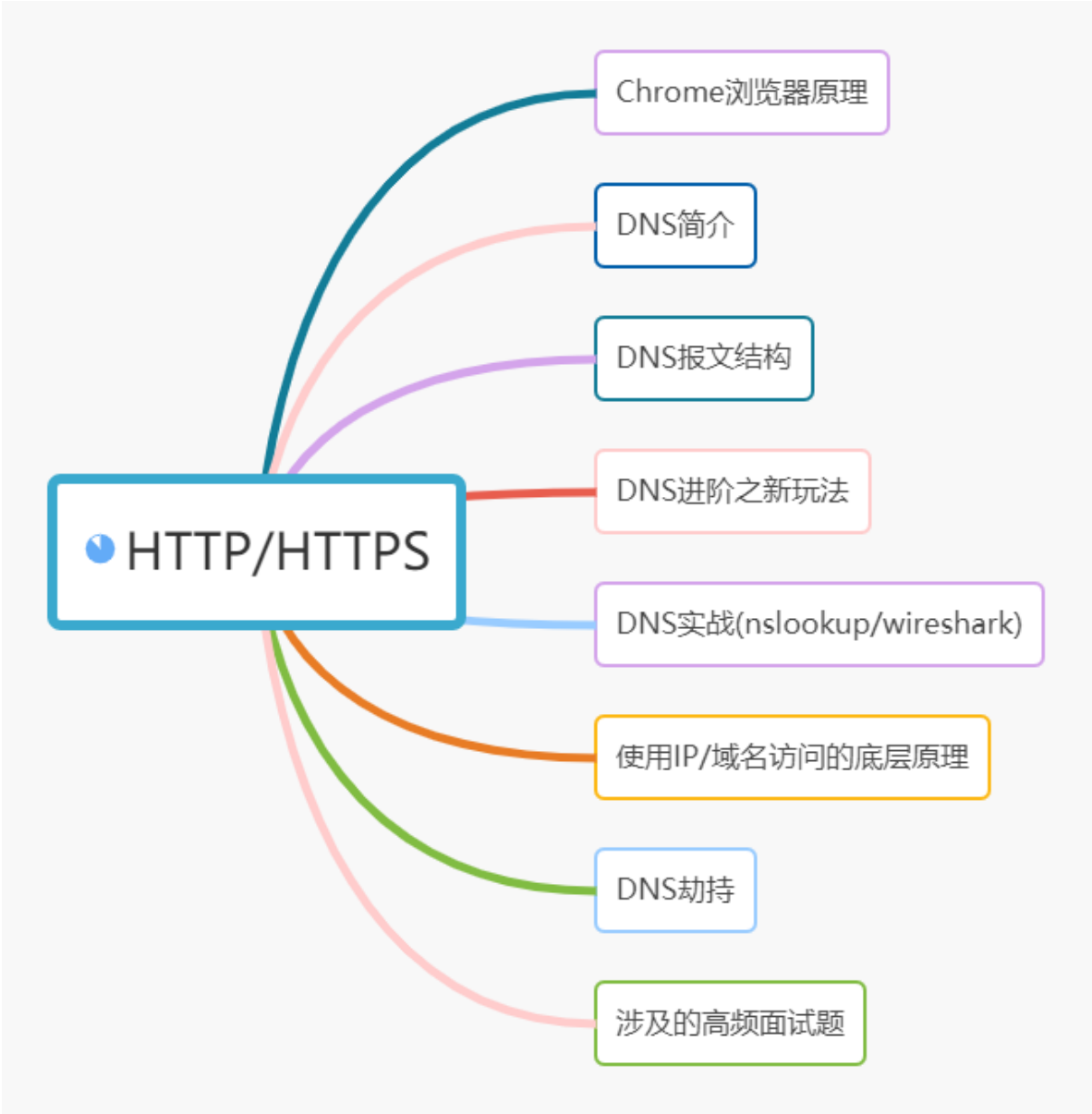
生活娱乐

高频面试相关



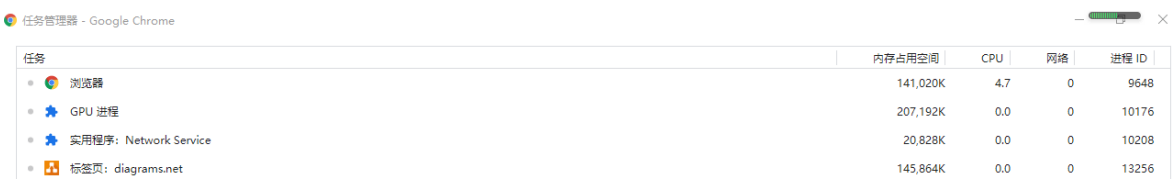
> 提到网络，基本上都能把DNS给扯上去。为啥呢，今天我们来一探究竟。

1 Chrome浏览器原理

> 还记得面试过程中被问了千百遍的"输入URL后发生了什么"这个经典问题吗，因为这个问题覆盖了太多的知识点，其中包括计算机网络，操作系统，数据结构等一些列问题，对于面试官和面试者来说都更方便后续面试的进展。想必很多小伙伴都做过web开发，或多或少都会和各种浏览器联系在一起，最终做测试的时候也会s使用多种浏览器测试保证能有更好的兼容性。那么现在我们先从Chrome浏览器说起。

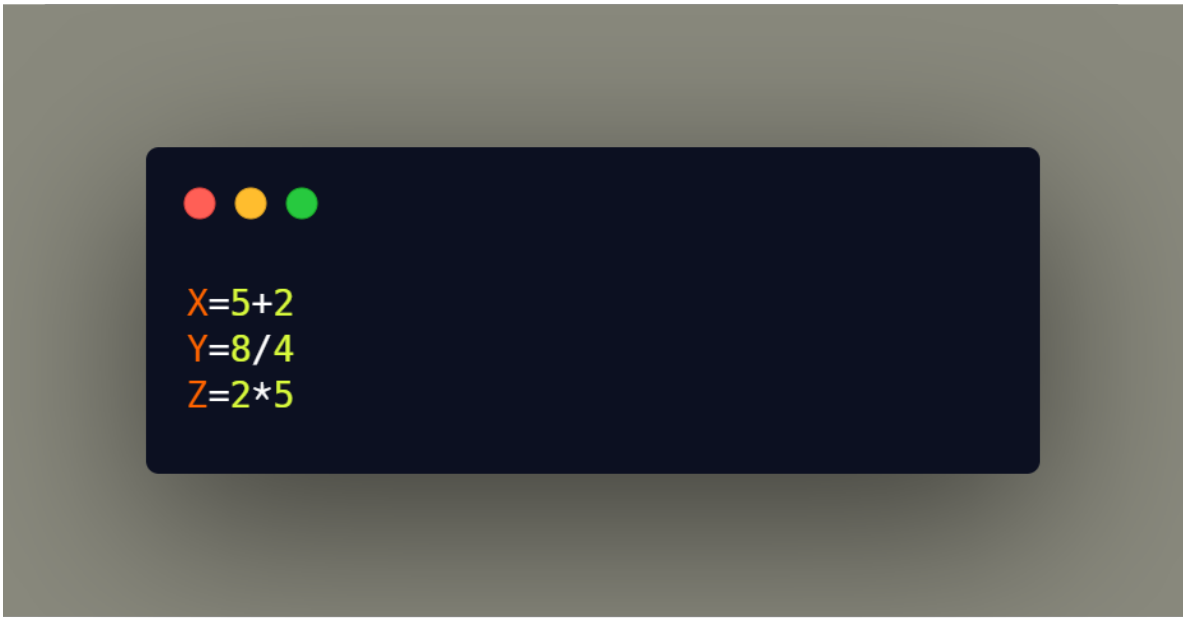
我们先想想一个问题，我们打开一个微信或者一个XX音乐，一个网页，到底会开几个进程。

我们实验看看，打开一个网页到底开了几个进程，又分别有什么作用



从上图我们发现，打开一个网页，使用了四个进程，分别为GPU进程，Network Service进程，当前网页进程和浏览器。我们先复习进程与线程。

假设现在有这样几行伪代码，我们看看应该怎么去执行，可能分为四步



- 计算X=5+2
- 计算y=8/4
- 计算z=2*5
- 显示出最后的结果

这也是采用串行的方式运行，也可说为单线程方式执行了四个任务，其好处是不用考虑诸如多线程的同步等问题。但是如果采用多线程

- 启动三个线程分别处理前面三个任务
- 最后一个线程显示结果

> 从上面这个小实验，我们可以知道使用多线程只需要两步就完成，但是单线程却使用了四步，可知使用多线程大大的提升了性能，记住：并不是多线程就一定会比单线程好，还需要从cpu使用率，IO磁盘等多个因素考虑。

进程

> 进程是是一个程序的运行实体，在上面我们比较直观的感受到了多线程并行处理提高性能的优点。一个进程可以包含多个线程，但是一个线程只能归属于一个进程，那么一个进程到底是什么样子呢(ps 下面是在Linux中执行的代码，道理差不多)

创建进程

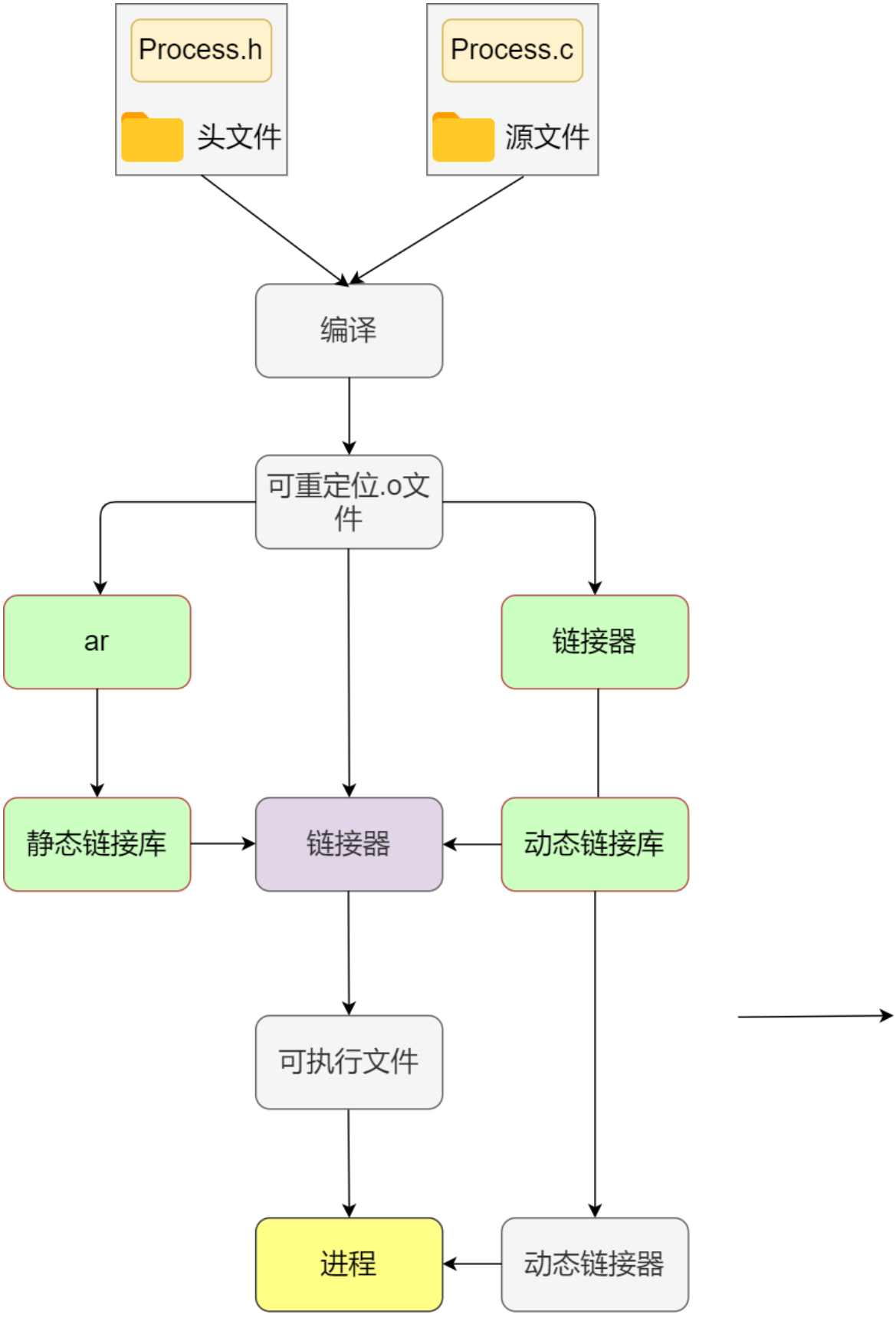
> 在Linux中使用fork创建进程，返回进程id。通过id的不同让父子进程各干其事，然后使用execvp执行具体任务

```c
#include <stdio.h>
#includue <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

extern int Create_Process(char* shell, char** P_list )

int create_process (char* shell, char** P_list)
{
    pid_t child_pid;//
    child_pid = fork ();//创建进程
    if (child_pid != 0)
        return child_pid;
    else {
        execvp (shell, P_list);//执行命令
        abort ();
    }
}
```

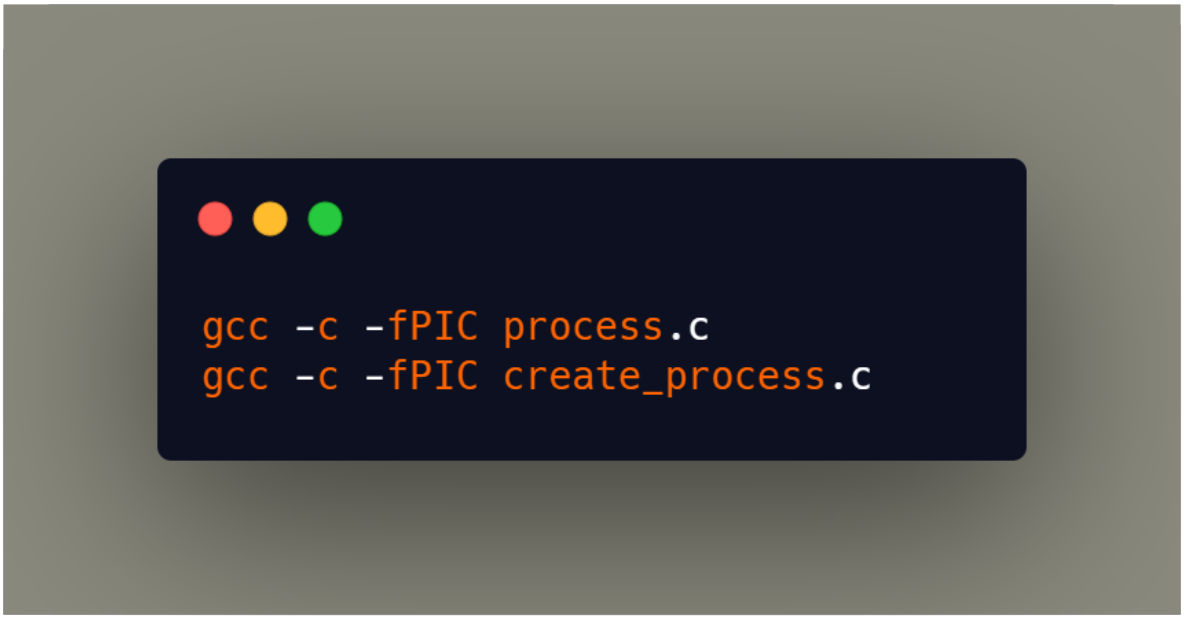
> 创建主函数来使用上面的函数，看看会出现什么情况

```c
#include <stdio.h>
#includue <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

extern int create_process (char* shell, char** P_list);
int main ()
{
    char* P_list[] = {
    "ls",
    "-l",
    "/proc/",
    NULL
    };
    create_process ("ls", P_list);
    return 0;
}
```

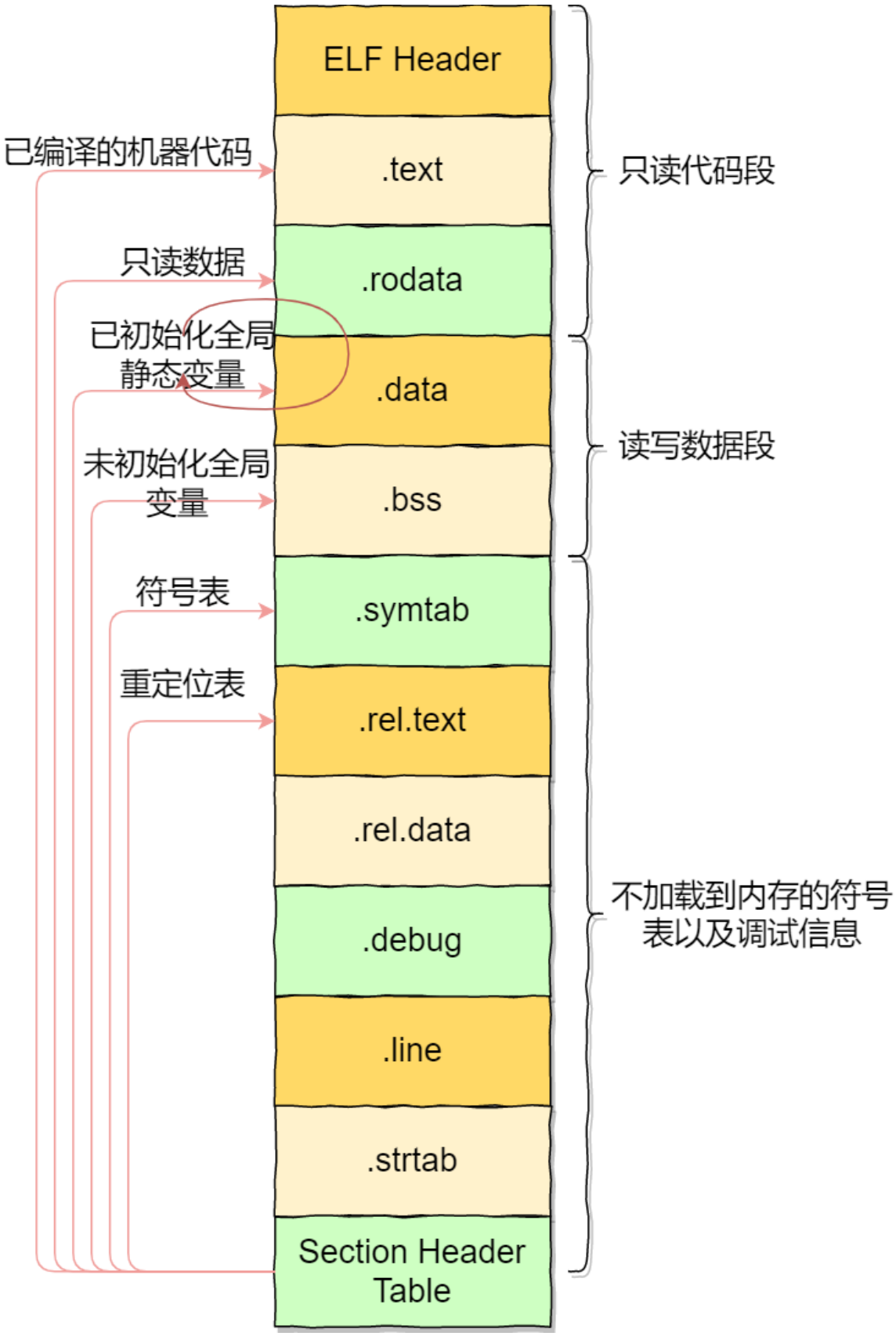
好了，现在主函数和执行函数都写完了，但是这还只是文本文件，对于计算机而言只喜欢"01"组合，cpu执行的命令需要是二进制，所以需要进行「编译」，但是二进制也得有一定的格式，不然定会乱套，在Linux中这种格式是"ELF"Executeableand Linkable Format）。其具体的样子如下所示



现在编译两个程序



> 在编译的过程中，第一步预处理，将头文件直接嵌入到文件正文中，将定义的相关宏展开，最终编译为.o文件(可重定文件)，那么ELF是什么样子呢



上图给大家准备了几个高频面试题目(哪些在代码段，数据段。。)

那么在Linux中如何查看呢(readelf工具即可)

可重定位什么意思呢?

> 字面意思是可以随时放在其他位置。对的，目前我们只是编译了文件，将来会被加载到内存里面，也就是加在某一个位置。可惜现在还是.o文件(代码片段)，不具备可执行的权限，它以后想变为函数库，哪里需要就在哪里去完成任务，搬到了哪里就重新定位了位置。要让它可重用，就得成为库文件，这个文件分为静态链接库(.a)和动态链接库，它能将一系列的.o文件归档为文件。怎

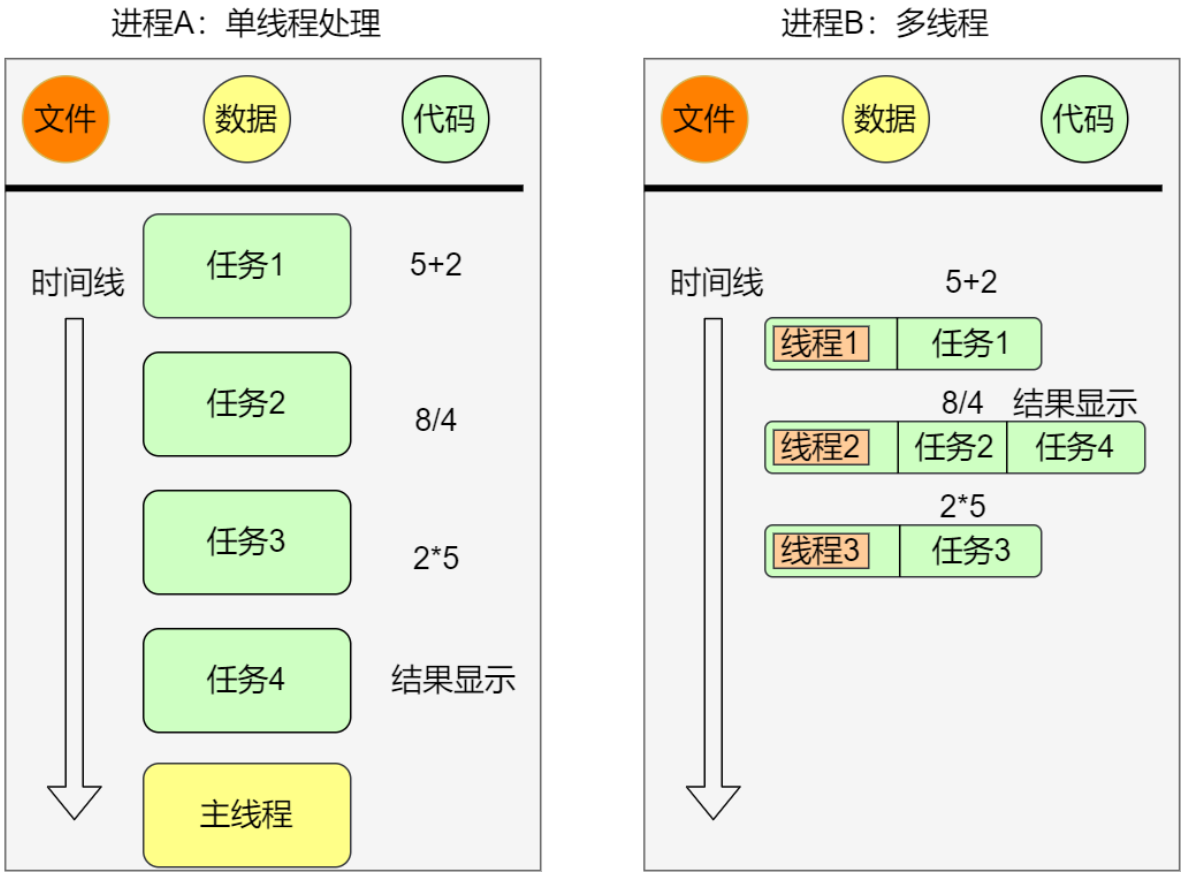
> 么创建呢

这个时候其他开发人员准备使用这个功能，加上参数连接过去就好了

> 上面命令中"-L"代表默认在当前目录寻找.a文件，然后取出.o文件和creteprocess.o做连接形成二进制执行文件 staticcreateprocess。

一旦静态链接库连接出去，它的代码和变量的section合并，一次程序运行不再依赖这个库。这就出问题，如果同样的代码段被多个程序使用，就会导致在内存中出现多份的情况，而且代码一旦更新，二进制文件也需要重新编译才能及时的更新。所以出现了动态链接库(10)

说的有点远了，回来回来。刚才我们说了多线程并行计算的优势，画个图对比加深印象下



ok总结下进程线程有哪些特点(面试跑不脱)

- 进程中的任意一个线程出错，将导致整个进程崩溃

> 假设将之前的伪代码修改为
> X=5+2
> Y=8/0
> Z=5*2

此时Y很明显就是错的，当线程执行到Y的时候就会报错，进程崩溃大致其他两个线程也没有结果

- 当一个进程关闭后，操作系统会回收进程占用的资源

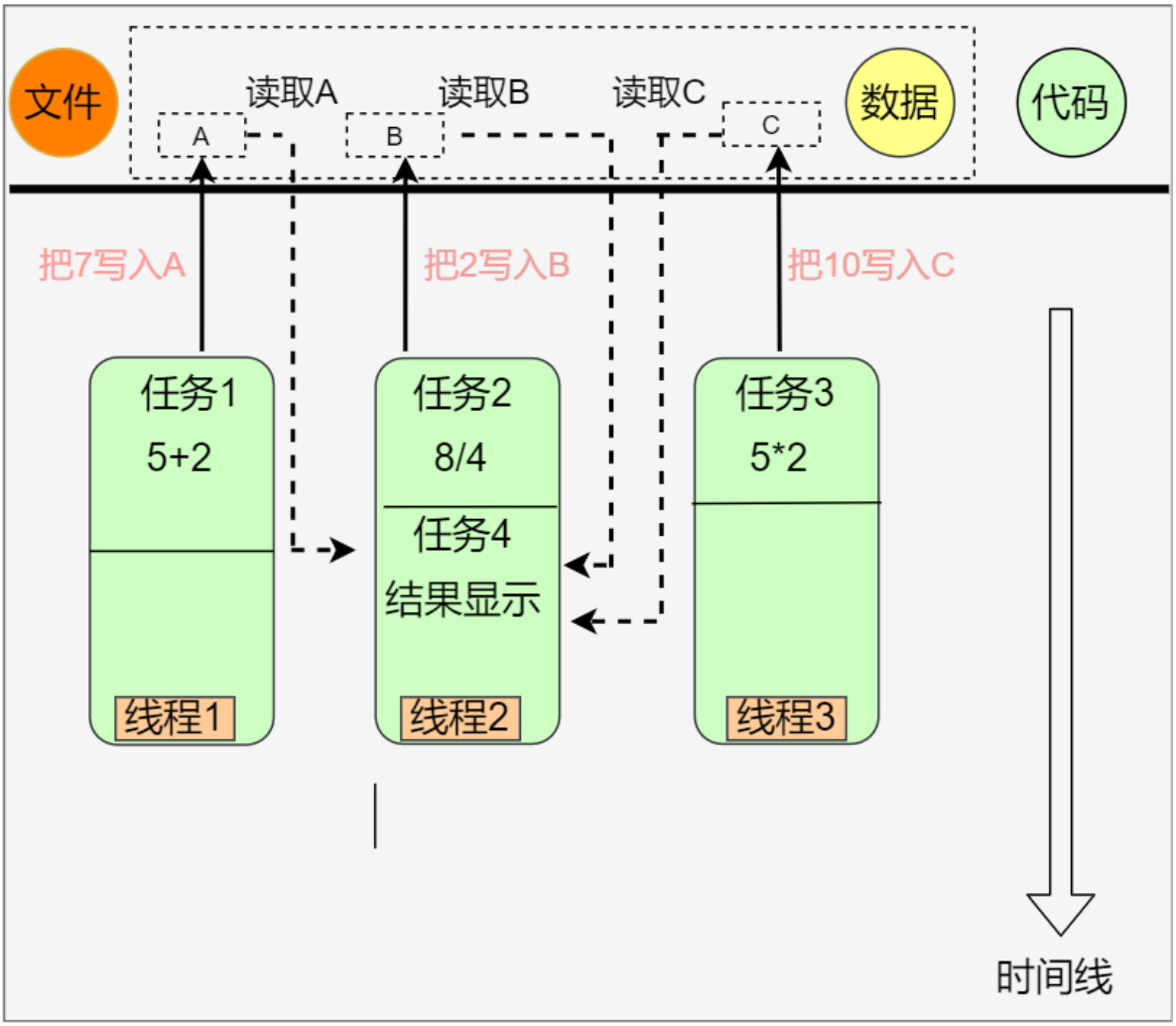
> 比如我们会使用很多不错的Chrome插件，当启动浏览器并打开这些插件的时候，都会占用内存，当关闭进程Chrome浏览器，这些内存就会被收回

- 进程之间内容相互隔离

> 这个机制是防止多个进程读写混乱，所以进程之间通信需要IPC(消息队列，共享内存等)。

- 线程之间共享进程数据

进程B：多线程

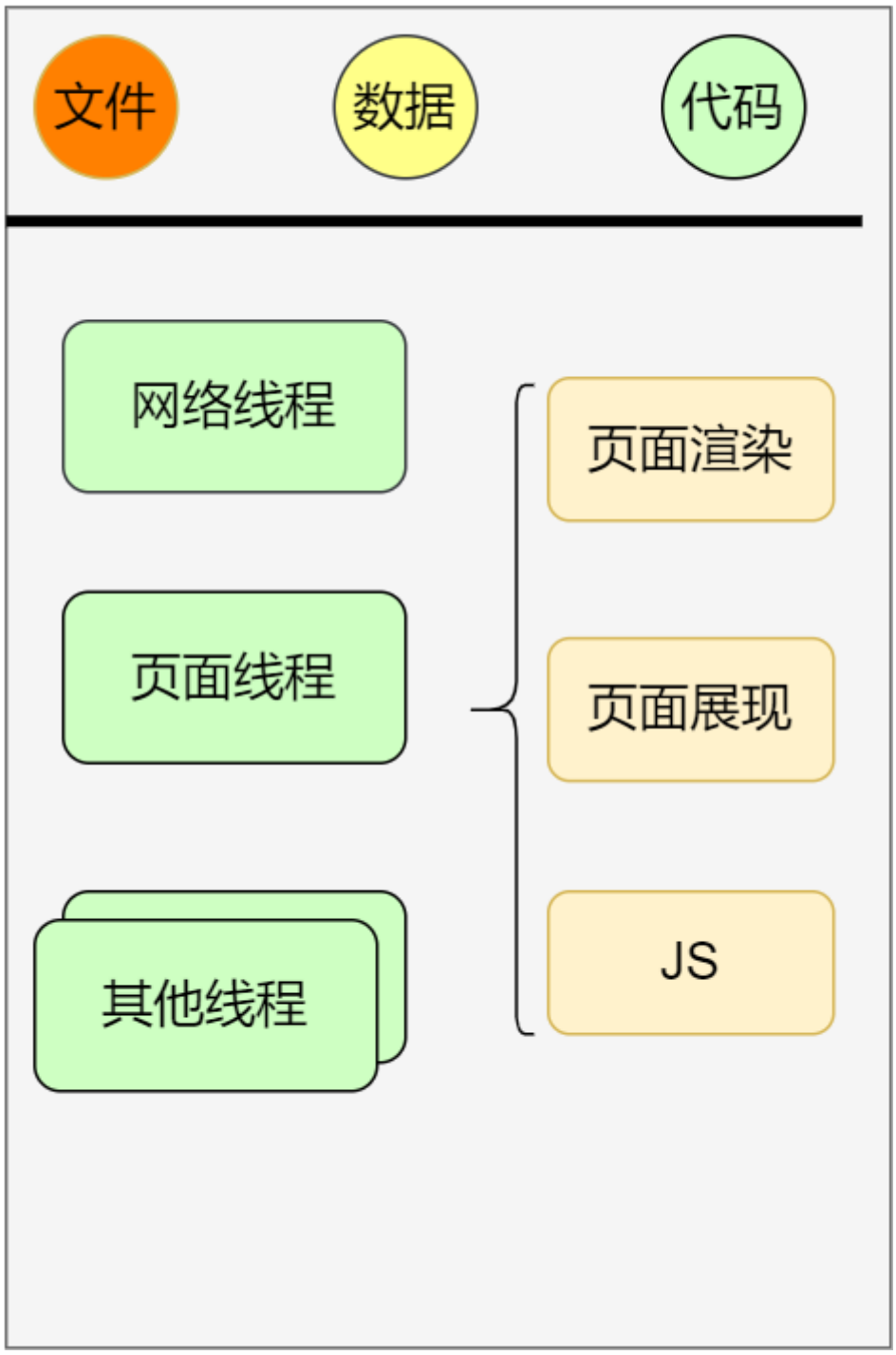


从上图我们可以知道线程1，2和线程3分别将数据写入ABC，线程2在负责处理ABC三种读取数据并显示

现在我们基本上了解了线程和进程。我们详细想象，某宝级别的系统架构一开始就能抵抗这么大的流量吗，当然不是，最开始小黄页的单体架构，随着需求的复杂和多样化主键演变而来。那么浏览器依然如此。我们看看最开始的Chrome单进程样子。

最初的浏览器单进程，意味着无论是网络，页面渲染引擎还是js环境都在一个进程中，如下图所示。

单进程浏览器



那个时候单体结构都有什么问题?

- 不稳定/不流畅

> 以前页面中的视频等元素需要使用插件才能观看，插件在页面进程中，插件出问题很容易导致浏览器崩溃。页面中如此多模块都运行在该线程中，一旦其中一个模块独占线程，其他的就只能当观众(ps 能不能完成了就走，别蹲着不X)，所以也就出现卡顿现象

- 安全性很难保障

> 当时很多插件能够比较轻松的拿到操作系统的shell，如果是页面脚本，可以通过浏览器爆出的漏铜来到shell，拿了shell就无法想象能干啥了

如何解决上述问题

- 不稳定和不流畅

> 原因是页面模块都在一个进程，采用进程分离，这样即使某个插件崩溃也只是影响某一部分，不会导致整个浏览器挂。

- 安全性问题

> 使用一个箱子(安全沙箱)，箱子里面程序可以运行且把箱子上锁，但是无法读取外部任何程序。这样的话，我把容易出错且关键的两个进程插件进程与喧嚷进程装进去，这样的话，及时两者之一执行恶意程序也只是在这个箱子里瞎摆弄，无法翻越出去拿到更高的权限干坏事。

当前架构

> 我们最初的时候，发现使用chrome浏览器打开一个网页的有四个进程，下面我来看看这些都有什么功能

从上图我们发现一共是四个进程，分别为网络进程，GUP进程，渲染进程和浏览器主进程。

网络进程

> 作为一个单独进程，负责页面网络资源的加载。

插件进程

> 由于插件容易崩溃，单独进程对其进行管理

GPU进程

> Chrome中UI界面绘制和3DCSS等需要GPU计算密集性的帮助，从而引入GPU进程

浏览器进程

> 浏览器进程负责用户交互，各个子进程等功能

乍一看全是优点，通常事物都会有两面性，进程多了，开销当然也大也就是更高的资源占用和更加复杂的体系结构。

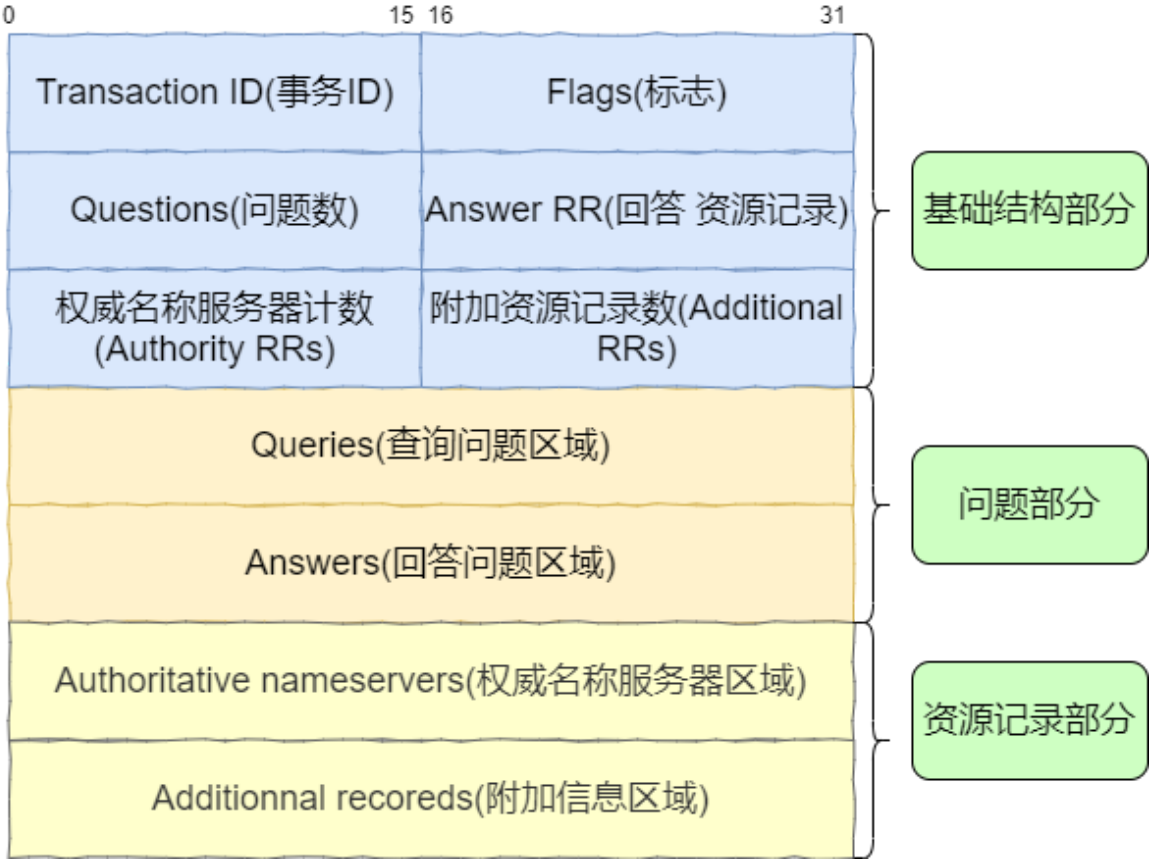
2 DNS简介

> 上面之所以介绍浏览器，因为DNS很多时候是我们在浏览器敲下回车时开始兴奋，这也是为什么从浏览器说起的原因。现在我们看看DNS到底是个啥玩意

mac地址诞生，可是太不容易记忆了，出现了简化了IP形式，它被直接暴露给外网不说，还让人类还是觉得比较麻烦，干脆用几个字母算了，也就是域名了。域名不仅仅能够代替IP，还有很多其他的用途比如在web应用中用来标识虚拟主机。

3 DNS报文结构

> 说了这么多，协议头部，到底有哪些字段，其含义是什么都还不知道，那怎么去分析报文，下面我们一起再看看报文什么样子



基础结构部分

> DNS报文基础部分为DNS首部。其中包含了事务ID，标志，问题计数，回答资源计数，回答计数，权威名称服务器计数和附加资源记录数。

- 事务ID:报文标识，用来区分DNS应答报文是对哪个请求进行响应
- 标志:DNS报文中标志字段
- 问题计数：DNS查询请求了多少次

- 回答资源记录数：DNS响应了多少次
- 权威名称服务器计数: 权威名称服务器数目
- 附加资源记录数: 权威名称服务器对应IP地址的数目

重点！！！！基础结构中的标志字段细分如下：

QR	Opcode	AA	TC	RD	RA	Z	rcode

- QR(Response)：查询请求，值为0；响应为1
- Opcode:操作码。0表示标准查询；1表示反向查询；2服务器状态请求
- AA（Authoritative）：授权应答，该字段在响应报文中有效。通过0,1区分是否为权威服务器。如果值为 1 时，表示名称服务器是权威服务器；值为 0 时，表示不是权威服务器。
- TC（Truncated）：表示是否被截断。当值为1的时候时，说明响应超过了 512字节并已被截断，此时只返回前512个字节。
- RD（Recursion Desired）：期望递归。该字段能在一个查询中设置，并在响应中返回。该标志告诉名称服务器必须处理这个查询，这种方式被称为一个递归查询。如果该位为 0，且被请求的名称服务器没有一个授权回答，它将返回一个能解答该查询的其他名称服务器列表。这种方式被称为迭代查询。
- RA（Recursion Available）：可用递归。该字段只出现在响应报文中。当值为 1 时，表示服务器支持递归查询。
- Z：保留字段，在所有的请求和应答报文中，它的值必须为 0。
- rcode（Reply code）：通过返回只判断相应的状态。

> 当值为0时，表示没有错误；当值为1时，表示报文格式错误（Format error），服务器不能理解请求的报文；当值为 2 时，表示域名服务器失败（Server failure），因为服务器的原因导致没办法处理这个请求；当值为 3 时，表示名字错误（Name Error），只有对授权域名解析服务器有意义，指出解析的域名不存在；当值为 4 时，表示查询类型不支持（Not Implemented），即域名服务器不支持查询类型；当值为 5 时，表示拒绝（Refused），一般是服务器由于设置的策略拒绝给出应答，如服务器不希望对某些请求者给出应答。

问题部分

> 该部分是用来显示DNS查询请求的问题，其中包含正在进行的查询信息，包含查询名（被查询主机名字）、查询类型、查询类。

- 查询名:一般为查询的域名，也可能是通过IP地址进行反向查询
- 查询类型：查询请求的资源类型。常见的如果为A类型，表示通过域名获取IP。具体参数如下图
- 查询类：地址类型，通常为互联网地址为1

资源记录部分

> 资源记录部分包含回答问题区域，权威名称服务器区域字段、附加信息区域字段，格式如下

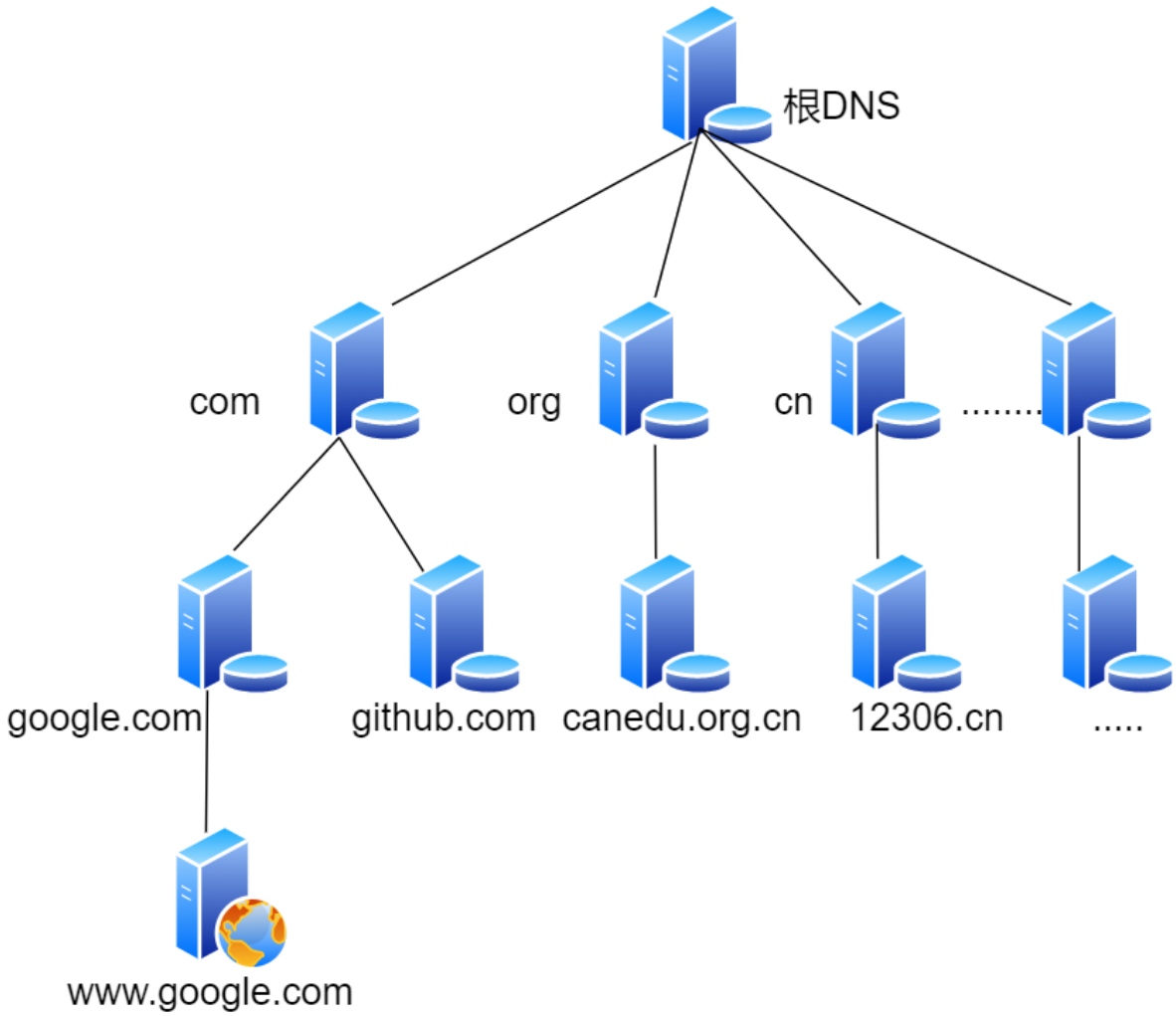
- 域名：所请求的域名
- 类型：与问题部分查询类型值一直
- 类：地址类型，和问题部分查询类值一样
- 生存时间：以秒为单位，表示资源记录的生命周期
- 资源数据长度：资源数据的长度
- 资源数据：按照查询要求返回的相关资源数据

4 DNS解析详解

> 知道了DNS大概是什么，它的域名结构和报文结构，是时候看看到底怎么解析的以及如何保证域名的解析比较稳定和可靠

DNS核心系统

- 根域名服务器(Root DNS Server),大哥，管理顶级域名服务并放回顶级域名服务器IP，比如"com","cn"
- 顶级域名服务器(Top-level DNS Server),每个顶级域名服务器管理各自下属，比如com可以返回baidu.com域名服务器的IP
- 权威域名服务器(Authoritative DNS Server),管理当前域名下的IP地址，比如Tencent.com可以返回www.tencent.com的IP地址



举个例子，假设我们访问"www.google.com"

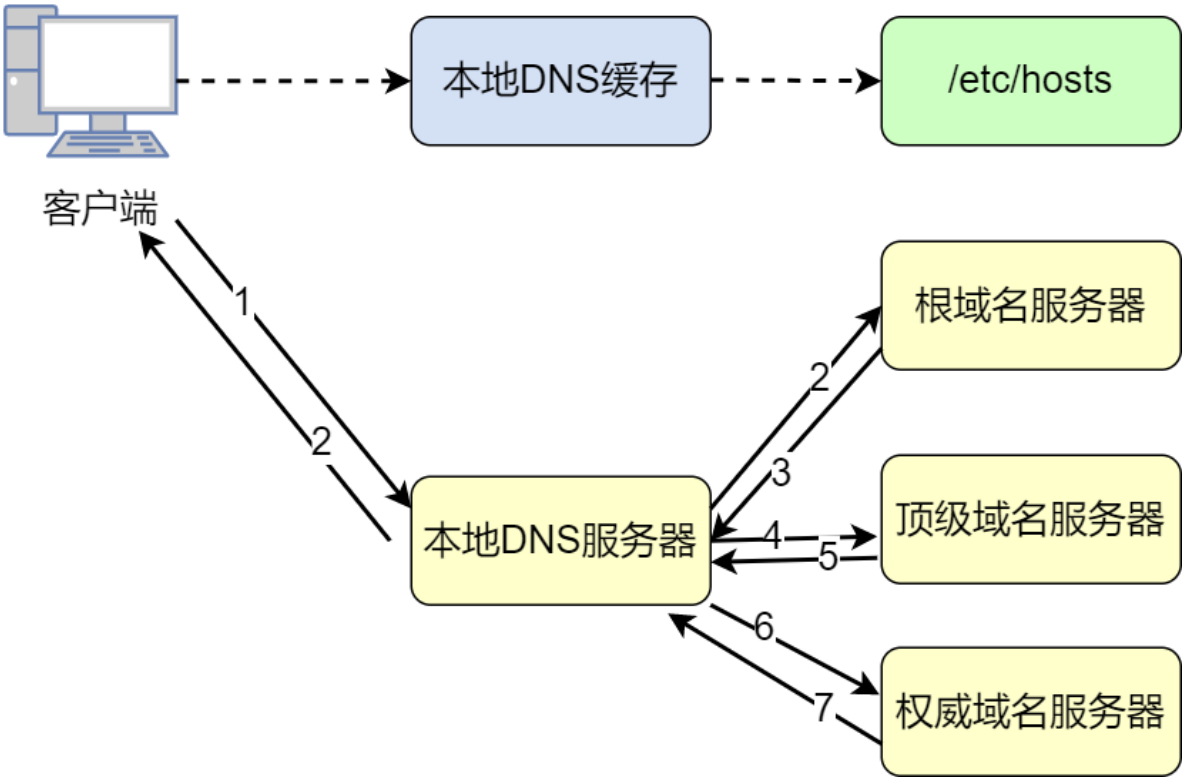
- 访问根域名服务器，这样我们就会知道"com"顶级域名的地址
- 访问"com"顶级域名服务器，可知道"google.com"域名服务器的地址
- 最后方位"google.com"域名服务器，就可知道"www.google.com"的IP地址

嘿嘿，目前全世界13组根域名服务器还有上百太镜像，但是为了让它能力更强，处理任务效率更高，尽量减少域名解析的压力，通常会加一层"缓存"，意思是如果访问过了，就缓存，下一次再访问就直接取出，也就是咱么经常配置的"8.8.8.8"等

操作系统中同样也对DND解析做缓存，比如说曾访问过"www.google.com"，

其次，还有我们熟知的hosts文件，当在操作系统中没有命中则会在hosts中寻找。

这样依赖，相当于有了DNS服务器，操作系统的缓存和hosts文件，能就近(缓存)完成解析就好，不用每次都跑到很远的地方去解析，这样大大减轻的DNS服务器的压力。画了一个图，加深印象



嗯？想必应该知道这个过程了，我们再举个例子，假设我们访问www.qq.com

- 客户端发送一个DNS请求，请问qq你的IP的什么啊，同时会在本地域名服务器(一般是网络服务是临近机房)打声招呼
- 本地收到请求以后，服务器会有个域名与IP的映射表。如果存在，则会告诉你，如果想访问qq，那么你就访问XX地址。不存在则会去问上级(根域服务器):"老铁，你能告诉我www.qq.com"的IP么
- 根DNS收到本地DNS请求后，发现是.com，"www.qq.com哟，这个由.com大哥管理，我马上给你它的顶级域名地址，你去问问它就好了"
- 这个时候，本地DNS跑去问顶级域名服务器，"老哥，能告诉下www.qq.com"的ip地址码",这些顶级域名负责二级域名比如qq.com
- 顶级域名回复："小本本记好，我给你www.qq.com区域的权威DNS服务器地址"，它会告诉你
- 本地DNS问权威DNS服务器："兄弟，能不能告诉我www.qq.com对应IP是啥"
- 权威DNS服务器查询后将响应的IP地址告诉了本地DNS，本地服务器将IP地址返回给客户端，从而建立连接。

5 DNS进阶之新玩法

> 这里主要分享DNS(GSLB)的全局负载均衡。不是所有的互联网服务都适用于GSLB。

全局负载均衡采用的主要技术是智能DNS，它综合多种不同的策略(比如根据地理位置或者根据繁忙程度的权重)将客户访问的域名解析到不同的线路上。开启介绍之前，再一次复习下DNS中A记录和NS记录

- A记录

> A记录是名称解析的重要记录，它用于将特定的主机名映射到对应主机的IP地址上。你可以在DNS服务器中手动创建或通过DNS客户端动态更新来创建

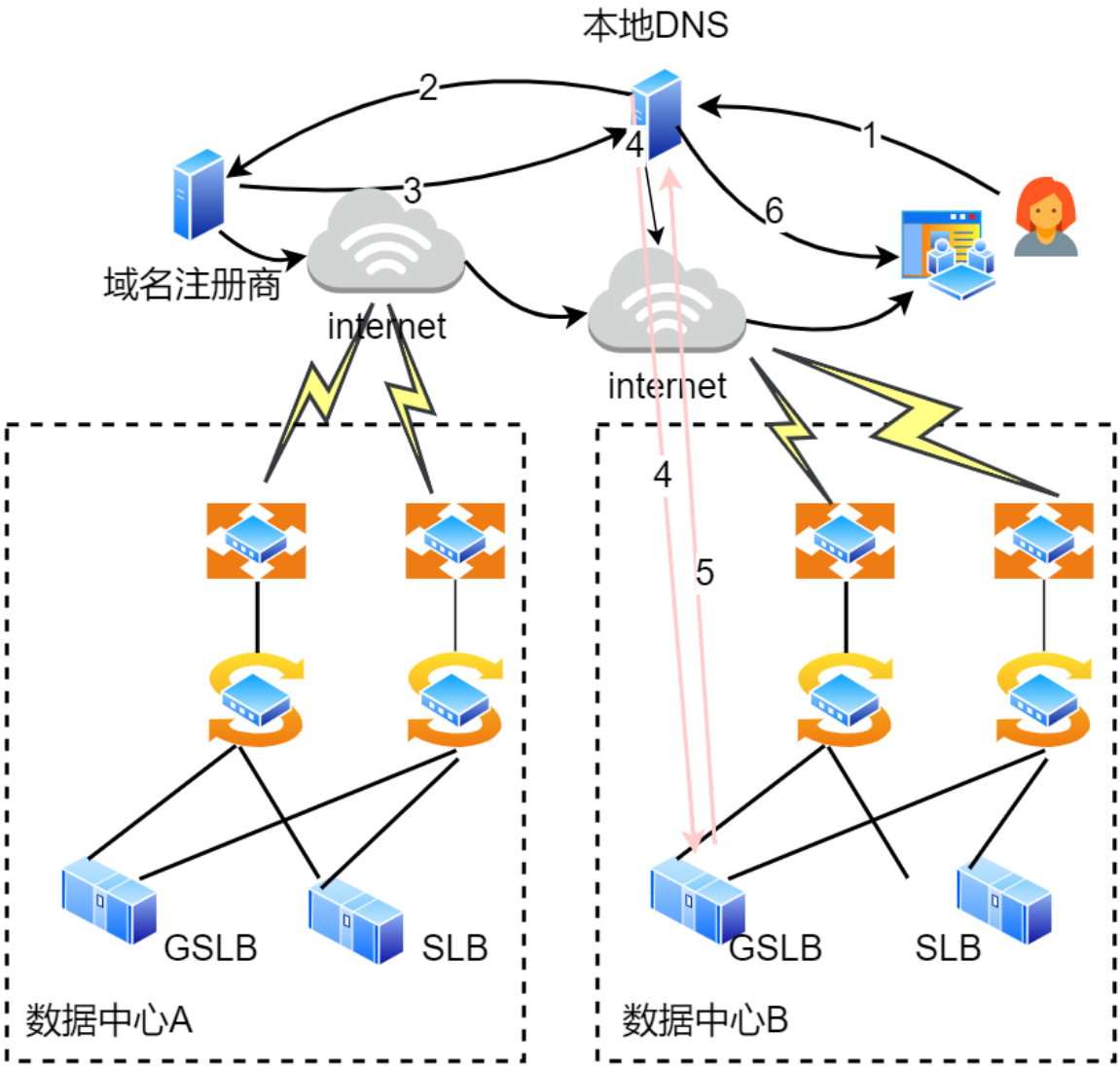
- NS记录

> NS记录此记录指定负责此DNS区域的权威名称服务器。

- 两者区别

> A记录直接给出目的IP，NS记录将DNS解析任务交给特定的服务器，NS记录中记录的IP即为该特定服务器的IP地址

在全局负载均衡解决方案中，NS记录指向具有智能DNS解析功能的GSLB设备，通过GSLB设备进行A记录解析。为了保证高可用，如果为多地部署GSLB，则均配置记录。另外，GSLB设备还会对所在的后端服务器公网IP进行健康检查，其结果通过自有协议在不同的的GLSB设备间同步。解析的步骤如下图



- 用户给本地DNS服务器发送查询请求，如果本地有缓存直接返回给用户，否则通过递归查询获得名服务商商处的授权DNS服务器
- 授权服务器返回NS记录给本地DNS服务器。其中NS记录指向一个GSLB设备接口地址
- GSLB设备决策最优解析结果并返回A记录给本地DNS服务器。
- 本地服务器将查询结果通过一条A记录返回给用户，并缓存这条记录。

6 DNS实战(wireshark)

> 使用工具为wireshark，访问www.baidu.com

```
32.2193860 192.168.1.104    60.255.80.18     DNS      77 Standard query 0xdfd2  A dss0.bdstatic.com
32.2195330 192.168.1.104    60.255.80.18     DNS      73 Standard query 0x0574  A sp0.baidu.com
32.2196500 192.168.1.104    60.255.80.18     DNS      77 Standard query 0xf3c2  A dss1.bdstatic.com
32.2414520 60.255.80.18     192.168.1.104    DNS     126 Standard query response 0xdfd2  CNAME sslbaiduv6.jomodns.com A
32.2414530 60.255.80.18     192.168.1.104    DNS     132 Standard query response 0x0574  CNAME www.a.shifen.com A 182.6
32.2414530 60.255.80.18     192.168.1.104    DNS     126 Standard query response 0xf3c2  CNAME sslbaiduv6.jomodns.com A
32.2580010 192.168.1.104    60.255.80.18     DNS      73 Standard query 0xfc1e  A sp1.baidu.com
32.2592690 192.168.1.104    60.255.80.18     DNS      76 Standard query 0x0568  A ss1.bdstatic.com
32.2592740 192.168.1.104    60.255.80.18     DNS      73 Standard query 0x70a7  A sp2.baidu.com
```

> 分析DNS请求帧，如下图所示

```
Domain Name System (query)
    [Response In: 311]
    Transaction ID: 0xdfd2
  Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  Queries
    dss0.bdstatic.com: type A, class IN
        Name: dss0.bdstatic.com
        Type: A (Host address)
        Class: IN (0x0001)
```

标准递归查询

请求的域名：dss0.bdstatic.com
Type:A(主机地址)
地址类型:IN(互联网地址)

从上图我们可知道请求计数为1，请求的域名为dss0.bdstatic.com

> 分析DNS响应帧

```
Domain Name System (response)
    [Request In: 308]
    [Time: 0.022066000 seconds]
    Transaction ID: 0xdfd2
  Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authen
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 0
```

响应段，没有错误且域名服务器支持递归查询。问题计数为1，回答计数为2

从响应头可以知道，问题计数为1，正好对应请求帧1个问题。回应了2个问题。分别为

```
Answers
  dss0.bdstatic.com: type CNAME, class IN, cname sslbaiduv6.jomodns.com
      Name: dss0.bdstatic.com
      Type: CNAME (Canonical name for an alias)
      Class: IN (0x0001)
      Time to live: 1 hour, 10 minutes, 36 seconds
      Data length: 21
      Primaryname: sslbaiduv6.jomodns.com
  sslbaiduv6.jomodns.com: type A, class IN, addr 60.255.149.33
      Name: sslbaiduv6.jomodns.com
      Type: A (Host address)
      Class: IN (0x0001)
      Time to live: 45 seconds
      Data length: 4
      Addr: 60.255.149.33 (60.255.149.33)
```

回答1：CNAME别名为:sslbaiduv6.jomodns.com

回答2：类型为主机地址：ip为60.255.217.109

```
Authoritative nameservers
  com: type NS, class IN, ns a
  com: type NS, class IN, ns g
  com: type NS, class IN, ns i
  com: type NS, class IN, ns l
  com: type NS, class IN, ns b
  com: type NS, class IN, ns j
  com: type NS, class IN, ns k
  com: type NS, class IN, ns h
  com: type NS, class IN, ns e
  com: type NS, class IN, ns c
  com: type NS, class IN, ns f
  com: type NS, class IN, ns m
  com: type NS, class IN, ns d
Additional records
  a.gtld-servers.net: type A,
```

从上图可以得出当前共有13个权威域名服务器，当然每一个的服务器地址不同，其中类型为NS代表权威域名服务器服务器

两个相似面试题

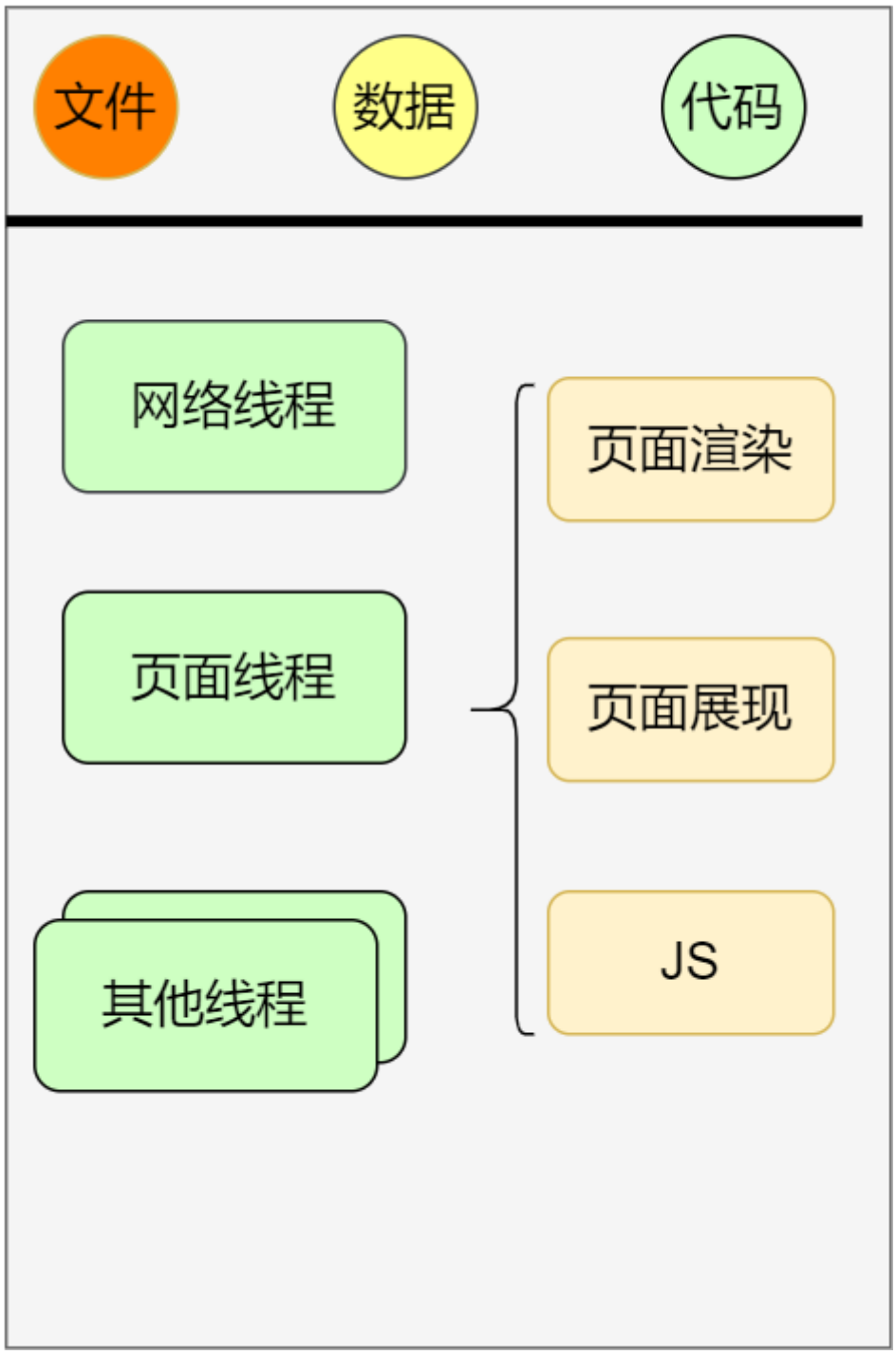
7 使用IP地址访问浏览器的原理

- 打开chrome浏览器，输入IP
- 三次握手建立连接
- 建立连接以后HTTP开始工作，通过TCP发送一个"GET / HTTP/1.1",服务端给予回应
- 解析请求，根据HTTP协议规定解析，看看那浏览器想干啥
- 哦，原来你想获取我的视频呀，那我读出来拼接为HTTP格式给你，回复"HTTP/1.1 200 OK"
- 作为浏览器回复一个TCP的ACK表示确认
- 浏览器收到响应数据后，需要使用相应的引擎进行渲染，将更好的页面展现给用户

8 使用域名访问浏览器的原理

> 这一次从浏览器角度回答，相信大家已经了解一部分浏览器知识了，我们先看看URL到网页展示的完整流程是什么样子

单进程浏览器



- 用户输入

> 在地址栏输入相应的内容，如果为关键字，如果直接输入搜索内容，浏览器默认引擎会合成为URL,如果符合URL规则，加上协议合成完整URL,回车就会出现加载页面，也就是等待提交文档的阶段

- URL请求过程

> 此时浏览器进程将URL通过进程间通信的方式发送给网络进程，开启真正的请求流程。注意了，网络进程这里也有缓存，它会现在本地缓存查看是否缓存了资源，如果有则直接返回。如果没有，那就需要DNS解析获取服务器IP地址(HTTPS还少不了TLS连接)

此时使用IP和服务器建立三次握手。连接成功开始构造请求头等信息。

服务器收到请求信，根据请求信息生成响应信息给网络进程。然后开始解析响应头内容。如果返回值为302/301,说明需要跳转到其他URL,如果为200则继续处理该请求。

> URL的请求数据类型多种，对于浏览器而言是怎么区分的呢

这个时候就必须强调下Content-type了，因为他明确服务器返回响应体数据属于什么类型，此时的浏览器也会根据Content-type对决定响应体是什么内容

- 进入渲染阶段

> 通常情况下，当前多进程架构的浏览器对于每一个页面都有一个渲染进程，前提是如果从X页面打开Y页面，x和y属于同一个"站点"(使用相同的协议和根域名)，此时y页面会复用x页面，否则y页面会单独对应一个渲染进程。

- 提交阶段

> 渲染进程收到浏览器进程的"提交文档"后，通过和网络进程使用"管道"的方式通信。一旦这些文档数据传输完成，渲染进程就会告诉浏览器进程"确认提交"，此时浏览器进程收到"确认提交"就会更新地址栏的URL，历史状态等，这就是为什么当我们在地址栏输入地址信息后需要加载一小会儿到另一个页面。over

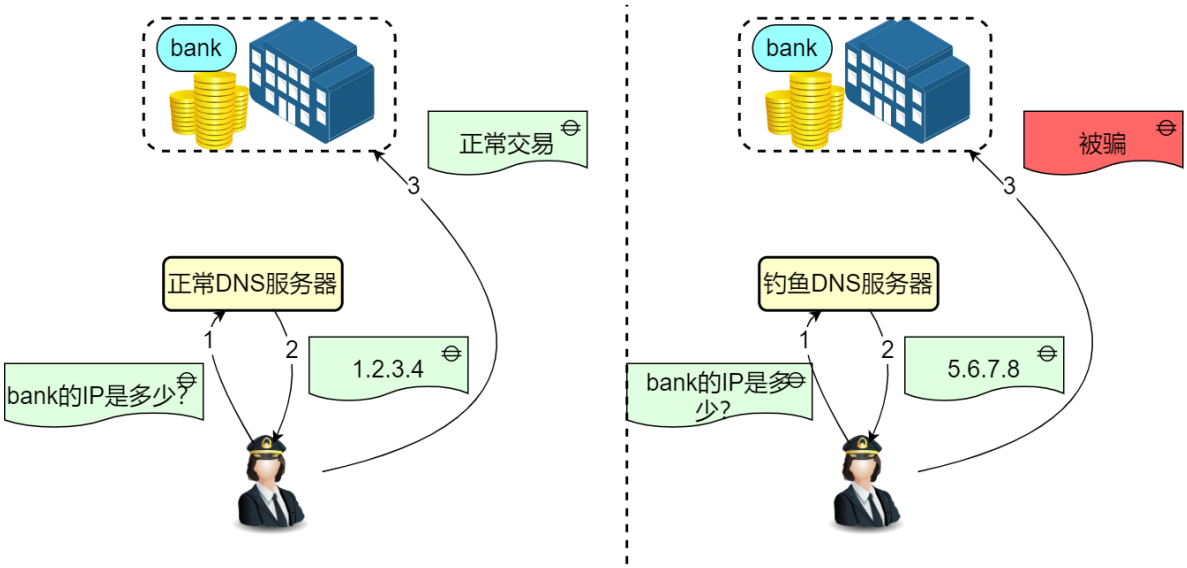
- 渲染阶段

> 文档提交以后，此时就需要使用js，css等美化页面，并通过构建DOM树等让用户有更好的使用体验。

9 DNS劫持

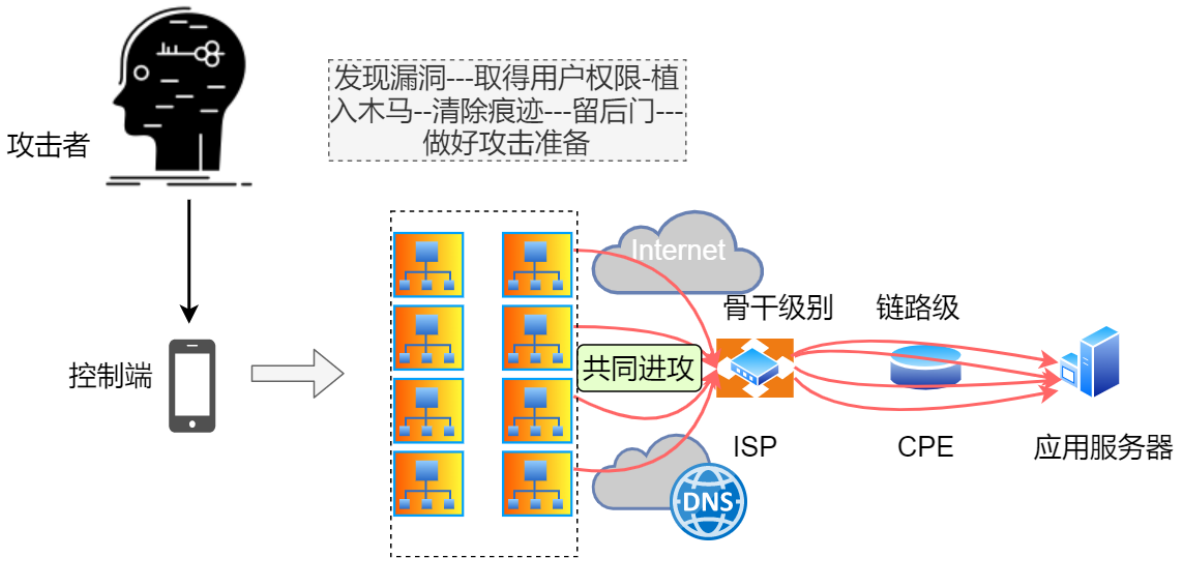
> 到这里我们至少知道了DNS可以将域名映射为IP，并且知道了使用了多种缓存方案来减少DNS访问的压力。那么DNS一旦出错，很可能将域名解析到其他IP地址，这样我们也就无法正确访问网页(PS是不是有的时候发现开启不了网页但是qq等可以使用，很可能就是DNS搞鬼了哟)

DNS劫持方法



- 利用DNS服务器进行DDOS攻击

> 什么是DDOS，我们应该直到SYN Flood，是一种DoS(拒绝服务攻击)与DDOS(分布式拒绝服务攻击的方式),利用大量的伪造TCP请求让被攻击方资源榨干。



我们假设攻击者已经知道了攻击者IP(如果需要了解这一部分内容，可以去搜索主动被动信息搜集/sodan等关键字)，此时攻击者使用此地址作为解析命令的源地址，当DNS请求的时候返回恰巧也是被攻击者。此时如果足够多的请求(群肉鸡)就很容易使网络崩溃。

- 缓存感染

> 我们已经知道了在DNS查询过程中，会经过有操作系统的缓存，hosts文件等，如果将数据放入有漏洞的服务器缓存中，当进行DNS请求的时候，就会将缓存信息返回给用户，这样用户就会莫名访问入侵者所设置的陷阱页面中。

- DNS信息劫持

> 看到这里的小伙伴，先思考一个问题，在TCP/IP协议栈中，三次握手中的序列号到底什么意思?

其功能之一就是避免伪装数据的插入。我们知道，如果我们知道DNS报文中的ID，我们就可以知道这个ID请球员位置。作为攻击者，会通过旁路监听客户端和服务端的会话，拿到DNS中的ID，此时相当于在DNS服务器之前拿到ID，伪装DNS服务器回复客户端，引导客户端访问恶意的网站

https://www.cnblogs.com/hacker520/p/12938369.html
http://www.360doc.com/content/18/0422/12/11935121_747766478.shtml

电脑小故障

> 比如qq可用但是浏览器就是不好使

- 输入：http://192.168.1.1(可能是http://192.168.0.1)，输入路由器用户名密码
- DHCP服务器-----DHCP服务-，修改DNS为更加可靠的DNS服务器IP.保存即可

方法2:修改路由器password

- 地址栏输入"http://192.168.1.1",登录并进入路由器页面
- 系统工具--修改登录口令页面

保护域名/尽量避免攻击

- 备份策略。一般至少会使用两个域名，一旦其中一个被攻击，用户可以通过另一个访问
- 随时留意域名注册中的电子邮件
- 保存好所有权信息(比如账单记录，注册信息等)
- 随时关注安全补丁

10 本文涉及高频面试题(测试)

- 讲讲DNS原理

- 进程与线程
- 递归查询和递归查询区别
- DNS解析流程
- chrome架构演变
- ELF是什么，数据段，代码段，全局变量等分别存放在哪儿
- DNS劫持
- 描述下DDOS与DOS攻击
- 使用IP地址访问web服务器
- 使用域名访问web服务器过程
- 可重定位什么意思?
- 静态库与动态库的区别
- 进程与线程间共享数据

唠嗑

今日的分享也就告一段落了，码字不易，不想被「白嫖」，文末点个「在看」吧，让我们一起「Smile」。

persist