

CSE 333

Section 6

HW3 Overview, Casting

Section Plan

- Casting
- HW 3 Overview

Casting!

Casting in C++

Four different casts that are more explicit:

1. `static_cast<to_type>(expression)`
2. `dynamic_cast<to_type>(expression)`
3. `const_cast<to_type>(expression)`
4. `reinterpret_cast<to_type>(expression)`

When programming in C++, you should use these casts!

Static Cast

```
static_cast<to_type>(expression)
```

Used to:

1) Convert pointers of *related* types

```
Base* b = static_cast<Base*>(new Derived);
```

- compiler error if types aren't related

2) Non-pointer conversion

```
int qt = static_cast<int>(3.14);
```

Static Cast

```
static_cast<to_type>(expression)
```

[!] Be careful when *casting down*:

```
Derived* d = static_cast<Derived*>(new Base) ;  
    d->y = 5;
```

- compiler will let you do this
- dangerous if you want to do things defined in **Derived**, but not in **Base**!

Dynamic Cast

```
dynamic_cast<to_type>(expression)
```

Used to:

1) Convert pointers of *related* types

```
Base* b = dynamic_cast<Base*>(new Derived);
```

- *compiler* error if types aren't related

- at *runtime*, returns `nullptr` if it is actually an unsafe downwards cast:

```
Derived* d = dynamic_cast<Derived*>(new Base);
```

Const Cast

`const_cast<to_type>(expression)`

Used to:

1) Add or remove const-ness

```
const int x = 5;
```

```
    const int *ro_ptr = &x
```

```
int *ptr = const_cast<int*>(ro_ptr);
```

Reinterpret Cast

`reinterpret_cast<to_type>(expression)`

Used to:

1) Cast between *incompatible* types

```
int* ptr = 0xDEADBEEF;
```

```
int64_t x = reinterpret_cast<int64_t>(ptr);
```

- types must be of same size
- does not do float-integer conversions

Exercise 1

```
class Base {
public:
    int x;
};
```

```
class Derived : public Base {
public:
    int y;
};
```

```
int64_t x = 0x7fffffffffe870;
char* str = reinterpret_cast<char*>(x);
```

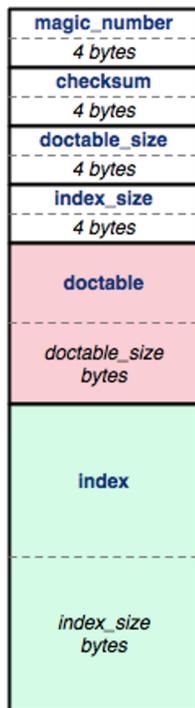
```
void foo(Base *b) {
    Derived *d = dynamic_cast<Derived*>(b);
    // additional code omitted
}
```

```
Derived *d = new Derived;
Base *b = static_cast<Base*>(d);
```

```
double x = 64.382;
int64_t y = static_cast<int64_t>(x);
```

HW 3 Overview!

Index File

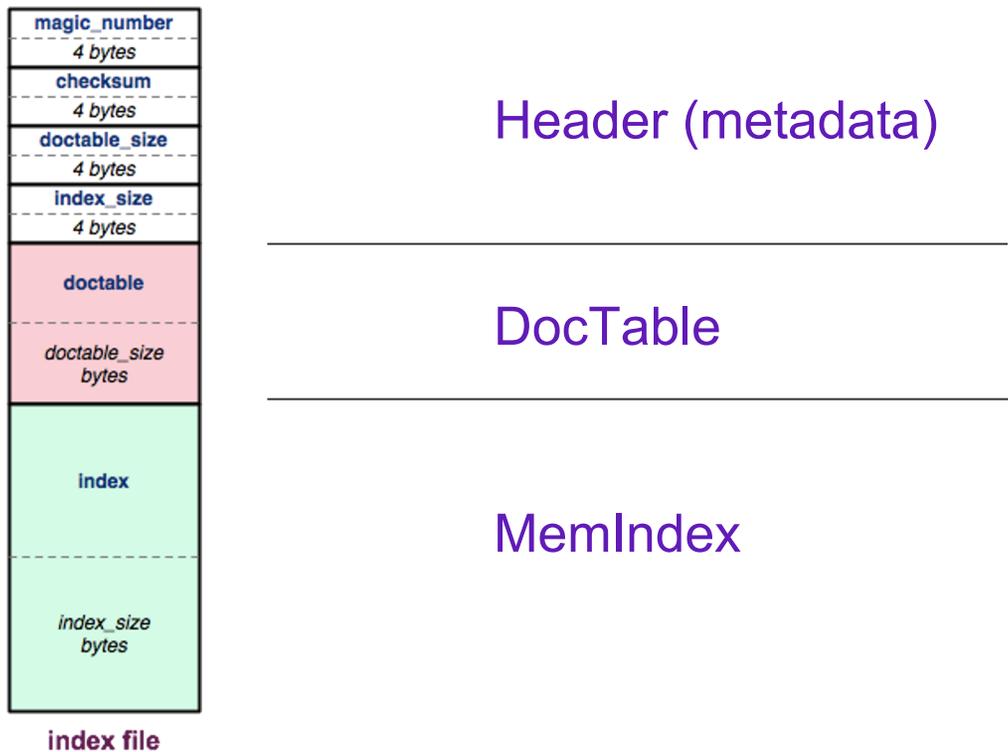


index file

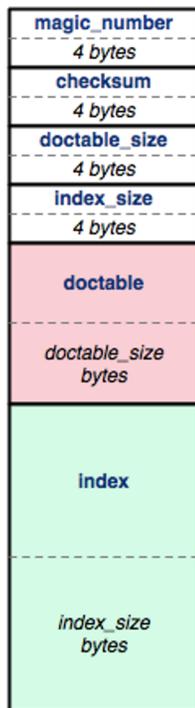
Crawling a file tree in HW2 takes a long time.

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

Index File Components



Index File Header



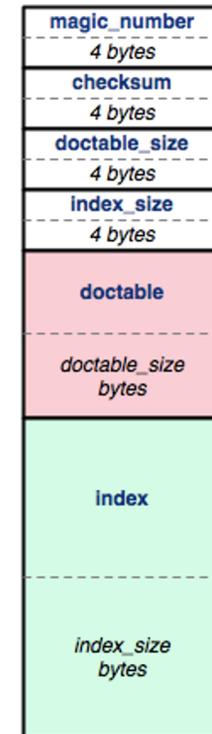
index file

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

Index File Header - HEX

- 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....
```



index file

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”

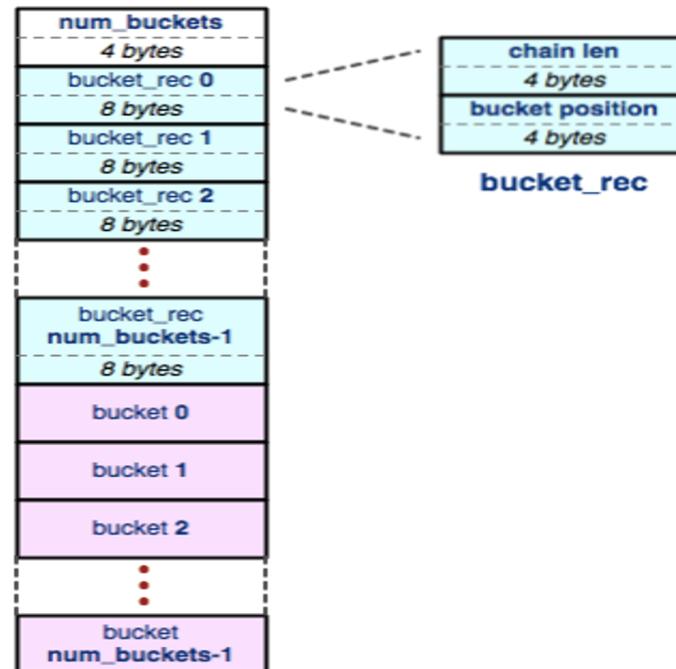
```
00000000: cafe f00d 1c42 4620 0000 205b 0000 075d  ....BF .. [...]
00000010: 0000 0400 0000 0000 0000 2014 0000 0001  ....
00000020: 0000 2014 0000 0001 0000 2031 0000 0001  .. ..... 1....
00000030: 0000 204e 0000 0000 0000 206b 0000 0000  .. N..... k....
00000040: 0000 206b 0000 0000 0000 206b 0000 0000  .. k..... k....
00000050: 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.

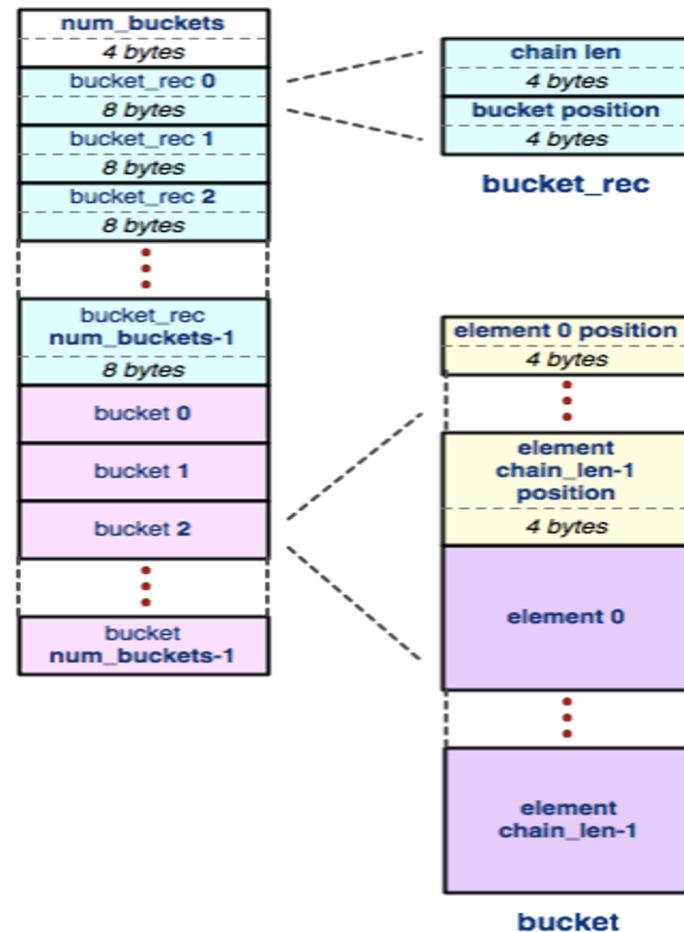
HashTable

- 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

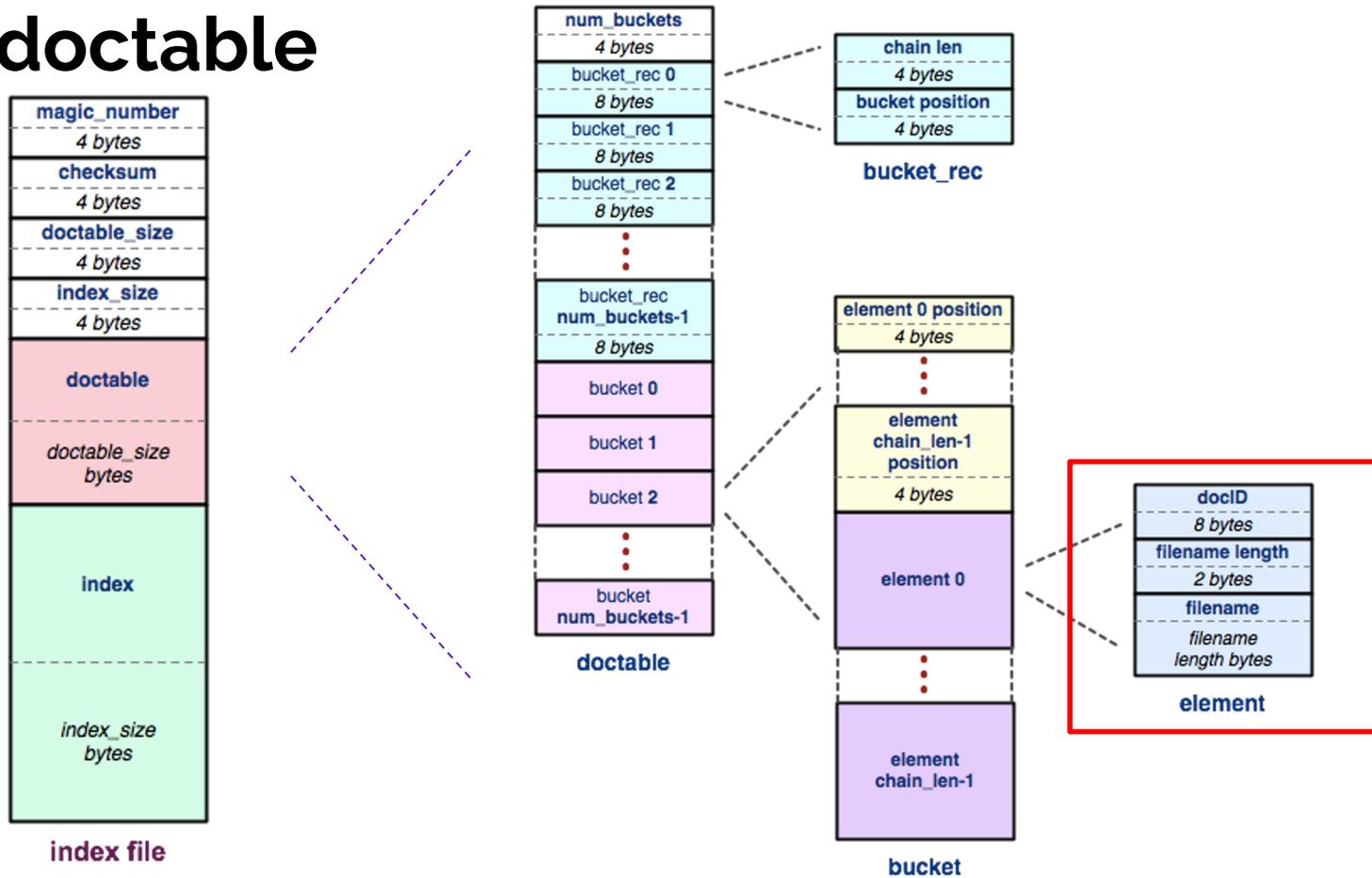
- A bucket is a list that contains elements in the table. Offset to a bucket is found in a bucket record.
- Elements can be of various sizes, so we need to store element positions to know where each element is.



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



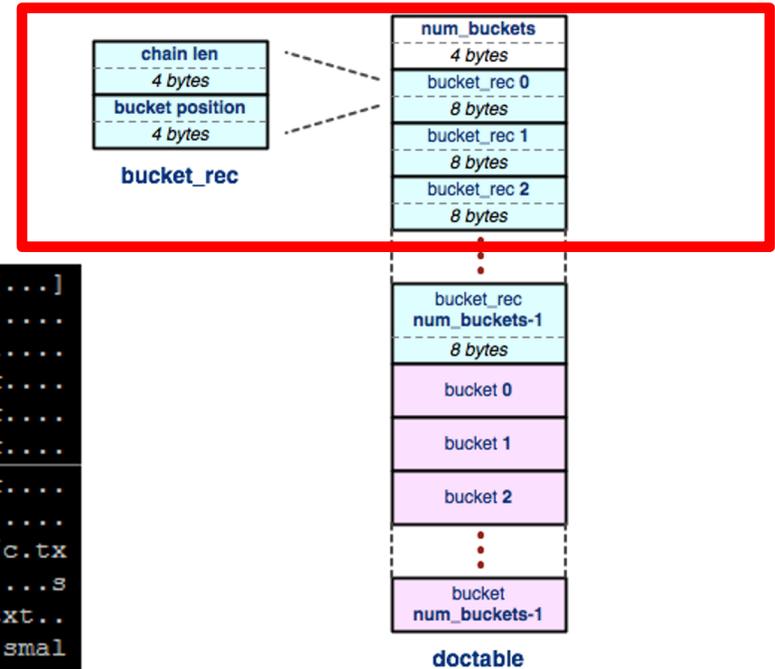
DocTable (Hex)

```

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.

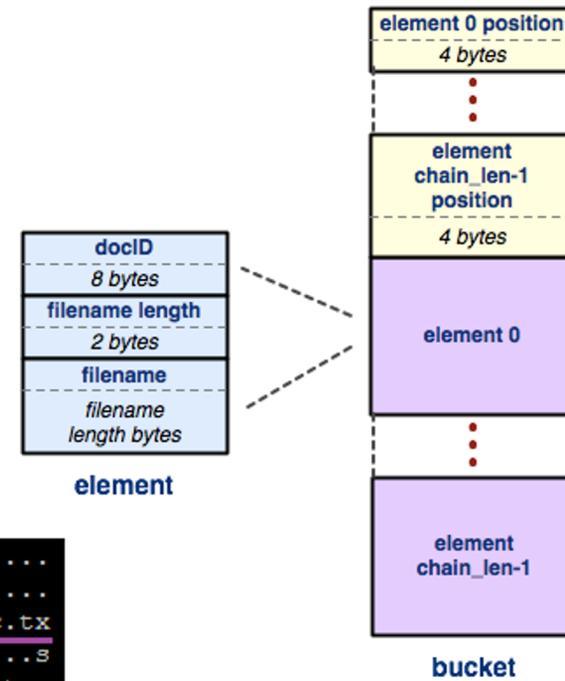
```



The header

Num buckets (Chain len Bucket offset)*

doctable



```

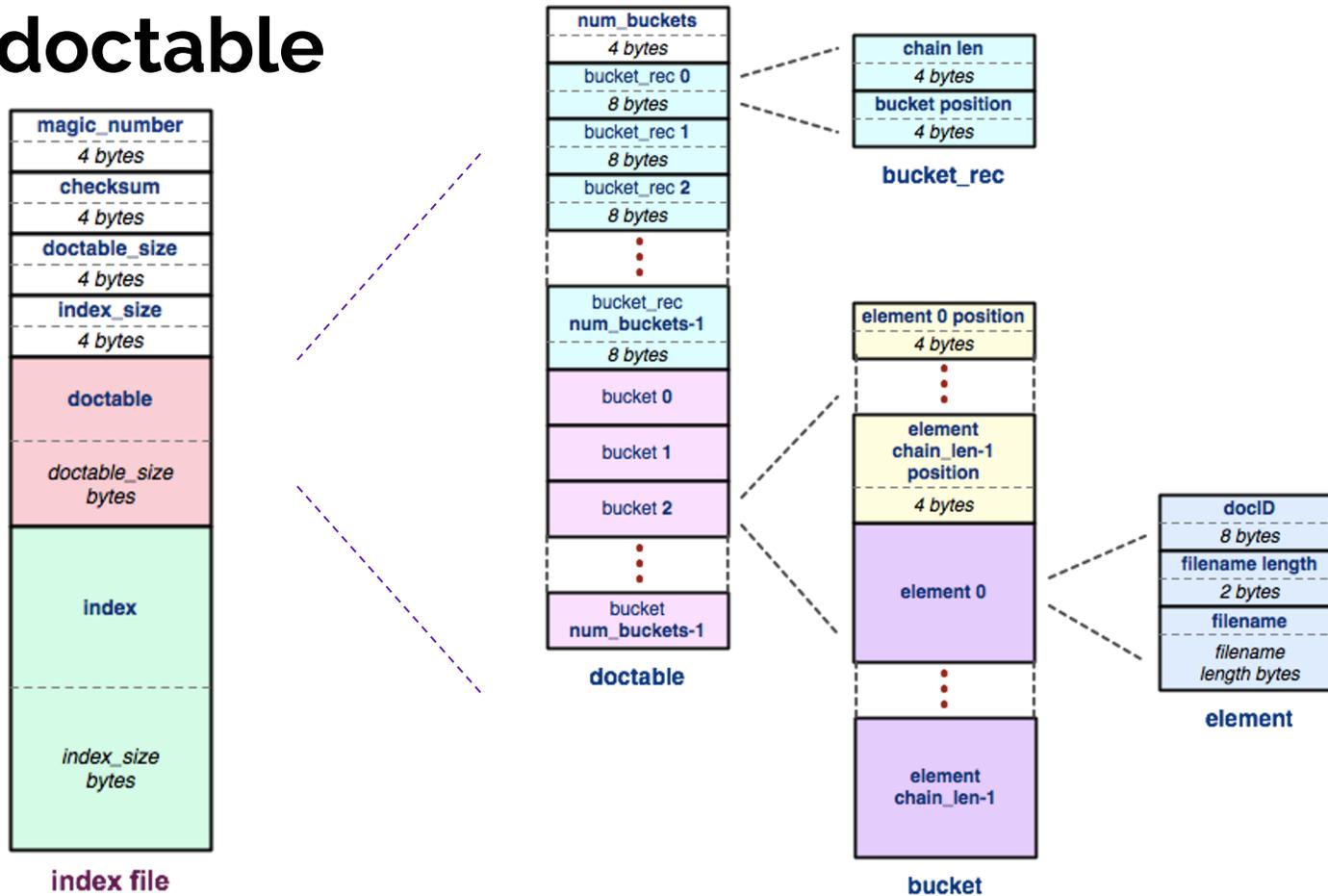
0002000: 0000 206b 0000 0000 0000 206b 0000 0000 .. k..... k....
0002010: 0000 206b 0000 2018 0000 0000 0000 0001 .. k.. .....
0002020: 000f 736d 616c 6c5f 6469 722f 632e 7478 ..small_dir/c.tx
0002030: 7400 0020 3500 0000 0000 0000 0200 0f73 t.. 5.....s
0002040: 6d61 6c6c 5f64 6972 2f62 2e74 7874 0000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

```

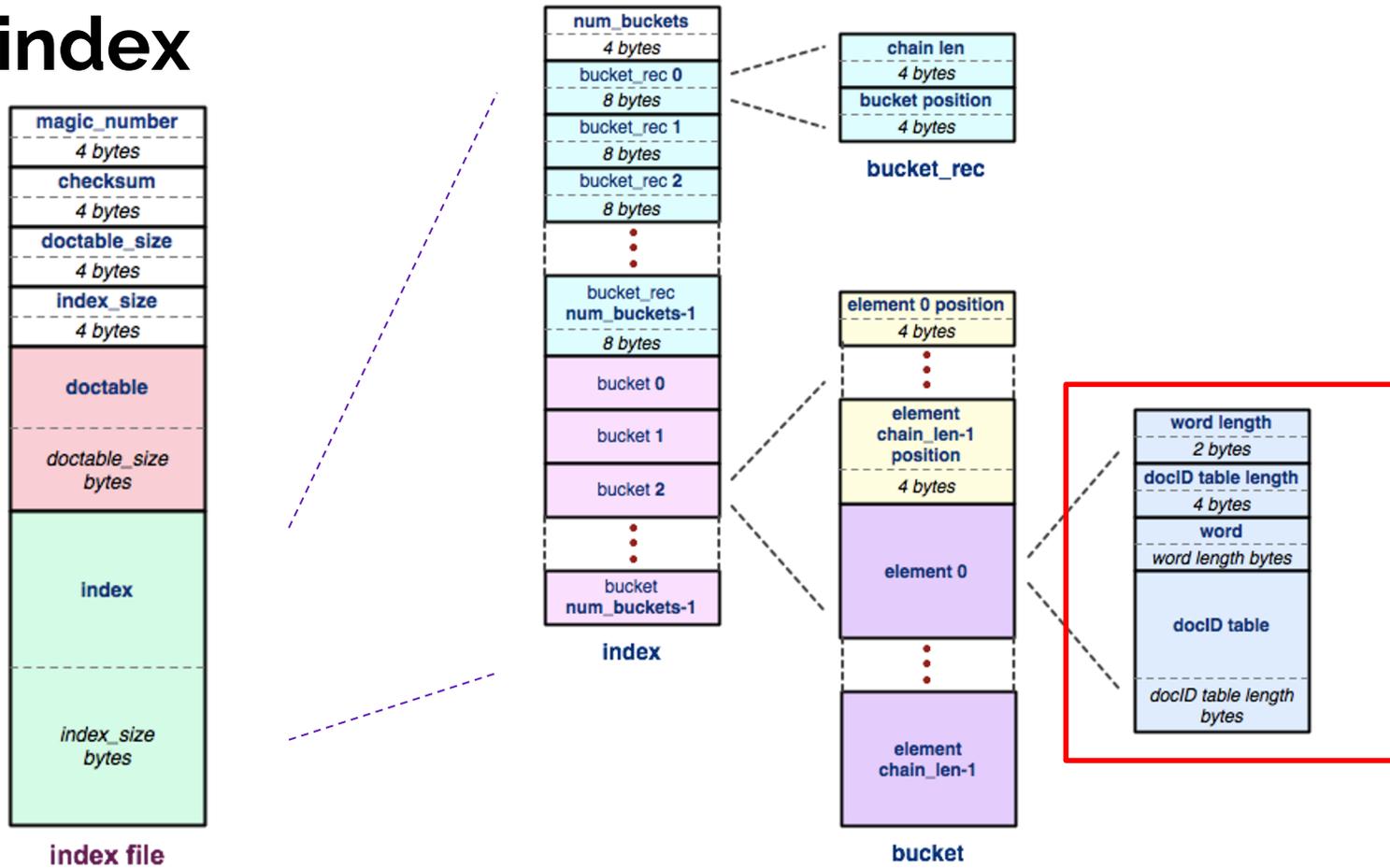
The buckets:

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

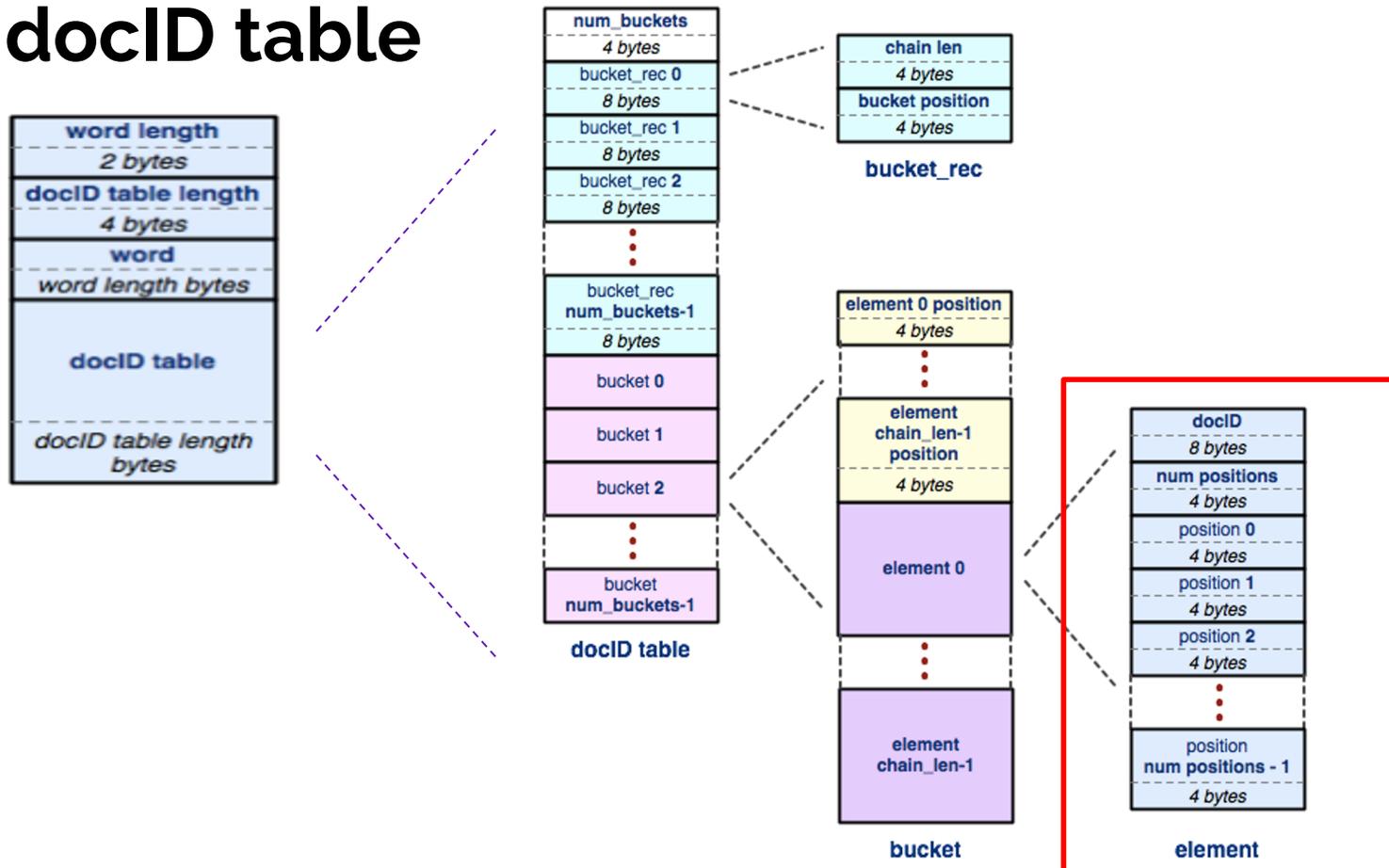
doctable



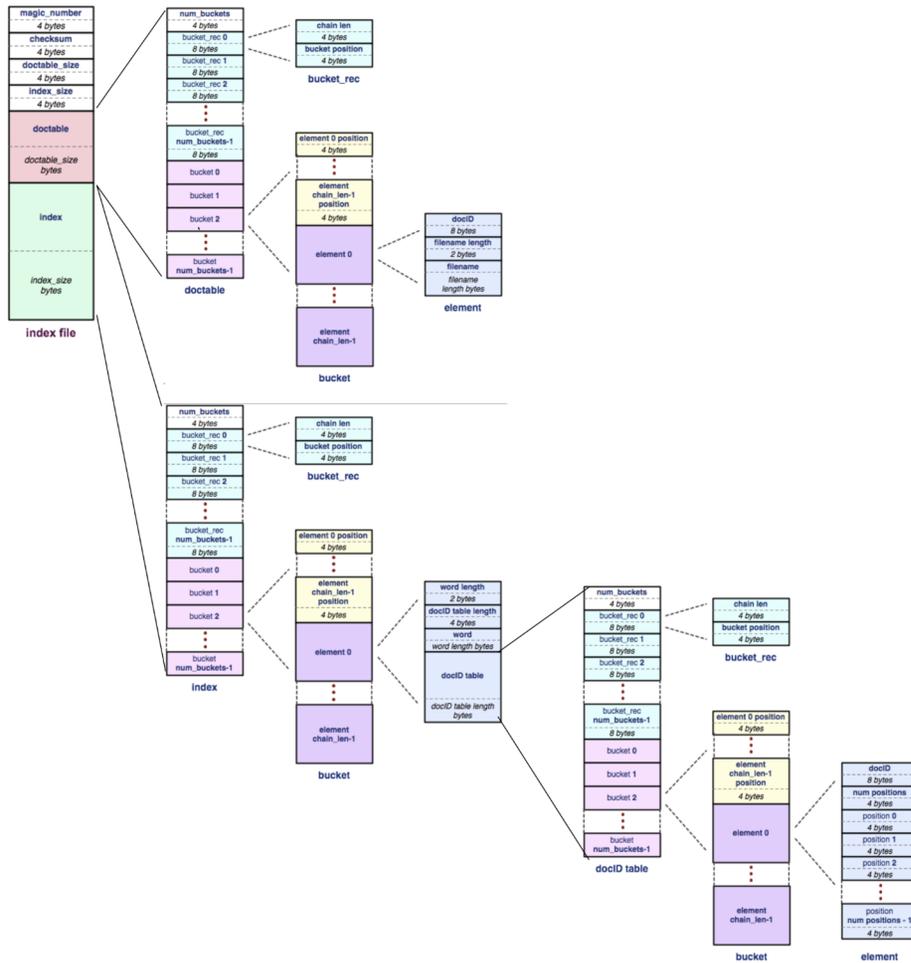
index



docID table



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()`.

Actual directory:
 /minidir
 /tinydir
 goodbye.txt
 hello.txt

```
[ @attu2 hw3]$ xxd test2.idx
0000000: cafe f00d f267 0e99 0000 004a 0000 0093 .....g.....]....
0000010: 0000 0002 0000 0001 0000 0024 0000 0001 .....$....
0000020: 0000 0044 0000 0028 0000 0000 0000 0002 ..D...(.....
0000030: 001c 10a4 9501 0000 0000 3015 9501 0000 .....0.....
0000040: 1c00 d0de 0000 0048 0000 0000 0000 0001 .....H.....
0000050: 0012 f01d 9501 0000 0000 0000 0010 0000 .....
0000060: 0001 0000 00de 0000 0000 0000 00e1 0000 .....
0000070: 0000 0000 00e1 0000 0001 0000 00e1 0000 .....
0000080: 0000 0000 00e4 0000 0000 0000 00e4 0000 .....
0000090: 0000 0000 00e4 0000 0000 0000 00e4 0000 .....
00000a0: 0000 0000 00e4 0000 0000 0000 00e4 0000 .....
00000b0: 0000 0000 00e4 0000 0001 0000 00e4 0000 .....
00000c0: 0000 0000 00e7 0000 0001 0000 00e7 0000 .....
00000d0: 0000 0000 00ea 0000 0001 0000 00ea 0000 .....
00000e0: 0000 0000 0000 0000 0000 0000 00ee 0001 .....
00000f0: 0000 0000 0000 0100 0000 0000 0100 0000 .....
0000100: 0100 0001 0500 0001 0900 0000 0000 0000 .....
0000110: 0100 0100 0500 0000 0f12 .....

[ @attu2 hw3]$ xxd test2.idx
0000000: cafe f00d 159b c4bc 0000 005c 0000 0093 .....\.
0000010: 0000 0002 0000 0001 0000 0024 0000 0001 .....$....
0000020: 0000 004d 0000 0028 0000 0000 0000 0002 ..M...(.....
0000030: 001b 6d69 6e69 6469 722f 7469 6e79 6469 ..minidir/tinydi
0000040: 722f 676f 6f64 6279 652e 7478 7400 0000 r/goodbye.txt...
0000050: 5100 0000 0000 0000 0100 116d 696e 6964 Q.....minid
0000060: 6972 2f68 656c 6c6f 2e74 7874 0000 0010 ir/hello.txt...
0000070: 0000 0001 0000 00f0 0000 0000 0000 00f3 .....
0000080: 0000 0000 0000 00f3 0000 0001 0000 00f3 .....
0000090: 0000 0000 0000 00f6 0000 0000 0000 00f6 .....
00000a0: 0000 0000 0000 00f6 0000 0000 0000 00f6 .....
00000b0: 0000 0000 0000 00f6 0000 0000 0000 00f6 .....
00000c0: 0000 0000 0000 00f6 0000 0001 0000 00f6 .....
00000d0: 0000 0000 0000 00f9 0000 0001 0000 00f9 .....
00000e0: 0000 0000 0000 00fc 0000 0001 0000 00fc .....
00000f0: 0000 0000 0000 0000 0000 0000 0000 0100 .....
0000100: 0004 0000 0000 0000 0400 0000 0000 0400 .....
0000110: 0000 0000 0001 2f00 0000 0100 0001 2f00 ...../...../
0000120: 0000 0000 0001 3200 0000 0000 0001 3200 .....2.....2.
0000130: 0500 0000 2700 0000 0000 0100 0101 .....'
```

```
[ @attu2 hw3]$ xxd test2.idx
0000000: cafe f00d 080a 4d2a 0000 005c 0000 01fe .....M*...\....
0000010: 0000 0002 0000 0001 0000 0024 0000 0001 .....$....
0000020: 0000 004d 0000 0028 0000 0000 0000 0002 ..M...(.....
0000030: 001b 6d69 6e69 6469 722f 7469 6e79 6469 ..minidir/tinydi
0000040: 722f 676f 6f64 6279 652e 7478 7400 0000 r/goodbye.txt...
0000050: 5100 0000 0000 0000 0100 116d 696e 6964 Q.....minid
0000060: 6972 2f68 656c 6c6f 2e74 7874 0000 0010 ir/hello.txt...
0000070: 0000 0001 0000 00f0 0000 0000 0000 0139 .....9.....9
0000080: 0000 0000 0000 0139 0000 0001 0000 0139 .....9.....9
0000090: 0000 0000 0000 017c 0000 0000 0000 017c .....|.....|
00000a0: 0000 0000 0000 017c 0000 0000 0000 017c .....|.....|
00000b0: 0000 0000 0000 017c 0000 0000 0000 017c .....|.....|
00000c0: 0000 0000 0000 017c 0000 0001 0000 017c .....|.....|
00000d0: 0000 0000 0000 01c7 0000 0001 0000 01c7 .....
00000e0: 0000 0000 0000 0223 0000 0001 0000 0223 .....#.....#
00000f0: 0000 00f4 0007 0000 0038 60c4 a000 0000 .....8`.....
0000100: 0000 0000 0400 0000 0000 0001 2500 0000 .....%.....%
0000110: 0000 0001 2500 0000 0100 0001 2500 0000 .....%.....%
0000120: 0000 0001 3900 0001 2900 0000 0000 0000 .....9...9.....
0000130: 0200 0000 0100 0000 0000 0001 3d00 0100 .....=.....
0000140: 0000 3870 0000 0004 0000 0000 0000 0168 ..8p.....h
0000150: 0000 0001 0000 0168 0000 0000 0000 017c .....h.....|
0000160: 0000 0000 0000 017c 0000 016c 0000 0000 .....|...l....
0000170: 0000 0001 0000 0001 0000 000f 0000 0180 .....
0000180: 0005 0000 003c f03a a000 0000 0000 0400 .....<:.....
0000190: 0000 0000 0001 af00 0000 0100 0001 af00 .....
00001a0: 0000 0000 0001 c700 0000 0000 0001 c700 .....
00001b0: 0001 b300 0000 0000 0000 0100 0000 0200 .....
00001c0: 0000 0000 0000 1100 0001 cb00 0600 0000 .....
00001d0: 4cb0 3ba0 0000 0000 0000 0400 0000 0000 L.j.....
00001e0: 0001 fb00 0000 0100 0001 fb00 0000 0100 .....
00001f0: 0002 0f00 0000 0000 0002 2300 0001 ff00 .....#.....
0000200: 0000 0000 0000 0100 0000 0100 0000 0700 .....
0000210: 0002 1300 0000 0000 0000 0200 0000 0100 .....
0000220: 0000 0900 0002 2700 0500 0000 3850 3da0 .....'.....8P=
0000230: 0000 0000 0004 0000 0000 0000 0256 0000 .....V.....V..
0000240: 0001 0000 0256 0000 0000 0000 026a 0000 .....V.....j..
0000250: 0000 0000 026a 0000 025a 0000 0000 0000 .....j...Z.....
0000260: 0001 0000 0001 0000 0017 .....
```

Hex View Exercise

- Split up into break out rooms.
- Take a look at <https://courses.cs.washington.edu/courses/cse333/20au/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/cse333/20au/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**