

## segments\_N

当IndexWriter执行commit()操作后，会生成一个segments\_N文件，该文件描述了当前索引目录中所有有效的段信息文件(active segment info)，即之前文章介绍的[segmentInfo](#)文件，仅仅通过[flush\(\)](#)生成的段成为无效的段信息文件。

索引目录中可能存在多个Segments\_N文件，每个Segment\_N文件代表某次[commit\(\)](#)时的索引状态，其中N值最大的Segments\_N文件代表最新的一次提交，它包含当前索引目录中所有的索引信息。

图1中最新的一次提交生成了Segments\_5文件。

图1:

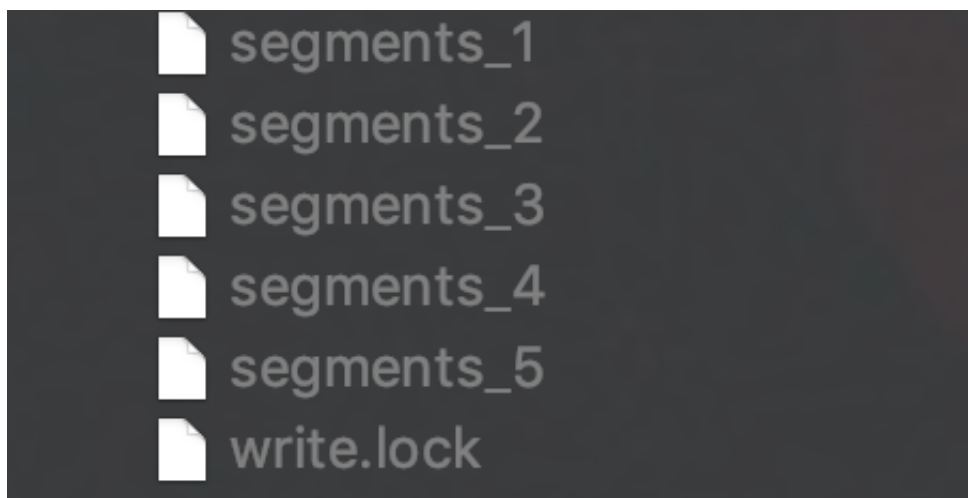
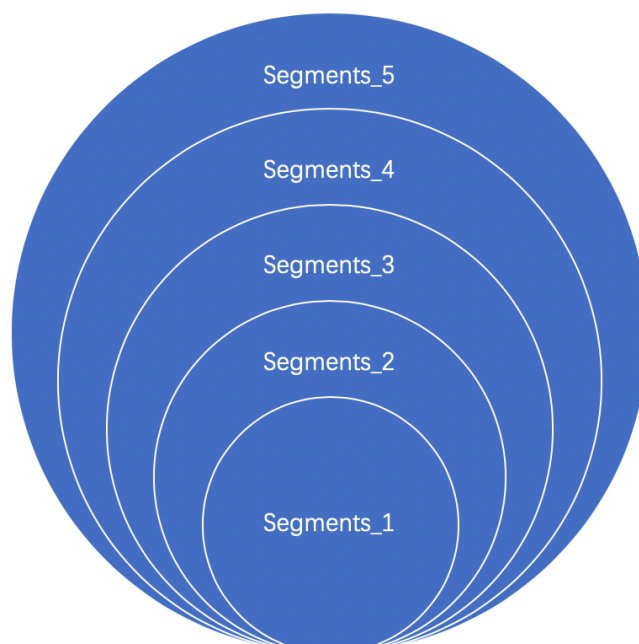


图1中Segments\_N文件包含的索引信息关系如下图:

图2:



一个索引目录中存在多个segments\_N文件的原因大体分为两点：

- 旧的segments\_N暂时不能被删除：原因很多，在后面介绍IndexWriter的文章中会提及
- 使用了非默认的IndexDeletionPolicy：IndexDeletionPolicy提供了一个策略，该策略描述了当一个新的commit()提交后，如果处理旧的提交，Lucene7.5.0中默认使用的是KeepOnlyLastCommitDeletionPolicy，它是IndexDeletionPolicy的其中一个实现，即当有新的提交时，删除前面的提交，比如在图1中，就只会保留segments\_5文件；例如同样作为IndexDeletionPolicy的另一个实现，NoDeletionPolicy，使用该策略就会保留每次的commit()，这么做的好处就相当于设置了每一个commit()检查点，配合CommitUserData(下文会介绍)，我们可以将索引信息恢复到任意一个检查点，缺点是很明显的，如图2中，每一个segments\_N都包含了以往所有的信息，索引目录的大小因此会很大。

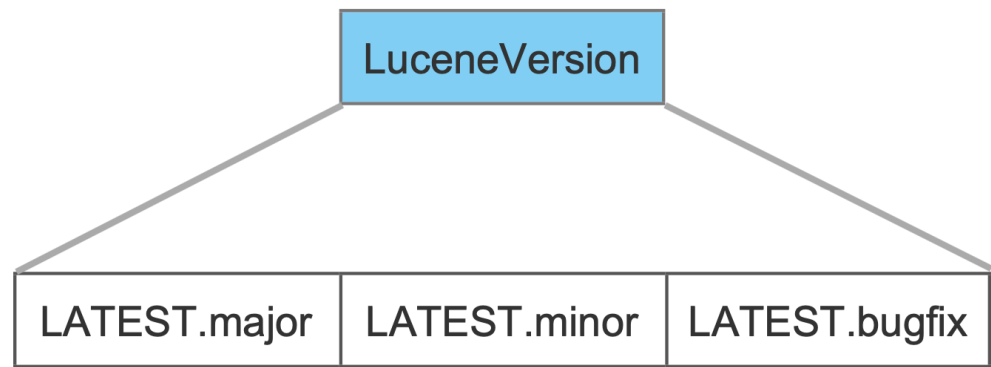
# segments\_N文件的数据结构

图3：



## LuceneVersion

图4：



LuceneVersio描述了当前运行的Lucene版本，比如本文基于Lucene7.5.0写的，那么LuceneVersion的值如下：

- LATEST.major： 7
- LATEST.minor： 5
- LATEST.bugfix： 0

## IndexCreatedVersionMajor

IndexCreatedVersionMajor描述的是创建该segment\_N文件的Lucene的major值，在读取阶段，该segment\_N文件可能被更高版本的Lucene读取，用来检查兼容性。

## Version

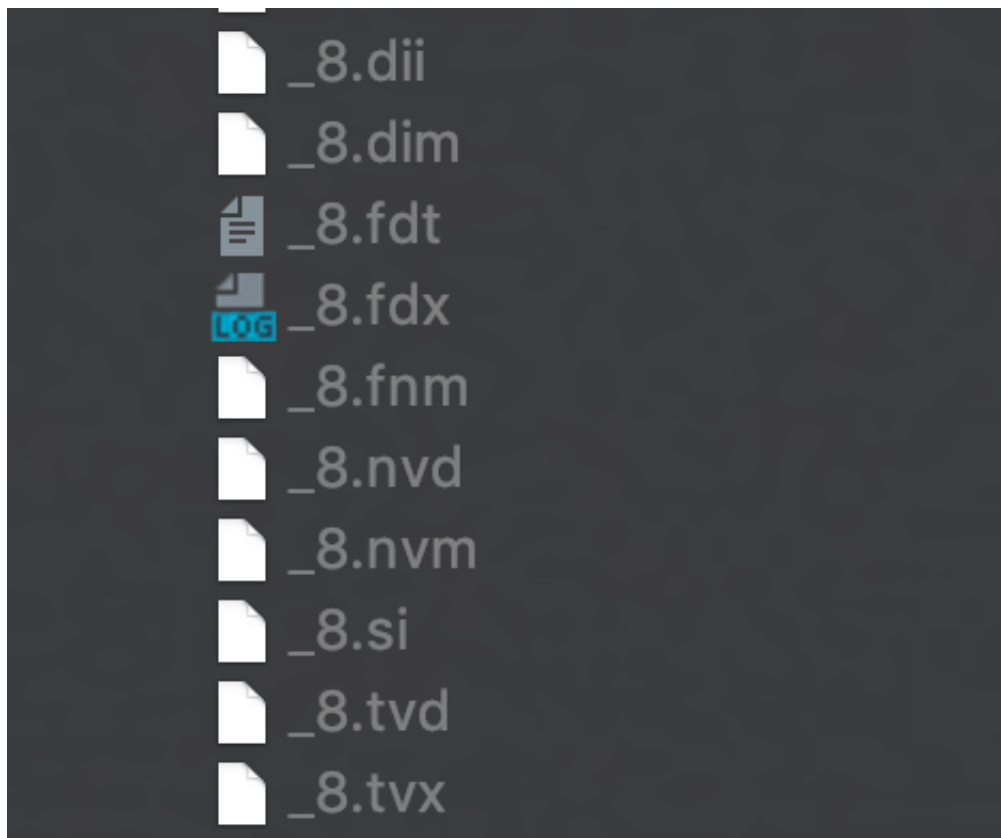
Version描述的是segmentInfos对象发生更改的次数。

segmentInfos对象的概念见文章[近实时搜索NRT \(一\)](#)中流程点 获得所有段的信息集合 SegmentInfos 的介绍。

## NameCounter

NameCounter用来给新的[segmentInfo](#)文件提供名字的前缀值，例如下图中 \_8 即为前缀值。

图5:

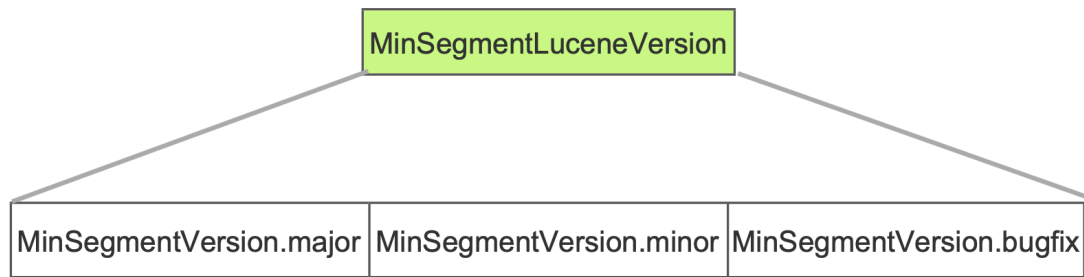


## SegCount

该字段描述了当前索引目录中的有效的段信息文件(active segment info)。

## MinSegmentLuceneVersion

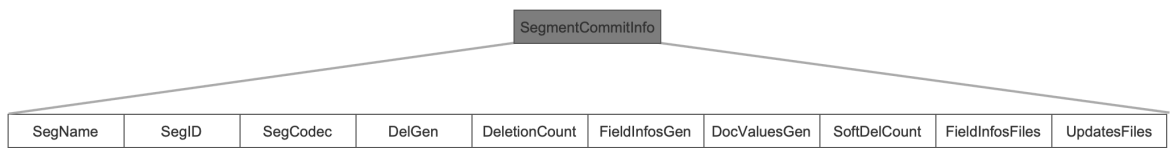
图6:



索引目录中的.si文件的版本可能各不相同，MinSegmentLuceneVersion记录版本最小的，不详细展开，同图4。

## SegmentCommitInfo

图7：



该字段描述了一个segmentInfo文件(.si文件)的信息。

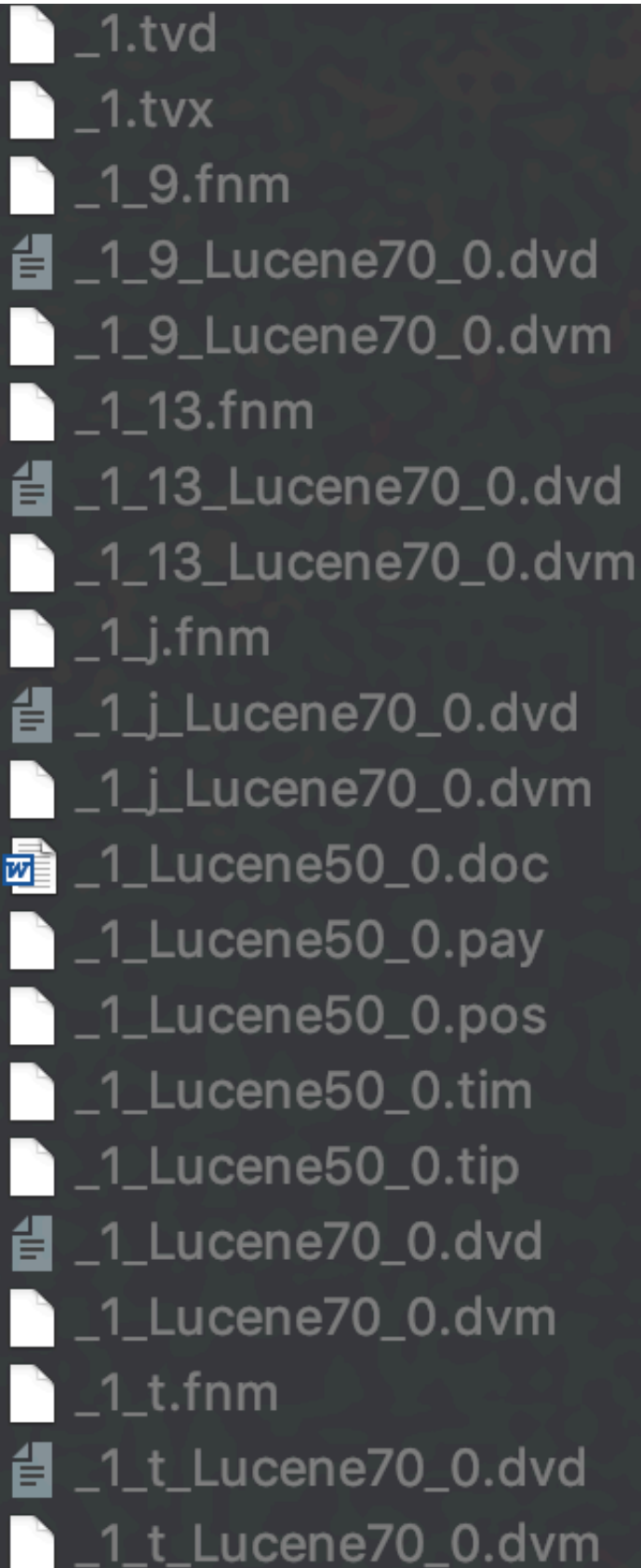
## SegName

该字段描述了segmentInfo文件及对应的其他索引文件的名字前缀，图8中，下面所有的文件属于同一个segment，segName的值为"\_1"

在读取segment\_N文件阶段，通过SegName找到[.si索引文件](#)，结合SegmentCommitInfo就可以获得一个段的完整的信息

图8：





- \_1.tvd
- \_1.tvx
- \_1\_9.fnm
- \_1\_9\_Lucene70\_0.dvd
- \_1\_9\_Lucene70\_0.dvm
- \_1\_13.fnm
- \_1\_13\_Lucene70\_0.dvd
- \_1\_13\_Lucene70\_0.dvm
- \_1\_j.fnm
- \_1\_j\_Lucene70\_0.dvd
- \_1\_j\_Lucene70\_0.dvm
- \_1\_Lucene50\_0.doc
- \_1\_Lucene50\_0.pay
- \_1\_Lucene50\_0.pos
- \_1\_Lucene50\_0.tim
- \_1\_Lucene50\_0.tip
- \_1\_Lucene70\_0.dvd
- \_1\_Lucene70\_0.dvm
- \_1\_t.fnm
- \_1\_t\_Lucene70\_0.dvd
- \_1\_t\_Lucene70\_0.dvm

## SegID

该字段描述了segmentInfo文件的一个唯一标示。

## SegCodec

该字段描述了segmentInfo文件编码值，例如"Lucene70"。

## DelGen

该字段描述了属于同一个segment的`.liv`文件的generation number，该值在后面介绍文档的添加、删除、更新时会给出详细含义。

## DeletionCount

该字段描述了segmentInfo文件中被删除文档的个数。

## FieldInfosGen

该字段描述了属于同一个segment的`.fnm`文件的generation number，当域的信息每发生一次变化，FieldInfosGen的值就会+1，比如说调用了IndexWriter.updateDocValues(..)的方法

## DocValuesGen

该字段描述了属于同一个segment的`.dvm`、`.dvd`文件的generation number，当有调用IndexWriter.updateDocValues(..)该值就会+1，该值在后面介绍DocValues域的更新时会给出详细含义。

## SoftDelCount

该字段记录软删除的文档个数，软删除的概念后面介绍文档的添加、删除、更新时会给出详细含义。

## FieldInfosFiles

如果域的信息发生了变化（更新），那么会记录最新生成的.fnm文件。

## UpdatesFiles

记录发生变化的索引文件，比如调用了IndexWriter.updateDocValues(..)的方法后，会生成新的.dvd、.dvm文件，那么域值跟索引文件名字的信息。

图9：

```
indexWriter.updateDocValues(new Term( fld: "content", text: "c"), new NumericDocValuesField( name: "new", value: 3));
```

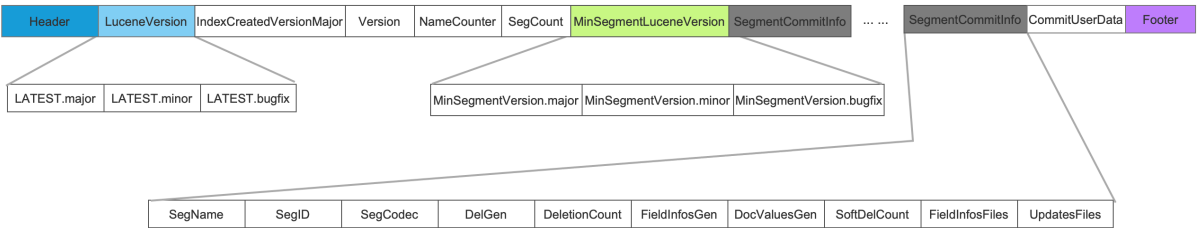
上图中，先找出包含域名为"content"，域值为"c"的文档，然后更新该文档中的NumericDocValuesField域，更新域名跟域值。此操作后，会生成新的.dvd、dvm文件。

## CommitUserData

该字段可以通过调用IndexWriter.setLiveCommitData(...)来在commit()时记录自定义的信息，上文中提到，如果使用了NoDeletionPolicy，那么Lucene会保留每一次commit()时的索引文件信息作为检查点，这样我们可以通过CommitUserData跟Segment\_N来回退到任意的检查点。

# segments\_N文件的总数据结构

图10：



## 结语

至此介绍了本人在业务中接触过的所有的[索引文件](#)（复合文件没有讲...😞），完全深入理解索引文件的所有内容需要了解IndexWriter添加、删除、更新文档、出错的逻辑，而写这篇文章的另一个目的也是为介绍IndexWriter作为预备知识。

[点击下载](#)Markdown文件