第11章 网络编程

C/S 编程模式(Client/Server)

- 基本操作：Transaction（事务）
- 基本流程：
    1. Client发起Transaction，发送request
    2. Server收到request，解释并操作资源
    3. Server发送response给Client，等待下一个request
    4. Client收到response并处理
- Client&Server是进程，而不是machine或host

Network

- LAN(Local Area Network)：局域网，其中最流行的是Ethernet(以太网)
    - Ethernet segment(以太网段)：包括wires(双绞线)和hub(集线器)，hub完全同步所有端口
    - Ethernet adapter(以太网适配器)：全球唯一48bit地址
    - Bridge Ethernet(桥接以太网)：多个Ethernet segment用wire和bridge(网桥，会分配算法)连接起来形成
- WAN(Wide-Area Network，广域网)：高速的点到点电话连接
- internet(interconnected network)：多个不兼容的LAN&WAN用router(路由器)连接起来形成
- Protocol(协议)：解决了不兼容局域网间传送数据的问题
    1. HostA 上的client调用system call，把数据从virtual address复制到kernel buffer
    2. HostA 上的protocol software在数据前附加PH和FH1形成frame(PH<互联网络包头>寻址到Host B，FH1<LAN1的帧头>寻址到router)，然后传送frame到LAN1 adapter
    3. LAN1 adapter把frame复制到network
    4. Router's LAN1 adapter读取frame传给protocol software
    5. Router从PH提取出目标网络地址，把FH1换成FH2，把新的frame传给adapter
    6. Router's LAN2 adapter把frame复制到network
    7. HostB's adapter读取frame并传给protocol software
    8. HostB's protocol software剥落PH和FH2，最后把数据通过system call复制到virtual address

The Global IP Internet

- IP地址
    - 地址结构(历史遗留问题)

```
struct in_addr {
    uint32_t s_addr;/*Address in network byte order(big-endian)*/
}
```

    - big-endian/little-endian转换

```
#include <arpa/inet.h>

//返回网络字节顺序值
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);

//返回主机字节顺序值
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

  - IP address/dotted-decimal string转换

```
#include <arpa/inet.h>

//AF_INET(IPv4地址)、AF_INET6(IPv6地址)
//src(dotted-decimal)转换为dst(IP地址，网络字节顺序)
//成功返回1，src非法返回0，出错返回-1并设置errno
int inet_pton(AF_INET, const char *src, void *dst);

//src(IP地址，网络字节顺序)转换为dst(dotted-decimal,最多size个byte)
//成功返回dst，出错返回NULL
const char *inet_ntop(AF_INET, const void *src, char *dst, socklen_t size);
```

- Domain name
  - 域名集合形成一个层次结构，可以表示成一棵树
    - 第一层：未命名根节点
    - first-level：由ICANN定义，例如com、edu、gov、org、net
    - second-level：由ICANN的各个授权代理按照先到先服务的基础分配的，例如cmu.edu
    - 一旦一个组织得到了一个二级域名，就可以在subdomain中创建任何新的域名了，例如cs.cmu.edu
  - 域名集合与IP地址集合之间的映射通过DNS(Domain Name System)
  - 域名和IP地址可以是一一映射、一多映射、多一映射、没有映射
- 因特网连接
  - 点对点、全双工、可靠的
  - socket：address：port(16-bit)
    - 客户端port自动分配(ephemeral port临时端口)
    - 服务器port知名端口(参考 /etc/services)
      1. Web服务器 port：80 知名服务名：http
      2. 电子邮件服务器 port：25 知名服务名：smtp
  - socket pair：(cliaddr:cliport, servaddr:servport)

Socket Interface

- 地址结构：

```

```

```
//历史遗留问题，本可以是一个标量类型
struct in_addr{
    uint32_t s_addr; //in network byte orde (big-endian)
}

//以下两种数据结构强制转换，解决早期没有void*的困难
struct sockaddr_in{
    uint16_t sin_family; //Protocol(always AF_INET)
    uint16_t sin_port; //in network byte order
    struct in_addr sin_addr; //in network byte order
    unsigned char sin_zero[8];
}

struct sockaddr{
    uint16_t sa_family;
    char as_data[14];
}
```

- 函数

```
#include <sys/types.h>
#include <sys/socket.h>

/*
 *成功返回socket descriptor,出错返回-1
 *Usage: clientfd = Socket(AF_INET, SOCK_STREAM, 0);
 */
int socket(int domain, int type, int protocol);

/*
 *成功返回0，出错返回-1
 *与地址为addr的服务器建立连接，会阻塞，一旦成功clientfd就可以用于读写
 */
int connect(int clientfd, const struct sockaddr *addr, socklen_t addrlen);

/*
 *成功返回0，出错返回-1
 *把addr和sockfd联系起来
 */
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

/*
 *成功返回0，出错返回-1
 *把sockfd从active(主动)变成listening(监听)，backlog暗示了排队数量，一般取1024
 */
int listen(int sockfd, int backlog);

/*
 *成功返回非负connfd,出错返回-1
 *listenfd存在于服务器整个生命周期，connfd每次接受连接请求创建一次
ing accept(int listenfd, struct sockaddr *addr, int *addrlen);
```

- Host & Service 转换

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

struct addrinfo{
    int ai_flags;
    int ai_family;
    int ai_socktype;
    int ai_protocol;
    char *ai_canonname;
    size_t ai_addrlen;
    struct sockaddr *ai_addr;
    struct addrinfo *ai_next;
}

/*
 *成功返回0，错误返回错误代码
 *将host(hostname/hostaddr),service(servname/port)字符串转化为sockaddr
 *是reentrant(可重入的)
 *返回result，result是指向addrinfo的链表，addrinfo包含指向sockaddr
 *同时还会遍历result，依次尝试sockaddr，直到调用socket和connect/bind成功，fd会绑定到sockaddr
 *freeaddrinfo用于释放result
 *gai_strerror用于把返回的错误代码转化为字符串
 *host和service至少指定一个
 *可选参数hints只能设置ai_family, ai_socktype, ai_protocol和ai_flags 字段，其他必须为0/NULL（通常memset）
 ***默认返回IPv4和IPv6，ai_family可以设置AF_INET(限制为IPv4) 和 AF_INET6(限制为IPv6)
 ***对于host关联的每个地址，默认返回最多三个addrinfo(他们的ai_socktype不同：connections,datagrams,raw socket)，ai_socket设置为SOCK_STREAM将链表限制为每个地址最多一个addrinfo，该addrinfo可以作为连接的一个端点
 ***ai_flags是bit amsk
 *****AI_ADDRCONFIG：如果主机被配置为IPv4就只返回IPv4，IPv6也一样
 *****AI_CANONNAME：默认为NULL。如果设置了，result中第一个addrinfo的ai_canonname指向host的canonical name
 *****AI_NUMERiCSERV：强制参数service为port number
 *****AI_PASSIVE：默认返回主动套接字。服务器监听时设置此位，同时host应为NULL，得到的sockaddr会是wildcard address(通配符地址)
 */
int getaddrinfo(const char *host, const char *service, const struct addrinfo *hints, struct addrinfo **result);

void freeaddrinfo(struct addrinfo *result);

const char *gai_strerror(int errcode);

/*
 *成功返回0，错误返回错误代码
```

```
 *把sa转化为对应的host和service
 *host和service必须设置其中之一，对应于hostlen/servlen设置为0
 *flags是bit mask
 ***NI_NUMERICHOST：默认返回host域名，设置此位返回数字地址字符串
 ***NI_NUMERICSERV：默认检查/etc/services，尽量返回servname，设置此位会简单返回
port number
 */
int getnameinfo(const struct sockaddr *sa, socklen_t aslen, char *host,
size_t hostlen,    char *service, size_t servlen,    int flags);
```

- 辅助函数

```
#include "csapp.h"

/*
 *建立与hostname上port端口的server的连接
 *成功返回clientfd，出错返回-1
 */
int open_clientfd(char *hostname, char *port);

/*
 *服务器在port端口创建一个listenfd，准备接受连接请求
 *成功返回listenfd，出错返回-1
 */
int open_listenfd(char *port);
```

Web Servers

- Web Basics
  - HTTP: 与FTP相比可以使用HTML的语言编写，并且可以包含任何host上的指针(超链接)
  - MIME(Multipurpose Internet Mail Extensions,多用途的网际邮件扩充协议)

MIME类型	描述
text/html	HTML页面
text/plain	无格式文本
application/postscript	Postscript文档
image/gif	GIF格式编码的二进制图像
image/png	PNG格式编码的二进制图像
image/jpeg	JPEG格式编码的二进制图像

  - Web服务器返回static content/dynamic content，这些内容都与一个URL标识的文件相关联
- HTTP Transactions
  - HTTP Request: request line + request headers + 空文本行

```
GET /index.html?2333 HTTP/1.1
Host: www.baidu.com
\r\n
```

- HTTP Response: response line + response headers + 空文本行 + response body

```
HTTP/1.0 200 OK
MIME-Version: 1.0
Date: Mon, 8 Jan 2010 44:59:42 GMT
Server: Apache-Coyote/1.1
Content-Type: text/html
Content-Length: 42092
\r\n
<html>...
</html>
```

Status code	**Status message**	描述
200	OK	处理请求无误
403	Forbidden	服务器无权访问所请求的文件
404	Not found	服务器不能找到所请求的文件
501	Not implemented	服务器不支持请求的方法

- Serving Dynamic Content
  - CGI(Common Gateway Interface, 通用网关接口): 解决了客户端-服务器-子进程间的参数传递和通信问题。
  - Client to Server Argument: 在request URI中?传递，POST在request body中传递
  - Server to Child Argument: fork->CGI环境变量QUERY_STRING="arg1&arg2..."->execve(/cgi-bin/adder)
  - Server to Child otherInfo: 通过下列环境变量

环境变量	描述
QUERY_STRING	程序参数
SERVER_PORT	服务端口
REQUEST_METHOD	请求方法
REMOTE_HOST	客户端域名
REMOTE_ADDR	客户端点分十进制IP地址
CONTENT_TYPE	POST only：请求体的MIME类型
CONTENT_LENGTH	POST only：请求体的字节大小

  - 子进程输出：CGI调用前使用dup2将标准输出重定向到clientfd