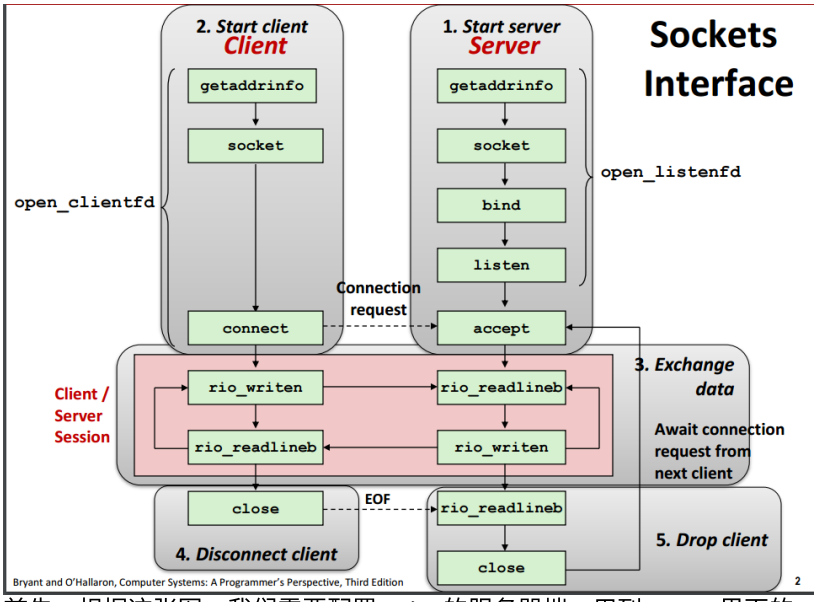
15213-ICS-Proxy

2017年9月7日 11:15

要设置代理服务器，首先我们要有一个服务器。首先我们使用ICS这本书的代码Tiny构建服务器。所有的网络通信都是基于socket的。



首先，根据这张图，我们需要配置socket的服务器端，用到csapp.c里面的Open_listenfd，需要制定服务器的端口号。然后调用Robust的Accept与客户端通信。根据HTTP原理，先到的请求肯定是HTTP请求头。我们对于这些请求建立一个缓存空间，便于对请求进行处理。从先到的请求内容里面提取出method，uri，version这些内容。
然后对uri进行解析。这里分两种情况，如果路径包含cgi-bin就是动态内容，否则是静态页面。如果请求的资源不存在，需要返回404:

```
void clienterror(int fd, char *cause, char *errnum,
                 char *shortmsg, char *longmsg)
{
    char buf[MAXLINE], body[MAXBUF];

    /* Build the HTTP response body */
    sprintf(body, "<html><title>Tiny Error</title>");
    sprintf(body, "%s<body bgcolor=""ffffff"">\r\n", body);
    sprintf(body, "%s%s: %s\r\n", body, errnum, shortmsg);
    sprintf(body, "%s<p>%s: %s\r\n", body, longmsg, cause);
    sprintf(body, "%s<hr><em>The Tiny Web server</em>\r\n", body);

    /* Print the HTTP response */
    sprintf(buf, "HTTP/1.0 %s %s\r\n", errnum, shortmsg);
    Rio_writen(fd, buf, strlen(buf));
    sprintf(buf, "Content-type: text/html\r\n");
    Rio_writen(fd, buf, strlen(buf));
    sprintf(buf, "Content-length: %d\r\n\r\n", (int)strlen(body));
    Rio_writen(fd, buf, strlen(buf));
    Rio_writen(fd, body, strlen(body));
```

然后判断对文件是否有相关权限，如果没有权限返回403。
如果是静态访问，只需要读取文件并且写入即可。

```
void serve_static(int fd, char *filename, int filesize)
{
    int srcfd;
    char *srcp, filetype[MAXLINE], buf[MAXBUF];

    /* Send response headers to client */
    get_filetype(filename, filetype);       //line:netp:servestatic:getfiletype
    sprintf(buf, "HTTP/1.0 200 OK\r\n");    //line:netp:servestatic:beginserve
    sprintf(buf, "%sServer: Tiny Web Server\r\n", buf);
    sprintf(buf, "%sConnection: close\r\n", buf);
    sprintf(buf, "%sContent-length: %d\r\n", buf, filesize);
    sprintf(buf, "%sContent-type: %s\r\n\r\n", buf, filetype);
    Rio_writen(fd, buf, strlen(buf));       //line:netp:servestatic:endserve
```

```
    printf("Response headers:\n");
    printf("%s", buf);

    /* Send response body to client */
    srcfd = Open(filename, O_RDONLY, 0);    //line:netp:servestatic:open
    srcp = Mmap(0, filesize, PROT_READ, MAP_PRIVATE, srcfd, 0);//line:netp:servestatic:mmap
    Close(srcfd);                           //line:netp:servestatic:close
    Rio_writen(fd, srcp, filesize);         //line:netp:servestatic:write
    Munmap(srcp, filesize);                 //line:netp:servestatic:munmap
}
```

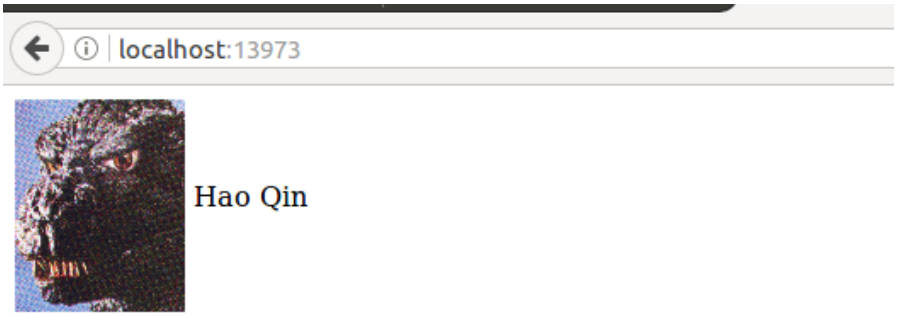
如果是动态访问，需要执行相关命令，并且重定向标准输出stdout到客户端stdout_fileno。

```
void serve_dynamic(int fd, char *filename, char *cgiargs)
{
    char buf[MAXLINE], *emptylist[] = { NULL };

    /* Return first part of HTTP response */
    sprintf(buf, "HTTP/1.0 200 OK\r\n");
    Rio_writen(fd, buf, strlen(buf));
    sprintf(buf, "Server: Tiny Web Server\r\n");
    Rio_writen(fd, buf, strlen(buf));

    if (Fork() == 0) { /* Child */ //line:netp:servedynamic:fork
        /* Real server would set all CGI vars here */
        setenv("QUERY_STRING", cgiargs, 1); //line:netp:servedynamic:setenv
        Dup2(fd, STDOUT_FILENO);         /* Redirect stdout to client */ //line:netp:servedynamic:dup2
        Execve(filename, emptylist, environ); /* Run CGI program */ //line:netp:servedynamic:execve
    }
    Wait(NULL); /* Parent waits for and reaps child */ //line:netp:servedynamic:wait
}
/* $end serve_dynamic */
```

好了，现在我们有了一个简单的服务器了，测试一下：



我们的代理服务器的作用是接收客户端的请求，然后对请求头进行各式处理，最后讲请求转发给制定服务器。
这样就很明显了，与Tiny服务器一样，我们需要安全可靠的socket通信，用到Open_listenfd。在doit函数里面，我们同样人为最开始传进去的是请求头，需要提取出method，uri，version等信息。并且对uri进行解析。然后我们需要将请求再转发给目标服务器。这样看起来这个代理服务器完全没有存在的意义。但是，接下来就会用到它了。

一个服务器必须得支持多用户访问，也就是并发。这里我们采用最简单的多线程方式。
当一个用户进入访问的时候，就新建一个线程，并且分离线程到后台执行（pthread_detach），就不用自己清理线程了。

```
Pthread_create(&tid, NULL, thread, (void *)connfd);
```

```
void *thread(void *vargp){
    int connfd = (int)vargp;
    Pthread_detach(pthread_self());
    doit(connfd);
    Close(connfd);
}
```

缓存比较复杂，后续再说。