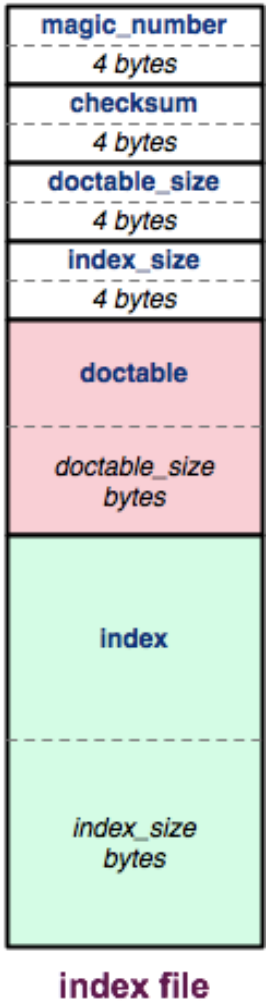
CSE 333 – SECTION 6

- HW3 Structure & Reading Hex-dumps

Overview

- HW3 due next Thursday @ 11 pm
- Questions?
- Reading HW3 index files

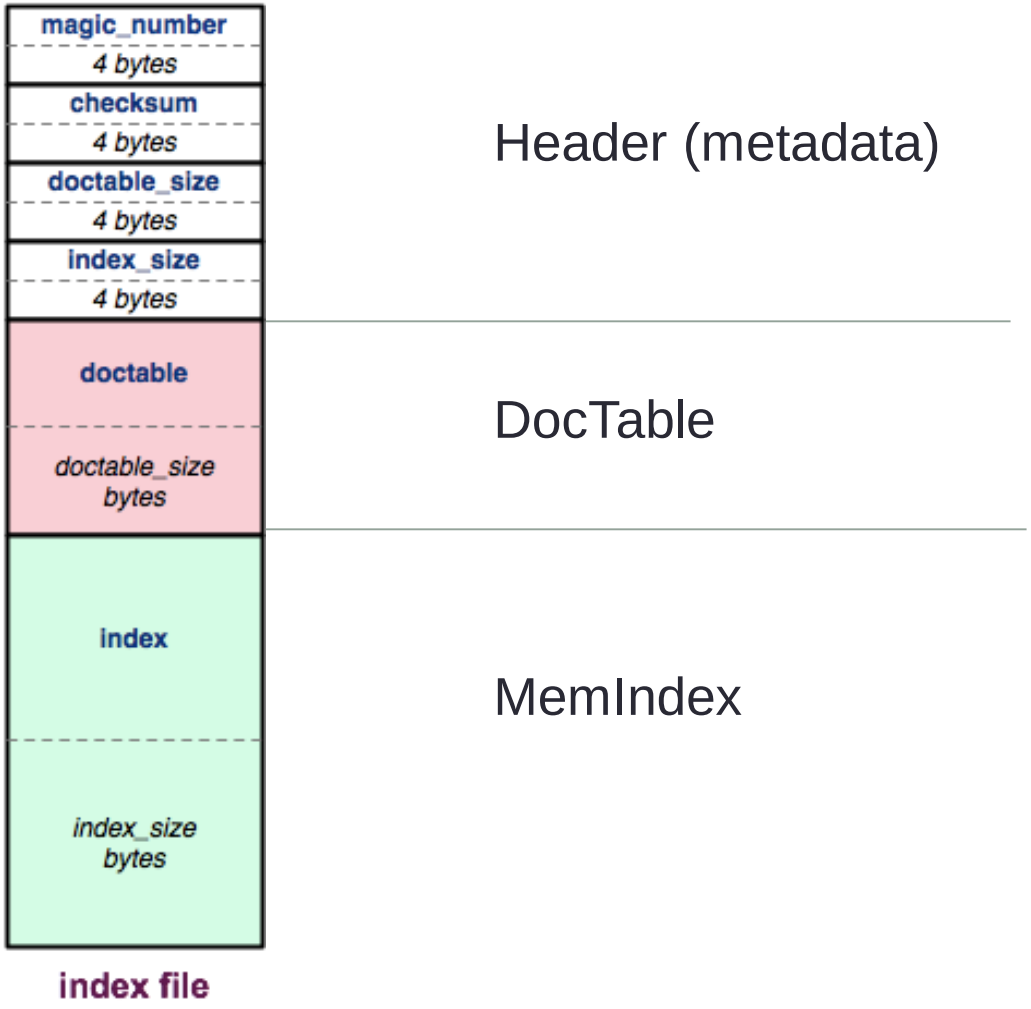
Index File



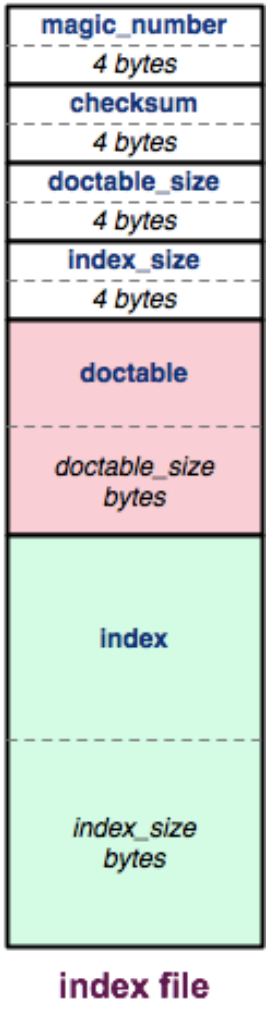
Crawling a file tree in HW2 takes a long time.

To save time, write the completed DocTable and MemIndex to a File!

Index File Components



Index File Header

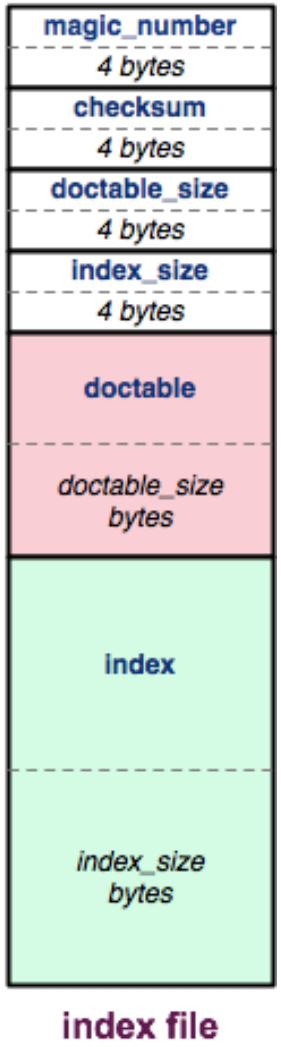


- magic_number: 0xCAFEF00D
- checksum: mathematical signature
- doctable_size: in bytes
- index_size: in bytes

Index File Header - HEX

1. Find a hex editor/viewer of your choice
   - xxd <indexfile>
   - hexdump –vC <indexfile>
   - Pipe the output into a file or less to view

```
0000000: cafe f00d 1c42 4620 0000 205b 0000 075d  .....BF .. [...]
0000010: 0000 0400 0000 0000 0000 2014 0000 0001  .......... .....
0000020: 0000 2014 0000 0001 0000 2031 0000 0001  .. ....... 1....
0000030: 0000 204e 0000 0000 0000 206b 0000 0000  .. N...... k....
0000040: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
0000050: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
```



The header:

Magic word Checksum Doctable size Index size

man xxd
man hexdump

Byte Ordering and Endianness

- Network (Disk) Byte Order (Big Endian)
  - The most significant byte is stored in the highest address
- Host byte order
  - Might be big or little endian, depending on the hardware
- To convert between orderings, we can use
  - uint32_t htonl (uint32_t hostlong); // host to network
  - uint32_t ntohl (uint32_t hostlong); // network to host

- Pro-tip:
  The structs in HW3 have toDiskFormat() and toHostFormat() functions that will convert endianness for you.

Hex View

- emacs “M-x hexl-mode”

```
File Edit Options Buffers Tools Hexl Help
87654321  0011 2233 4455 6677 8899 aabb ccdd eeff  0123456789abcdef
00000000: cafe f00d ce52 0578 0000 205e 0000 0944  .....R.x.. ^...D
00000010: 0000 0400 0000 0000 0000 2014 0000 0001  .......... .....
00000020: 0000 2014 0000 0001 0000 2032 0000 0001  .. ....... 2....
00000030: 0000 2050 0000 0000 0000 206e 0000 0000  .. P...... n....
00000040: 0000 206e 0000 0000 0000 206e 0000 0000  .. n...... n....
00000050: 0000 206e 0000 0000 0000 206e 0000 0000  .. n...... n....
```

- vim “:%!xxd”

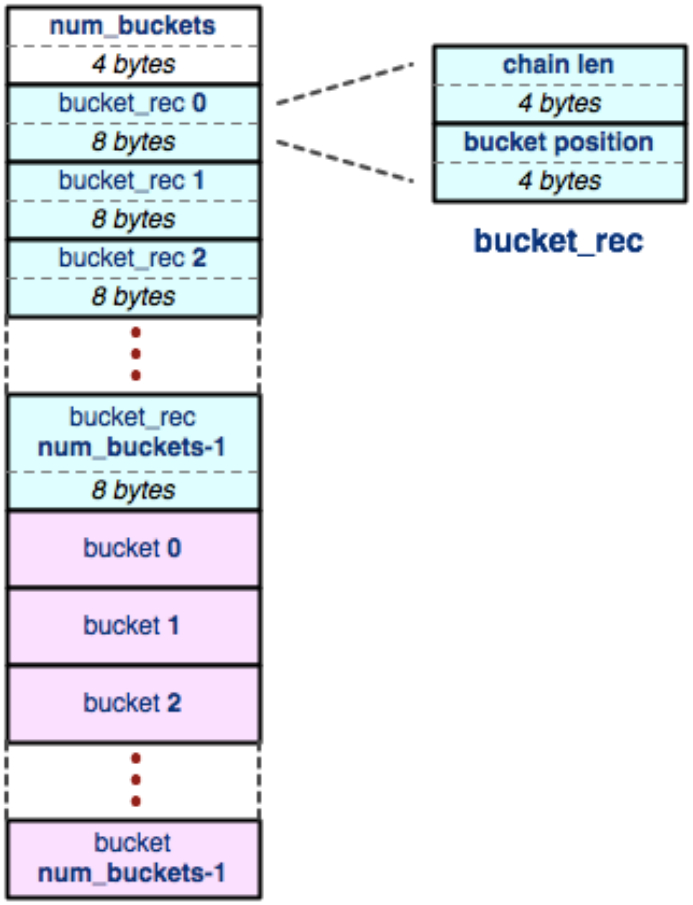
```
0000000: cafe f00d 1c42 4620 0000 205b 0000 075d  .....BF .. [...]
0000010: 0000 0400 0000 0000 0000 2014 0000 0001  .......... .....
0000020: 0000 2014 0000 0001 0000 2031 0000 0001  .. ....... 1....
0000030: 0000 204e 0000 0000 0000 206b 0000 0000  .. N...... k....
0000040: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
0000050: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
```

DocTable & MemIndex

- At their core, both DocTable & MemIndex are HashTables.
- Lets first look at how we write a HashTable.

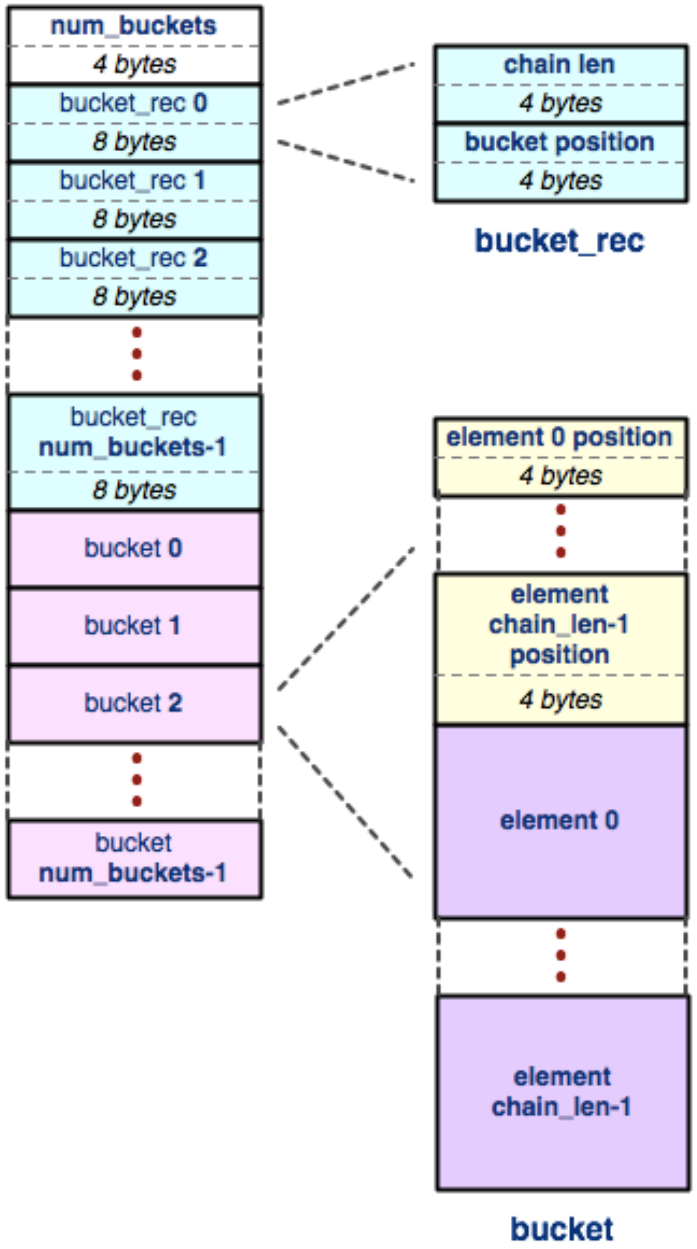
HashTable

- HashTable can have varying amount of buckets, so start with num_buckets.
- Buckets can be of varying lengths. To know the offset, we store some bucket records.



Buckets

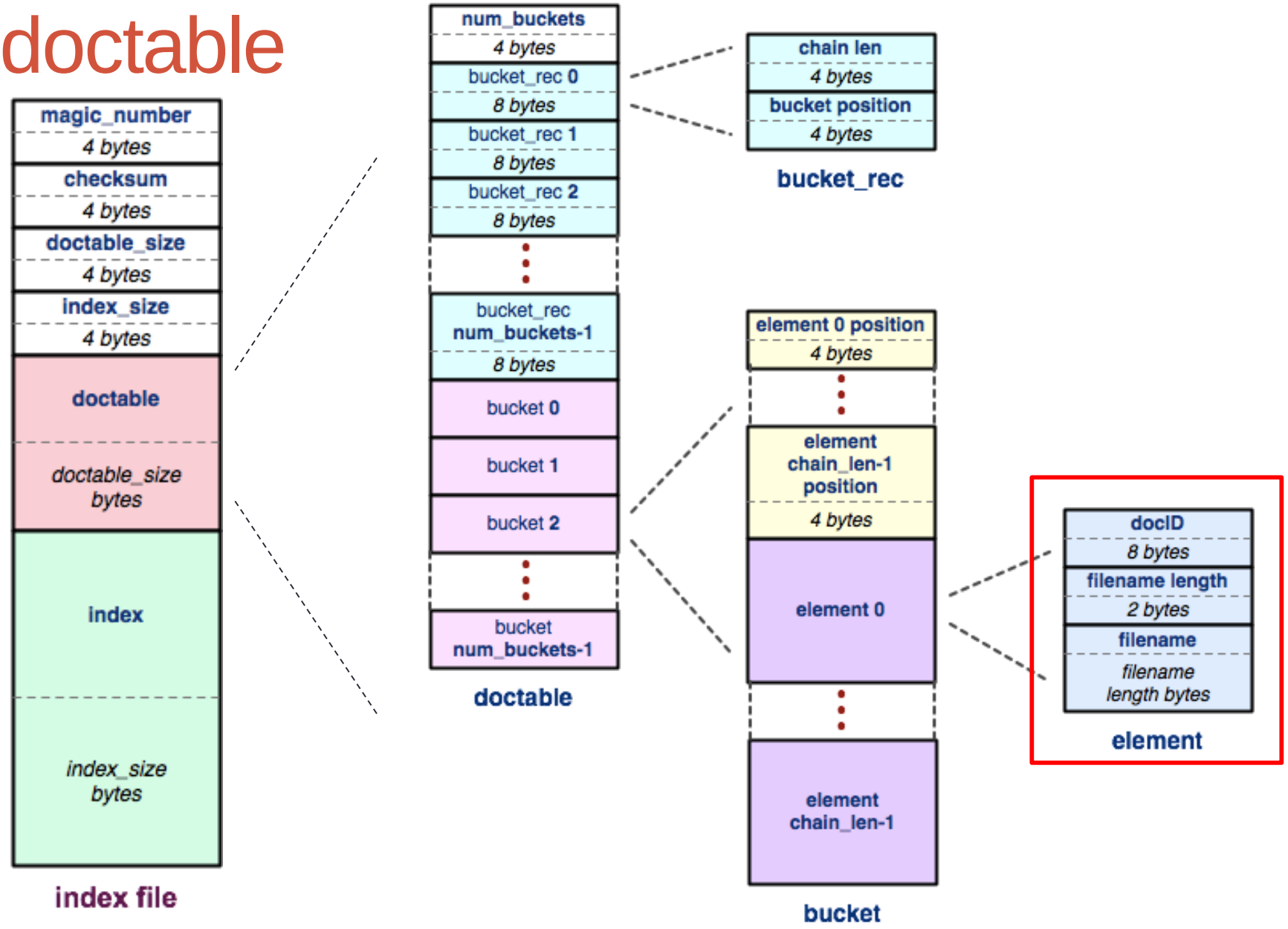
- A bucket is a list that contains elements in the table. Offset to a bucket is found in a bucket record.
- Elements can be of various sizes, so we need to store element positions to know where each element is.



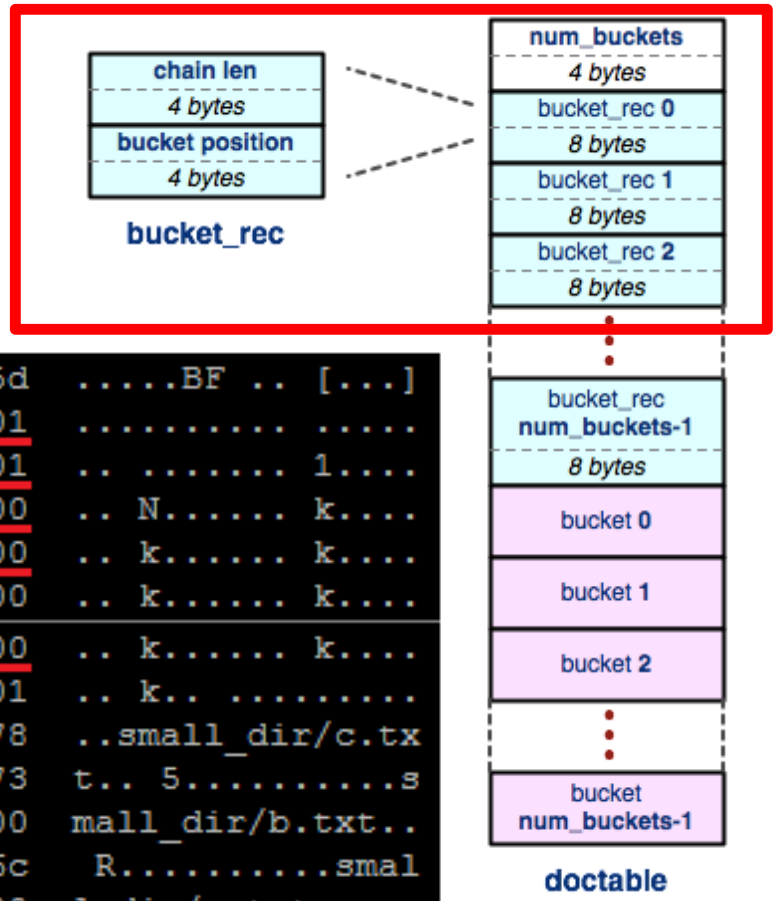
DocTable & MemIndex

- At their core, both DocTable & MemIndex are HashTables.
- The difference between DocTable and MemIndex is entirely what type of element is stored in them.

doctable


magic_number
4 bytes
checksum
4 bytes
doctable_size
4 bytes
index_size
4 bytes
doctable
doctable_size bytes
index
index_size bytes
index file
num_buckets
4 bytes
bucket_rec 0
8 bytes
bucket_rec 1
8 bytes
bucket_rec 2
8 bytes
bucket_rec num_buckets-1
8 bytes
bucket 0
bucket 1
bucket 2
bucket num_buckets-1
doctable
chain len
4 bytes
bucket position
4 bytes
bucket_rec
element 0 position
4 bytes
element chain_len-1 position
4 bytes
element 0
element chain_len-1
bucket
docID
8 bytes
filename length
2 bytes
filename
filename length bytes
element

DocTable (Hex)

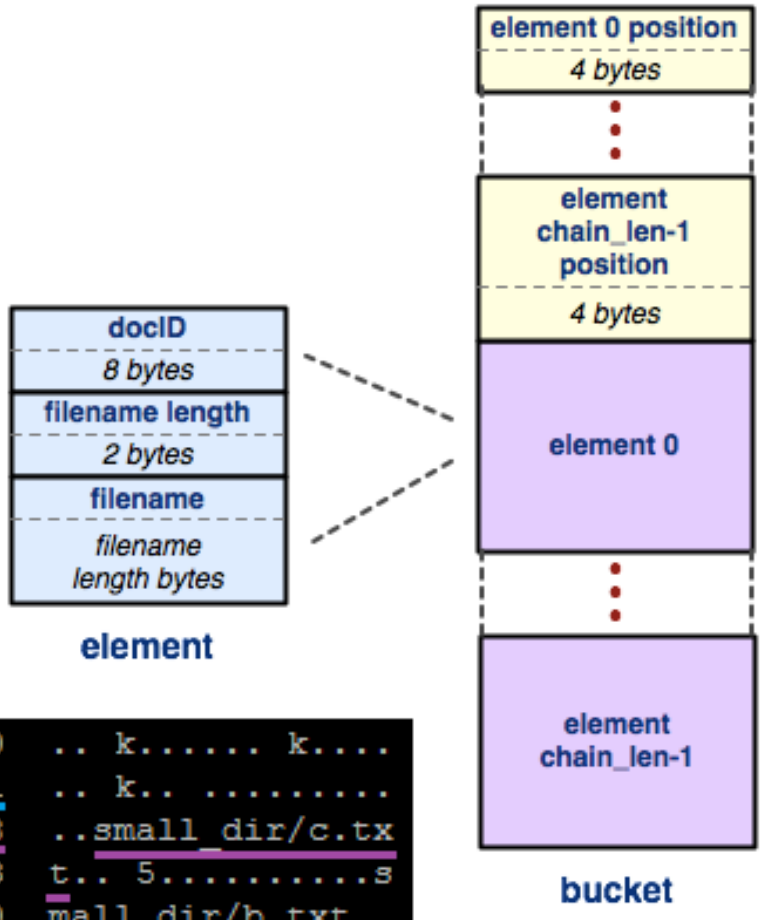



```
0000000: cafe f00d 1c42 4620 0000 205b 0000 075d  .....BF .. [...]
0000010: 0000 0400 0000 0000 0000 2014 0000 0001  .......... .....
0000020: 0000 2014 0000 0001 0000 2031 0000 0001  .. ....... 1....
0000030: 0000 204e 0000 0000 0000 206b 0000 0000  .. N...... k....
0000040: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
0000050: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
0002000: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
0002010: 0000 206b 0000 2018 0000 0000 0000 0001  .. k.. .........
0002020: 000f 736d 616c 6c5f 6469 722f 632e 7478  ..small_dir/c.tx
0002030: 7400 0020 3500 0000 0000 0000 0200 0f73  t.. 5..........s
0002040: 6d61 6c6c 5f64 6972 2f62 2e74 7874 0000  mall_dir/b.txt..
0002050: 2052 0000 0000 0000 0003 000f 736d 616c   R..........smal
0002060: 6c5f 6469 722f 612e 7478 7400 0000 8000  l_dir/a.txt.....
0002070: 0000 0000 0024 6f00 0000 0000 0024 6f00  .....$o......$o.
```

The header

Num buckets (Chain len Bucket offset)*

doctable

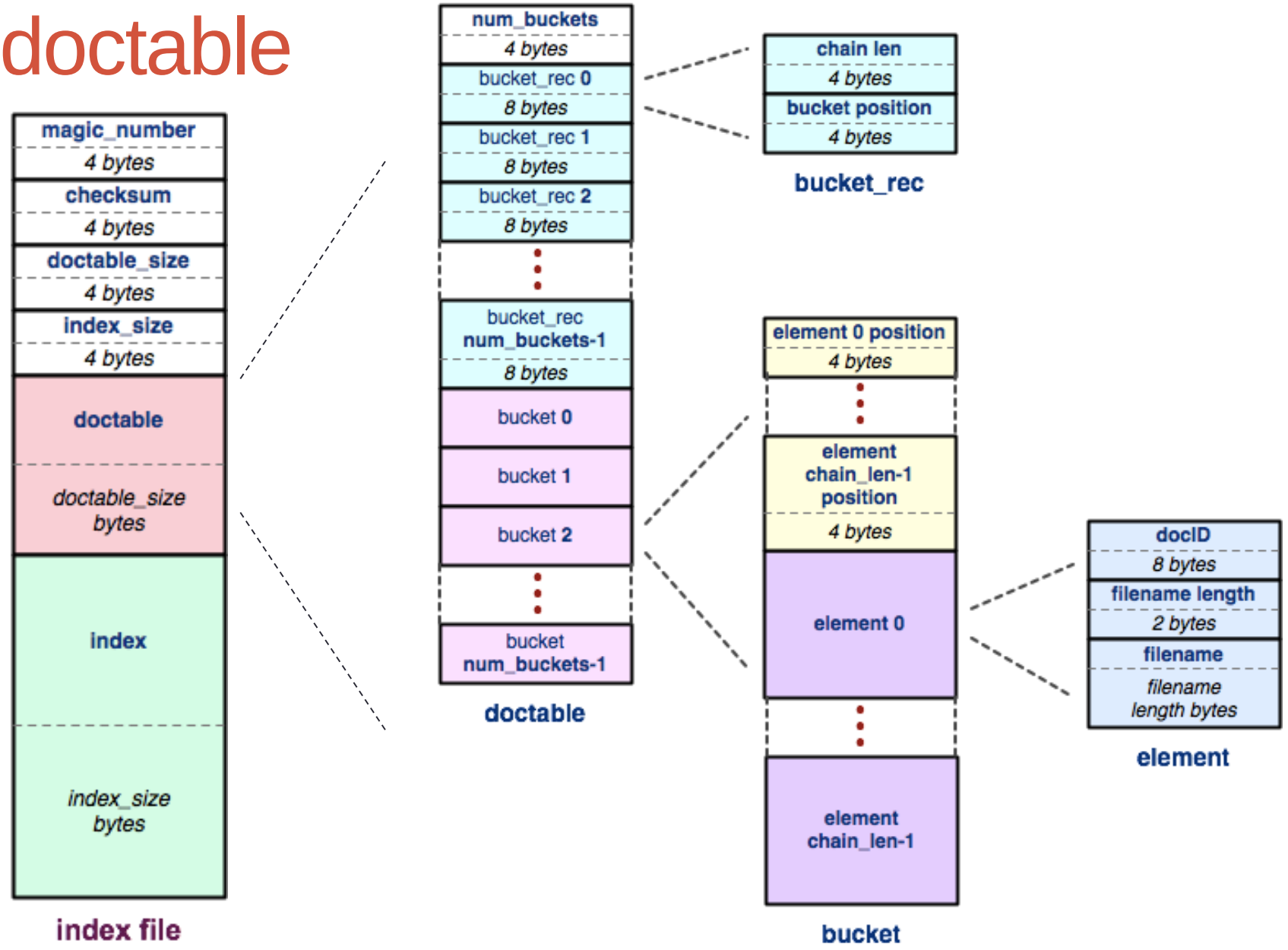



```
0002000: 0000 206b 0000 0000 0000 206b 0000 0000  .. k...... k....
0002010: 0000 206b 0000 2018 0000 0000 0000 0001  .. k.. .........
0002020: 000f 736d 616c 6c5f 6469 722f 632e 7478  ..small_dir/c.tx
0002030: 7400 0020 3500 0000 0000 0000 0200 0f73  t.. 5..........s
0002040: 6d61 6c6c 5f64 6972 2f62 2e74 7874 0000  mall_dir/b.txt..
```

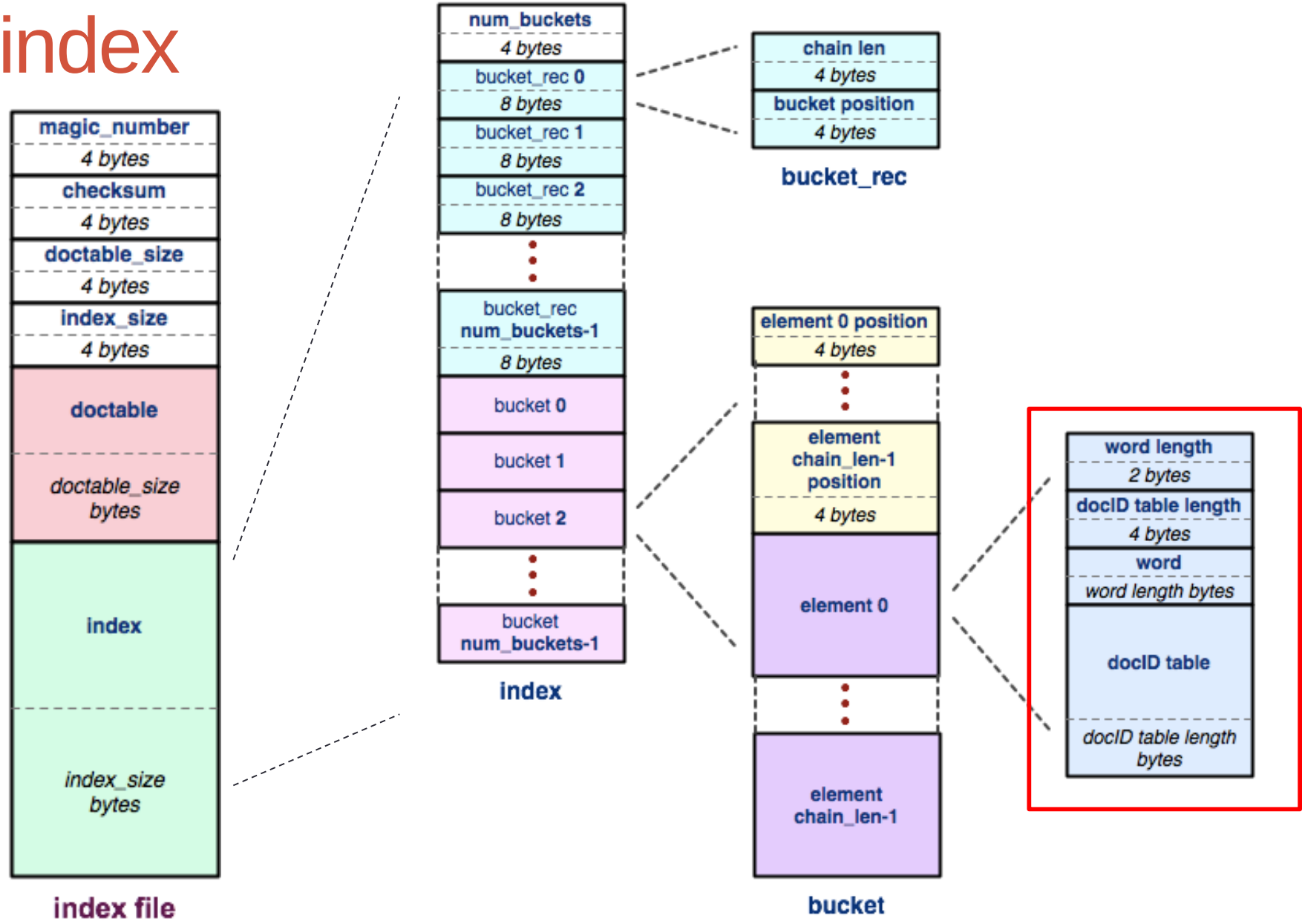
The buckets:

((Element offset)^n (DocID Filename len Filename)^n)*

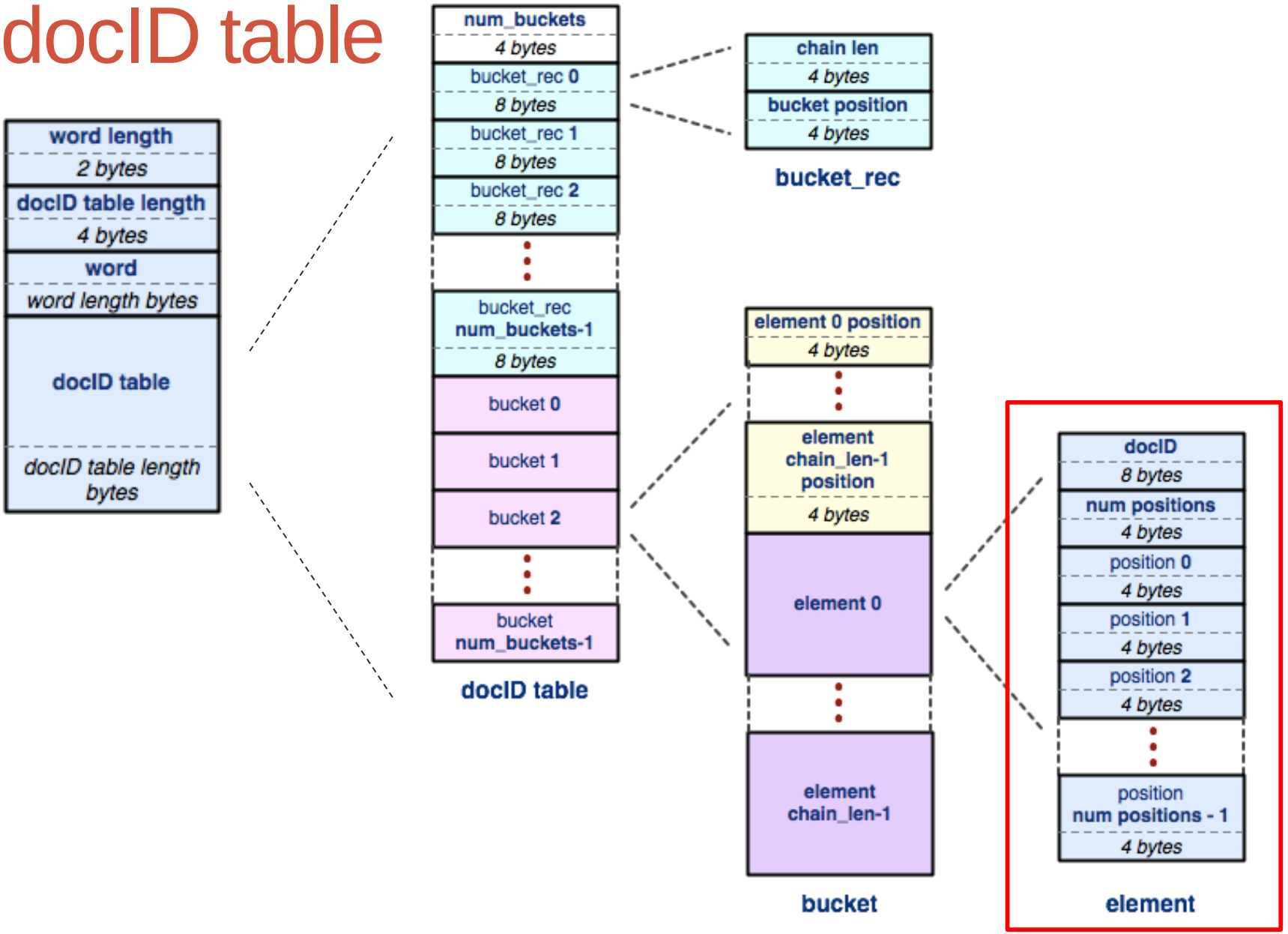
doctable


magic_number
4 bytes
checksum
4 bytes
doctable_size
4 bytes
index_size
4 bytes
doctable
doctable_size bytes
index
index_size bytes
index file
num_buckets
4 bytes
bucket_rec 0
8 bytes
bucket_rec 1
8 bytes
bucket_rec 2
8 bytes
bucket_rec num_buckets-1
8 bytes
bucket 0
bucket 1
bucket 2
bucket num_buckets-1
doctable
chain len
4 bytes
bucket position
4 bytes
bucket_rec
element 0 position
4 bytes
element chain_len-1 position
4 bytes
element 0
element chain_len-1
bucket
docID
8 bytes
filename length
2 bytes
filename
filename length bytes
element

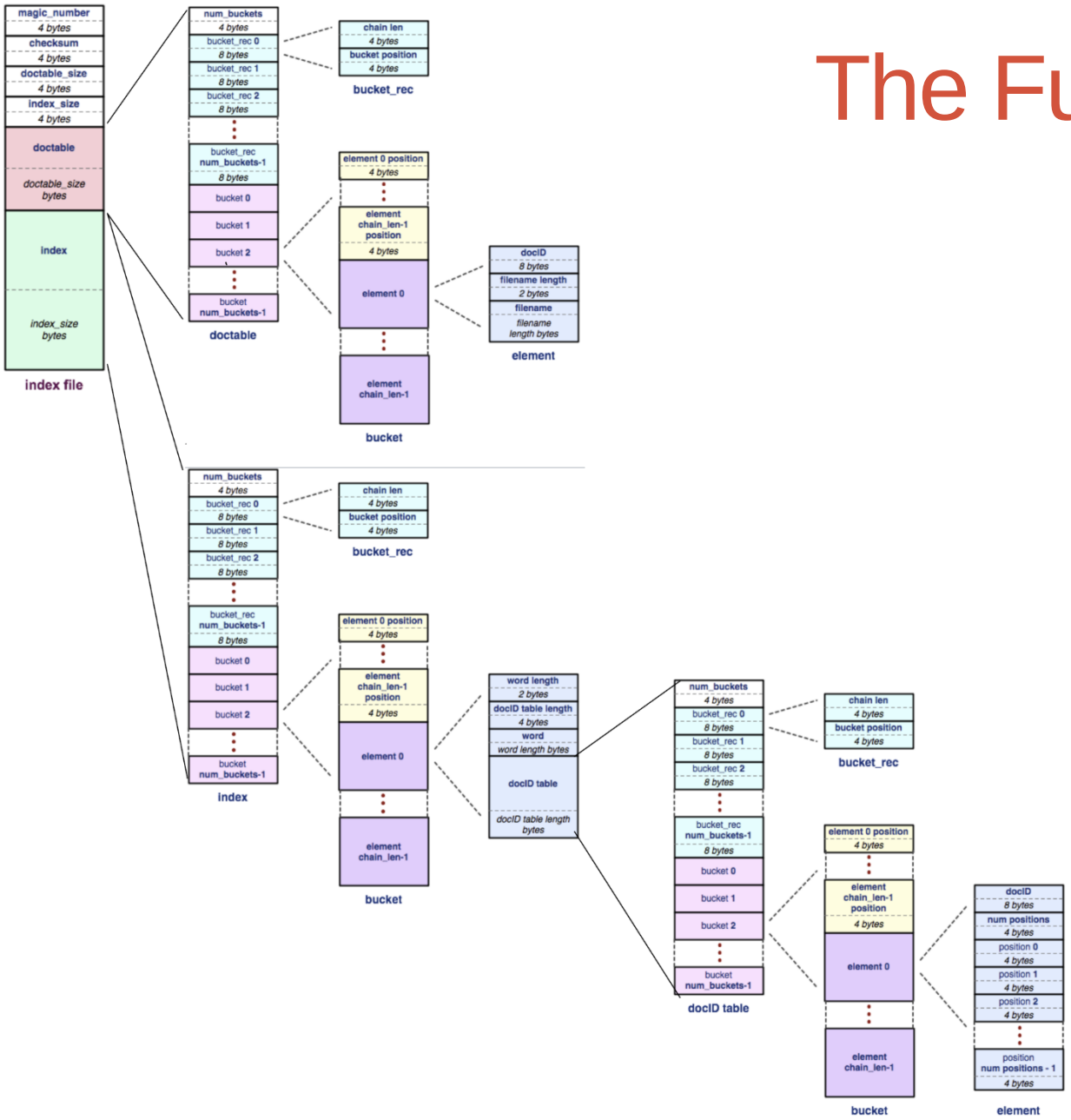
index


magic_number
4 bytes
checksum
4 bytes
doctable_size
4 bytes
index_size
4 bytes
doctable
doctable_size bytes
index
index_size bytes
index file
num_buckets
4 bytes
bucket_rec 0
8 bytes
bucket_rec 1
8 bytes
bucket_rec 2
8 bytes
bucket_rec num_buckets-1
8 bytes
bucket 0
bucket 1
bucket 2
bucket num_buckets-1
index
chain len
4 bytes
bucket position
4 bytes
bucket_rec
element 0 position
4 bytes
element chain_len-1 position
4 bytes
element 0
element chain_len-1
bucket
word length
2 bytes
docID table length
4 bytes
word
word length bytes
docID table
docID table length bytes

docID table


word length
2 bytes
docID table length
4 bytes
word
word length bytes
docID table
docID table length bytes
num_buckets
4 bytes
bucket_rec 0
8 bytes
bucket_rec 1
8 bytes
bucket_rec 2
8 bytes
bucket_rec num_buckets-1
8 bytes
bucket 0
bucket 1
bucket 2
bucket num_buckets-1
docID table
chain len
4 bytes
bucket position
4 bytes
bucket_rec
element 0 position
4 bytes
element chain_len-1 position
4 bytes
element 0
element chain_len-1
bucket
docID
8 bytes
num positions
4 bytes
position 0
4 bytes
position 1
4 bytes
position 2
4 bytes
position num positions - 1
4 bytes
element

The Full Picture



HW Tips

- When Writing, you should (almost) always:
  1. .toDiskFormat()
  2. fseek()
  3. fwrite()
- When Reading, you should (almost) always:
  1. fseek()
  2. fread()
  3. .toHostFormat()

- The most common bugs in the hw involve forgetting to change byte ordering, or forgetting to fseek().

Hex View Exercise

- Split up into break out rooms.
- Take a look at https://courses.cs.washington.edu/courses/cse333/20sp/sections/sec06.idx
  - Log into attu, use wget to download the file, then look into it.
- Try to figure out:
  - How many documents are in this index?
  - Which words are in each document?

Hex View Exercise

- Split up into break out rooms.
- Take a look at https://courses.cs.washington.edu/courses/cse333/20sp/sections/sec06.idx
  - Log into attu, use wget to download the file, then look into it.
- Try to figure out:
  - How many documents are in this index?
  - Which words are in each document?

- **Answer: This index file was built off of test_tree/tiny**