orx\227:\247~\371U\226\227\263\370\
|n&\247^G\246\352\217\255\255^Q\370v\370\206: \272\212T@\200^@D^@@\250^D^B^A^@ \200@q^@ \200\350\ \
200\302\200@*^L (^D^F^P"^^@ \201: \240^P^E^@ \2010\250^P
^P\212^@ \362(9N#\223\313\204\253^X5\357\360mR8s\322\241\247?\302, \361\253^ \363\335i^^ \215^F\3\
00^E\206\326\333\350\252#\234yJ\225c^R\256\337^E5\376\261\200\222\260\322^G\360\305^Y\337\303o \ \
311kl \257\320Z\340\245\306\226\241\246\243F9*3\353\277\254
d\261, #Lj-00\317+Q\232\231\344\222\255CZ^Q0^ ["9~" \253w\210#a\346p\252#\240\2042=G\355wH\265~^R \ \
H\316^Z^UDs\325\262\225\371q^C\266Tj0*U7s, \330\246n\2629\351
B\315- \333na\323\265\225L\347\313R^^\327j\345\205\322^A^Xh> \212\355, U\231\243^ [\204E) \230\350\2\
16\270\316=^TV\265\242\267\306n\227^] \325^ZNQLP\246\310\300B
\345\270; \250\322^Y\243\322\340\340\247\241n\235\372\230\267^Y\242A \202\242\302^C\204^M\220d(^H\
^] \207nvV^Ug\230Ze^D\214^A\230o \245VM\322^Q^R^S\266\341A\227D\377^@^V '\304\377^@\264\335\226\2\
21- \226\264\323\326\272^W\236\252\304wt\257\327^X(8\372\247y9\364\ \332r\267\233\ \367" [\^C^K\217\ \
242\213^X^U \^YUL\337^Rw^F\366f2U\325\251\266\347^L\360\240\230\265\325^Q\371{\^P\223^R\345\364\3\
64
^K=-^ [\2166\347\334\265\244h\206\20000^B^H\346\204H^HTb\326QI^S\213\343^_ADD\226\352\343\235^R\
-:---F1 file1.pdf 4% L?? (Text)----12:09PM 0.74 Mail-----
```

# Most files on the hard drive are not fragmented. JPEGs in PDFs can be identified by scanning backwards.

In previous research, we found that only 15% of forensically interesting files are fragmented [Garfinkel 2007].

Therefore, we can use a sector's *context* to assist in identification:



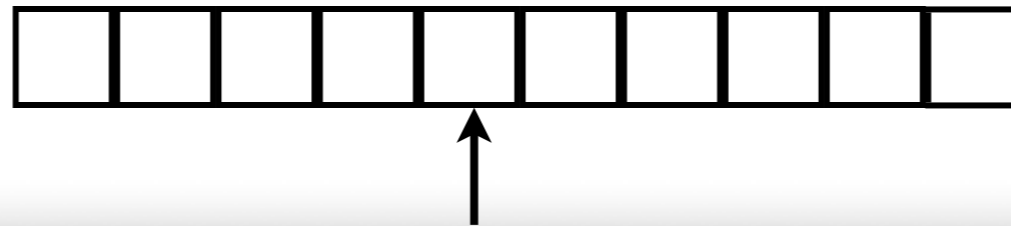
Terminal — emacs-i386 — 96x20

```
0\301\364\213\315,|\362\360\252>\271\341
a^E\242!\216\201^F\352^E\302^A^H^A^D^A^F^A^P^A^X@a^B\200\200\353\356@ ^P&^P)^H^L 0\200\302^A@a^@\200\35\
2\250D^A^F^A^P^A^H^L(^Q^a@\250^P^A^X@(^Q^P^a^@\240^U^A^H\203^A^V\373?\207^K\260\245X\346\242~"t\207\336\242\27\
1F\270\327]\216w^a@\253^Q\3506\330DP\264^A\205\232\261\242\321\262\252\202\267&^G\206\363\302#\2\
04\270\335\252h$~\254\350^|\202Jzs^0\277\317%Y-|\243Q\345\222\251\270\352xz\266y\244\303\301\35\
7\225(\356\341\335\200\234\362R*
\3710\334^D\331" \265$Y$\236ef5\245\326\214\234^NJ\262\270\314F\320\264\210\234\362\342\212^CQ^L \
\250\231\220F\342\203^B\256\350\371d-\213 '\334\211\263b\212Y7\221\337^EUe\2605\243\314UD\221\3\
07^[\216^G5^B\315L^[\314\343\342\212\251$R4\345\216D$ur\303\215Y!EjR\^[\6^C\216\350\253\204gpvD0\
\216\350\240^"\2Z\351p\233Uj\211K\201\302\315\253":y\216\254^i)b\331^@\214\205Q^_TXy \204^Q^S\ \
207(\251\343vZ\265^Y\247^)\302\250\245W^P^sH!e\246^D\241\324\263\357\366
\273e;\2441\351\221\212\216\212\321X&\215\2479*\245|\215\306Q^V0^X\304\310F\240\340p\271\3677\2\
47\234\361\357^T\307\340\331X<6\357\346Z\236R\370y\3650^S\323\303\221^P^Y\375\326\2523g\342\25\
2\207g\302\211\345M\303T\332j\313\325\311\332i\240 ^_\274F\311\277\241\322\332x:\345W\207\326\3\
16\354s-^C^A^E\313\217^B\304\326d\371\210\356\210\253E\303\202<\200^@#\226\313:jV\375\263\207\2\
34H\361^F\312\310m\322\322Y\241\210^M\202\251^Z0S\2627\267KG0\250\364[Q^_Ef{.\230\271U\374^B^WF\
t\212W\206^E-I^T\344\254^M<\326{\232\3226\327o\315N\345\322\3245!\373ejV\Y^[\205\2445\341\270R\ \
265^X\267\310\201\201\307^] ^W<\243q\300q;\364Ze \364+\233o\225orx\227:\247~\371U\226\227\263\37\
-:---F1 file1.pdf 4% L?? (Text)---12:09PM 0.74 Mail-----
```

# Most files on the hard drive are not fragmented. JPEGs in PDFs can be identified by scanning backwards.

In previous research, we found that only 15% of forensically interesting files are fragmented [Garfinkel 2007].

Therefore, we can use a sector's *context* to assist in identification:

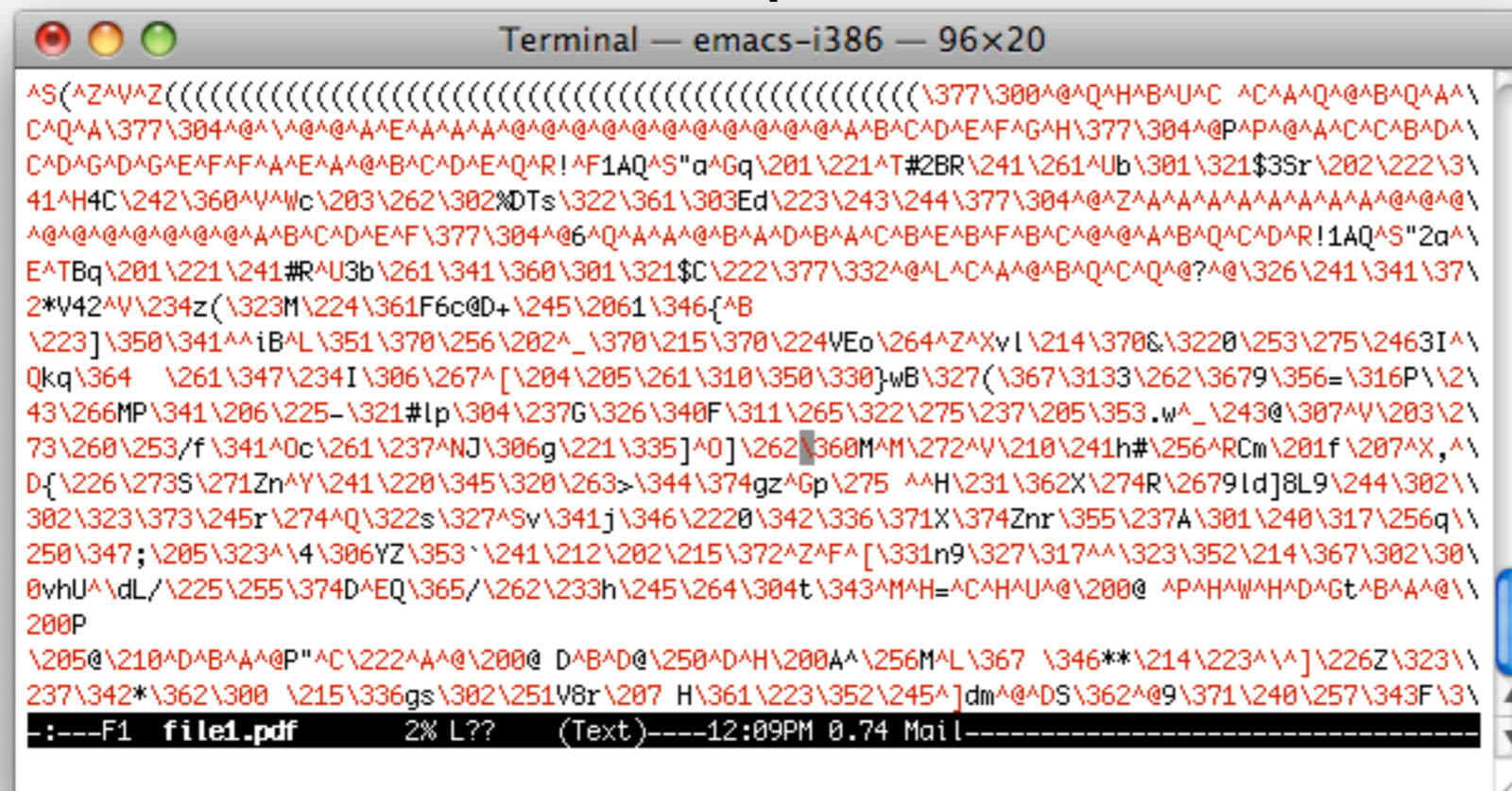
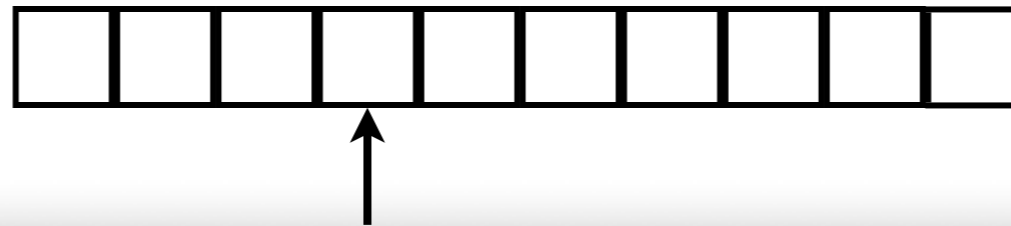


```
2*\362\300 \215\336gs\302\251\8r\207 H\361\223\352\245^]dm^@^DS\362^@9\371\240\257\343F\374\234\  
\251\245D*i\333\237^Q\2439M.\311\343\307!\323^Y  
kBX\232s\272D\251\334C^[\222\2643*gt\217\322\322\245\244\211 \203^C.\346\262\253qG\250\214rT\dm\  
crp\265\2445\362\376^P\202<\271\334\320/!\271\300A^L\2251\263\233\221^P\272\2761\260\346\201\24\  
2\340:^D  
\332\370\372\204^S2\246 '\375\344^R\202^]\366H(^P\201\215\302)\216 \364PF\340G$%#^c^N\346^RE\267G\  
\210^@\334+\244\336\312|\243t^L89!@\316E^B\214^T_F\221\202\240{^^z\255DI\315TW\250\211\257i^D^M\  
\324V^]\201\345\314\344\222\232> \3657c\376\212\246\232\226\332\322]\241\307q\352\2527\  
\241~\266\242$\352\203I\3342\323^@\335\332\210\350\262\257)\366\227^]\333\207\264\311J^L\221\27\  
4\340\202y+G\234>\341\304^U\204\371\304 \366\311*m\255^Z\333=\322\253y\352&>\200\341<\232Y\213\  
205G9\344\317\367\235\225^Mk\312A\303\364Q\2373\331\360  
\351^T\256\255\267\321\3239\2619\200\235\263\325Q\303\272\333%eItm:  
\315\246\235^E\263\206\271^W\2056\275\256\256\337gdA\241\261\222]^B\203z\232\3215C]^_ \206^Z\323\  
\260Z\220\333\323\270^G\202\251\255\261 \237^X2;rp\272c\213\235\257A\212^F\260^@\321\214.\223^\  
V-N^[\335kI\262\206^M\32164^D^M,@\307G\224\322\312\253SH\311Zr\321\362\356-L\234\265\316\213\3\  
50\362^R\321\345\312\347f\235%\333^N\362\355^V\3710\242\212\371k\214d2\337j]\234\341\330V" \227^\  
0\301\364\213\315,|\362\360\252>\271\341  
a^E\242!\216\201^F\352^E\302^C^H^D^F^P^X@a^B\200\200\353\356@ ^P&^P)^H^L 0\200\302^A@a^@\200\35\  
:---F1 file1.pdf 3% L?? (Text)---12:09PM 0.74 Mail-----
```

# Most files on the hard drive are not fragmented. JPEGs in PDFs can be identified by scanning backwards.

In previous research, we found that only 15% of forensically interesting files are fragmented [Garfinkel 2007].

Therefore, we can use a sector's *context* to assist in identification:









# Sectors can also be identified from statistical properties.

File type	Identified By
NULL	direct examination.
HTML	n-gram analysis
JPEG	High-entropy with markers
ZIP	High-entropy that's not encrypted
Encrypted	High-entropy that passes encryption tests

# Using sector identification, we can identify the *content* of a hard drive within 10 seconds (after it spins up).

Time to read 10,000 randomly chosen 64K runs: 4.4 seconds

## Identifiable:

- ⇒ Blank sectors
- ⇒ JPEGs
- ⇒ Encrypted data
- ⇒ HTML



## Standard error:

- ⇒ 0.1% for 10% determination
- ⇒ 5% for 50% determination (max error)

## Sample report:

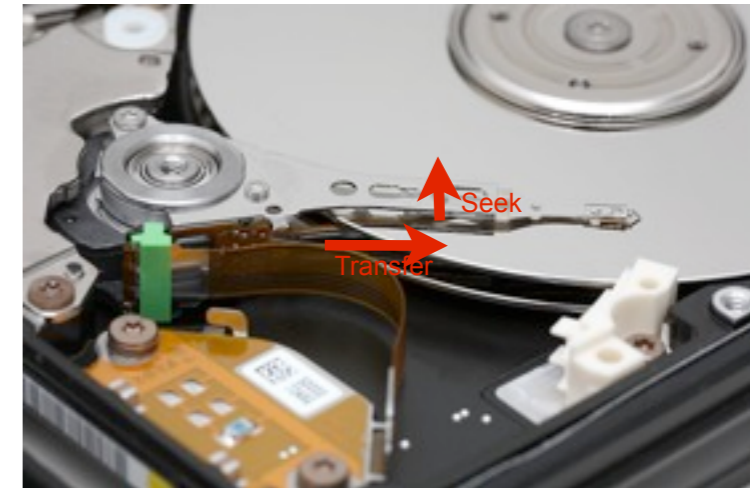
- ⇒ 10% encrypted; 30% JPEG; 50% MP3



# Conclusion: We can dramatically speed traditional forensics.

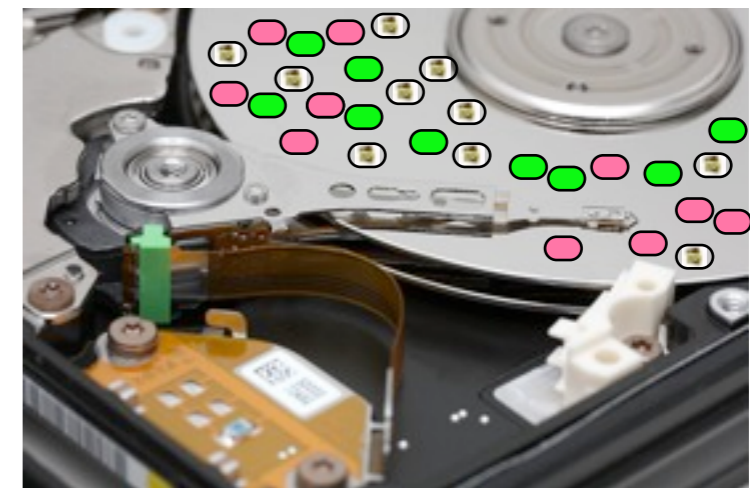
## Traditional forensics:

- ⇒ Speed processing by sorting files by sector order.
- ⇒ Irrelevant for flash storage.
- ⇒ Works today.



## Statistical Sampling:

- ⇒ Determine the content of a disk in 10 seconds.
- ⇒ Full content, not file content.
- ⇒ Research today.



For further information, see <http://simson.net/>

Question?