**Proxy\_Lab**

1. **PartA**

该部分要求实现一个浏览器的Proxy，参考书中第十章和第十一章的内容即可完成。十章介绍了Unix I/O系统和可靠的RIO，在实现代理的时候，主要用到了Rio\_writen(不含缓冲的写入)以及Rio\_readlineb(含缓冲的按行读取)。第十一章介绍了网络编程以及Socket的使用，并给出了实现Web服务器所需的部分示例编码，我们需要实现的是根据题目要求处理浏览器的请求并转发给服务器。需要实现的三个关键函数分别是parse\_uri-解析uri提取出主机名、文件路径和端口号；build\_header-读取请求头中的键值对信息，并按题目要求重新构建请求头；doit-处理浏览器的请求，解析处理重构后发送给服务器，并将服务器返回的结果发送给浏览器。具体的实现思路与代码逻辑参考proxy.c，所有的代码段均含有注释。

1. **PartB**

PartB在PartA的基础上要求加上并发，即基于线程的并发，参考书中第十二章并发编程Posix线程。主要用到的函数有Pthreads\_create()创建线程，以及Pthreads\_detach()分离线程。在主函数中，每监听到一个请求，便创建一个新的线程去处理该请求，将处理请求的相关函数放到线程路线函数中即可。

1. **PartC**

PartC要求在PartB的基础上实现缓存，采用LRU替换原则，即每次遇到新的页面，都将其加入缓存，如果缓存满了则选择一个最久未用的进行替换。如果需要访问的页面已经在缓存中则不向服务器发起请求，直接返回缓存中的页面。这一部分的实现类似于之前的高速缓存。不同的是，Web缓存建立在多线程的前提下，因此需要采用信号量来控制对Cache的互斥访问操作。多个读者可以同时读一个Cache块，但同时只能有一个人写Cache。这是一个PV操作中典型的读写问题，添加两个信号量即可，一个用来控制对Cache块的写操作，一个用来控制对读者数量的修改操作。具体的实现思路与代码逻辑参考proxy.c，所有的代码段均含有注释。