1、HTTP有个请求方法Head。它的特点呢？

head响应报文中没有body，但是有响应头head.

作用呢：可以用来获取一些信息，比如获取下载文件的大小。我们在进行文件下载时，就可以先用这个请求方法获取文件大小。

之后根据文件大小，就可以选择一次下载还是断点续传。

2、请求头Header,有个Range可以指定获取文件的字节范围。

Range / Accept-Range

按范围取数据

Accept-Range: bytes 响应报⽂中出现，表示服务器⽀持按字节来取范围数据

Range: bytes=<start>-<end> 请求报⽂中出现，表示要取哪段数据

Content-Range:<start>-<end>/total 响应报⽂中出现，表示发送的是哪段数据

作⽤：断点续传、多线程下载。

3、所谓断点下载，其实也不复杂。注意以下几点内容：

（1）httpURLconnection#setRequestProperty方法通过设置，可以从服务器指定位置读取数据。eg:conn.setRequestProperty("Range", "bytes=" + 500 + "-" + 1000);

（2）普通的File对象并不支持从指定位置写入数据，我们需要使用RandomAccessFile来实现从指定位置给文件写入数据的功能。void seek(long offset)

（3）每次下载时，需要记录断点的位置信息。

1. 断点续传：记录上次上传（下载）节点位置，下次接着该位置继续上传（下载）。多线程断点续传下载则是根据**目标下载文件长度，尽可能地等分给多个线程同时下载文件块**，当各个线程全部完成下载后，**将文件块合并成一个文件**，即目标文件。多线程断点续传不仅为用户避免了断网等突发事故需要重新下载浪费流量的尴尬局面，也大大提高了下载速率，当然，不是线程越多越好，网络带宽才是硬道理！

java，android中可以使用RandomAccessFile类生成一个同目标文件大小的占位文件，以便于各个线程可以同时操作该文件，并写入各线程实时下载的数据。

先网络请求获取文件的长度mFileLength，根据长度借助RandomAccessFile类在本地生成相同长度的占位文件mTmpFile，再根据线程数量THREAD\_COUNT拆分下载任务，最后for循环出THREAD\_COUNT数量的异步请求下载拆分内容（字节）并从mTmpFile的对应位置写入mTmpFile，每个线程（任务）每写入一定的数据后将任务的下载进度写入通过RandomAccessFile生成的对应任务的记录缓存文件中，以便于下次下载读取该线程已下载的进度。

// 创建一个Request

// 设置分段下载的头信息。 Range:做分段数据请求,断点续传指示下载的区间。格式: Range bytes=0-1024或者bytes:0-1024

header("RANGE", "bytes=" + startIndex + "-" + endIndex)，在OkHttp请求头中添加RANGE（范围）参数，告诉服务器需要下载文件内容的始末位置。

1. 问题：如果有4个线程在下载，当剩下一个要下载完成时。这时app被杀掉了。下次回到这里，要怎么操作呢？

反正就是记录最新的位置。没有下载的就重新下载。

在本地记录唯一资源的唯一性。