哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场

计算机网络之网尽其用

主讲人：李全龙

本讲主题

Socket编程-服务器软件设计

4种类型基本服务器

- 循环无连接(Iterative connectionless)服务器
- 循环面向连接(Iterative connection-oriented)服务器
- 并发无连接(Concurrent connectionless)服务器
- 并发面向连接(Concurrent connection-oriented)服务器

循环：一次只处理一个client请求，处理完后再处理下一个（无并发，顺序处理）

无连接：UDP

循环无连接服务器基本流程

1. 创建套接字(UDP)
2. 绑定端点地址（**INADDR_ANY**+端口号）
3. 反复接收来自客户端的请求
4. 遵循应用层协议，构造响应报文，发送给客户

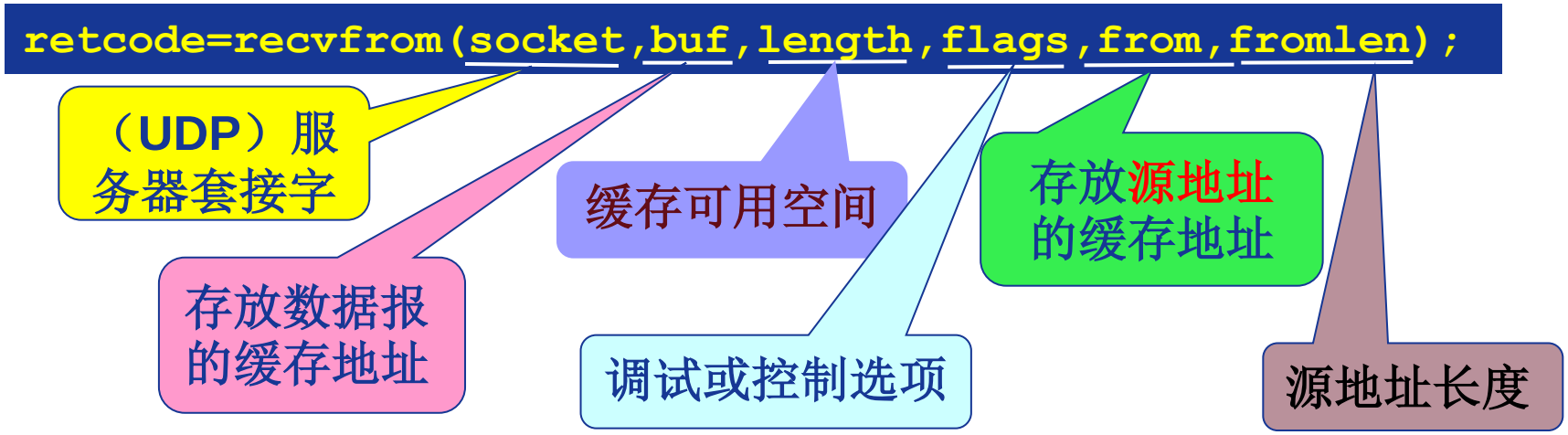
数据发送

- 服务器端不能使用*connect()*函数
- 无连接服务器使用*sendto()*函数发送数据报

```
retcode=sendto(socket,data,length,flags,destaddr,addrlen);
```

获取客户端点地址

- 调用*recvfrom()*函数接收数据时，自动提取



循环面向连接服务器基本流程

1. 创建（主）套接字，并绑定熟知端口号；
2. 设置（主）套接字为被动监听模式，准备用于服务器；
3. 调用**accept()**函数接收下一个连接请求（通过主套接字），创建新套接字用于与该客户建立连接；
4. 遵循应用层协议，反复接收客户请求，构造并发送响应(通过新套接字)；
5. 完成为特定客户服务后，关闭与该客户之间的连接，返回步骤3.

并发无连接服务器基本流程

主线程1: 创建套接字，并绑定熟知端口号；

主线程2: 反复调用***recvfrom()***函数，接收下一个客户请求，并创建新线程处理该客户响应；

子线程1: 接收一个特定请求；

子线程2: 依据应用层协议构造响应报文，并调用***sendto()***发送；

子线程3: 退出(一个子线程处理一个请求后即终止)。

并发面向连接服务器基本流程

主线程1: 创建（主）套接字，并绑定熟知端口号；

主线程2: 设置（主）套接字为被动监听模式，准备用于服务器；

主线程3: 反复调用***accept()***函数接收下一个连接请求（通过主套接字），并创建一个新的子线程处理该客户响应；

子线程1: 接收一个客户的服务请求（通过新创建的套接字）；

子线程2: 遵循应用层协议与特定客户进行交互；

子线程3: 关闭/释放连接并退出（线程终止）.

服务器的实现

- 设计一个底层过程隐藏底层代码：
  - *passivesock()*
- 两个高层过程分别用于创建服务器端UDP套接字和TCP套接字（调用passivesock()函数）：
  - *passiveUDP()*
  - *passiveTCP()*

服务器的实现-*passivesock()*

```
/* passsock.cpp - passivesock */
#include <stdlib.h>
#include <string.h>
#include <winsock.h>
void	errexit(const char *, ...);

/*------------------------------------------------------------------------
 * passivesock - allocate & bind a server socket using TCP or UDP
 *------------------------------------------------------------------------
 */
SOCKET passivesock(const char *service, const char *transport, int qlen)
```

服务器的实现-passivesock()

```
{
    struct servent *pse;    /* pointer to service information entry      */
    struct protoent *ppe;  /* pointer to protocol information entry     */
    struct sockaddr_in sin;/* an Internet endpoint address              */
    SOCKET      s;          /* socket descriptor                         */
    int     type;     /* socket type (SOCK_STREAM, SOCK_DGRAM)*/

    memset(&sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = INADDR_ANY;

   /* Map service name to port number */
    if ( pse = getservbyname(service, transport) )
            sin.sin_port = (u_short)pse->s_port;
    else if ( (sin.sin_port = htons((u_short)atoi(service))) == 0 )
            errexit("can't get \"%s\" service entry\n", service);
```

服务器的实现-*passivesock()*

```
 /* Map protocol name to protocol number */
 if ( (ppe = getprotobyname(transport)) == 0)
        errexit("can't get \"%s\" protocol entry\n", transport);
/* Use protocol to choose a socket type */
 if (strcmp(transport, "udp") == 0)
        type = SOCK_DGRAM;
 else
        type = SOCK_STREAM;
/* Allocate a socket */
 s = socket(PF_INET, type, ppe->p_proto);
 if (s == INVALID_SOCKET)
        errexit("can't create socket: %d\n", GetLastError());
/* Bind the socket */
 if (bind(s, (struct sockaddr *)&sin, sizeof(sin)) == SOCKET_ERROR)
        errexit("can't bind to %s port: %d\n", service,
                    GetLastError());
 if (type == SOCK_STREAM && listen(s, qlen) == SOCKET_ERROR)
        errexit("can't listen on %s port: %d\n", service,
                    GetLastError());
 return s;}
```

TCP需要调用

服务器的实现-*passiveUDP()*

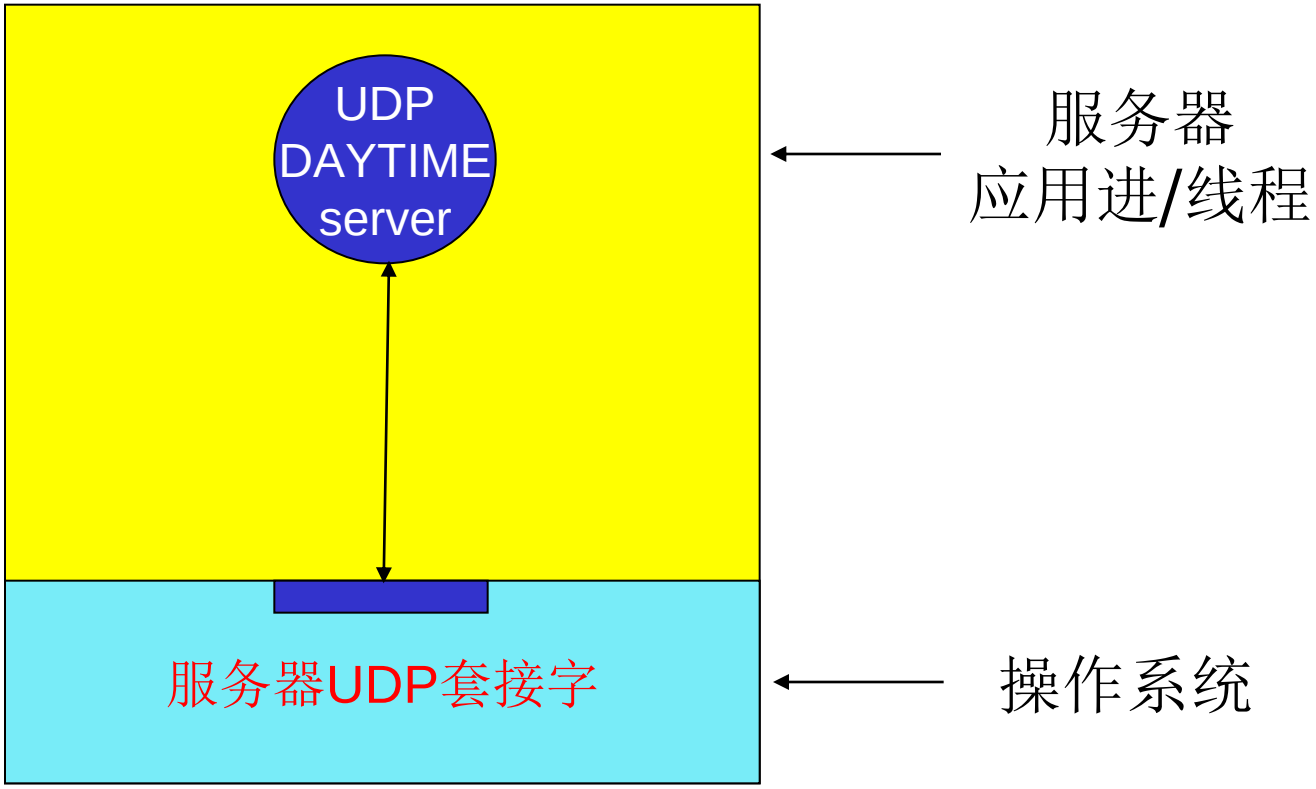
```
/* passUDP.cpp - passiveUDP */

#include <winsock.h>

SOCKET passivesock(const char *, const char *, int);

/*------------------------------------------------------------------------
 * passiveUDP - create a passive socket for use in a UDP server
 *------------------------------------------------------------------------
 */
SOCKET passiveUDP(const char *service)
{
    return passivesock(service, "udp", 0);
}
```

服务器的实现-passiveTCP()

```
/* passTCP.cpp - passiveTCP */

#include <winsock.h>

SOCKET passivesock(const char *, const char *, int);

/*------------------------------------------------------------------------
 * passiveTCP - create a passive socket for use in a TCP server
 *------------------------------------------------------------------------
 */
SOCKET passiveTCP(const char *service, int qlen)
{
    return passivesock(service, "tcp", qlen);
}
```

队列大小

例1：无连接循环DAYTIME服务器



例1：无连接循环DAYTIME服务器

```
/* UDPdtd.cpp - main, UDPdaytimed */
#include <stdlib.h>
#include <winsock.h>
#include <time.h>

void	errexit(const char *, ...);
SOCKET passiveUDP(const char *);

#define WSVERS		MAKEWORD(2, 0)

/*------------------------------------------------------------------------
 * main - Iterative UDP server for DAYTIME service
 *------------------------------------------------------------------------
 */
void main(int argc, char *argv[])
```

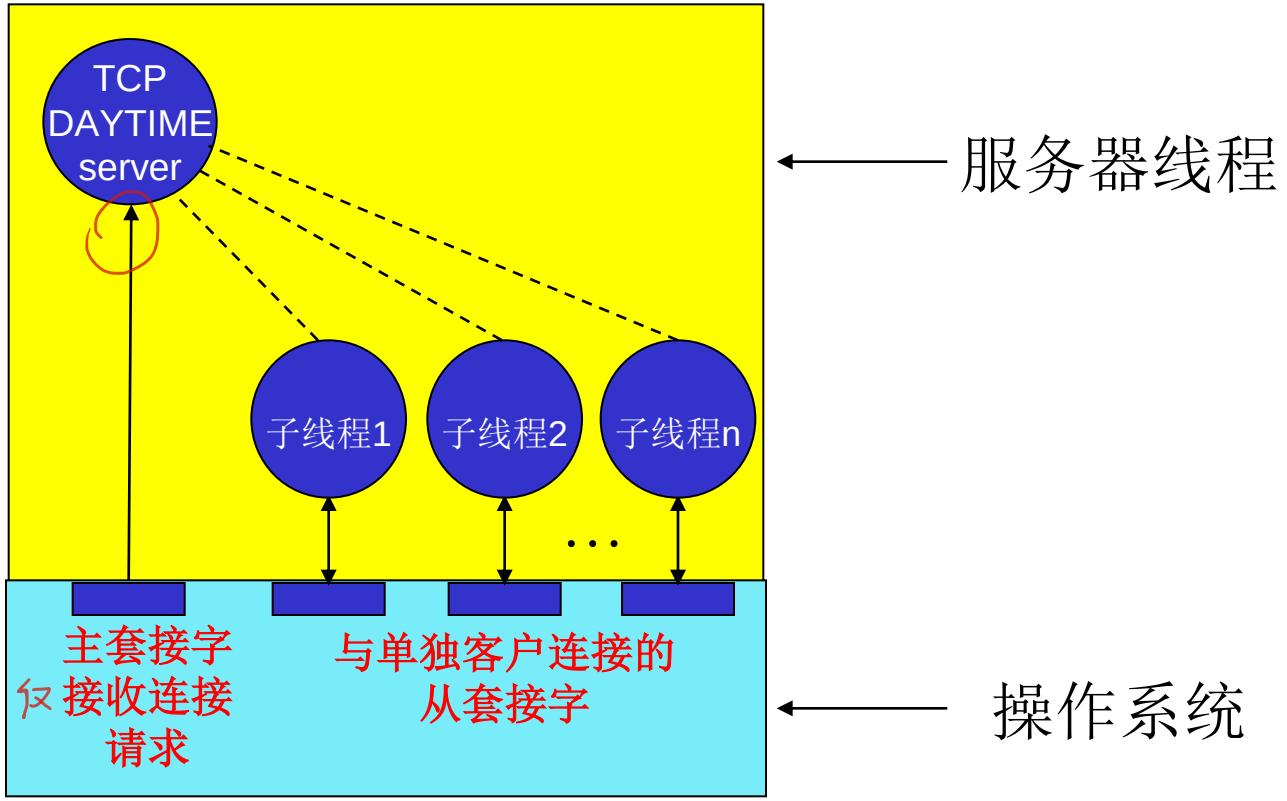
例1：无连接循环DAYTIME服务器

```
{
    struct sockaddr_in fsin;          /* the from address of a client     */
    char  *service = "daytime";       /* service name or port number      */
    SOCKET sock;                      /* socket                           */
    int     alen;                     /* from-address length              */
    char  *  pts;                     /* pointer to time string           */
    time_t  now;                      /* current time                     */
    WSADATA wsadata;

    switch (argc)
    {
      case             1:
            break;
      case             2:
            service = argv[1];
            break;
      default:
            errexit("usage: UDPdaytimed [port]\n");
    }
```

例1：无连接循环DAYTIME服务器

```
if (WSAStartup(WSVERS, &wsadata) != 0)
        errexit("WSAStartup failed\n");

    sock = passiveUDP(service);

    while (1)
     {
        alen = sizeof(struct sockaddr);
        if (recvfrom(sock, buf, sizeof(buf), 0,
            (struct sockaddr *)&fsin, &alen) == SOCKET_ERROR)
                errexit("recvfrom: error %d\n", GetLastError());
        (void) time(&now);
        pts = ctime(&now);
        (void) sendto(sock, pts, strlen(pts), 0,
                (struct sockaddr *)&fsin, sizeof(fsin));
    }
    return 1;       /* not reached */
}
```

例2：面向连接并发DAYTIME服务器



例2：面向连接并发DAYTIME服务器

```
/* TCPdtd.cpp - main, TCPdaytimed */
#include <stdlib.h>
#include <winsock.h>
#include <process.h>
#include <time.h>

void	errexit(const char *, ...);
void	TCPdaytimed(SOCKET);
SOCKET passiveTCP(const char *, int);

#define QLEN  5
#define WSVERS  MAKEWORD(2, 0)

/*------------------------------------------------------------------------
 * main - Concurrent TCP server for DAYTIME service
 *------------------------------------------------------------------------
 */
void main(int argc, char *argv[])
```

例2：面向连接并发DAYTIME服务器

```
{
    struct sockaddr_in fsin;        /* the from address of a client     */
    char *service = "daytime";      /* service name or port number*/
    SOCKET msock, ssock;            /* master & slave sockets           */
    int    alen;                    /* from-address length              */
    WSADATA wsadata;

    switch (argc) {
    case 1:
            break;
    case 2:
            service = argv[1];
            break;
    default:
            errexit("usage: TCPdaytimed [port]\n");
    }
```

例2：面向连接并发DAYTIME服务器

```
if (WSAStartup(WSVERS, &wsadata) != 0)
        errexit("WSAStartup failed\n");

    msock = passiveTCP(service, QLEN);

    while (1) {
        alen = sizeof(struct sockaddr);
        ssock = accept(msock, (struct sockaddr *)&fsin, &alen);
        if (ssock == INVALID_SOCKET)
                errexit("accept failed: error number %d\n",
                        GetLastError());
        if (_beginthread((void (*)(void *)) TCPdaytimed, 0,
            (void *)ssock) < 0) {
                errexit("_beginthread: %s\n", strerror(errno));
        }
    }
    return 1;       /* not reached */
}
```

例2：面向连接并发DAYTIME服务器

```
/*------------------------------------------------------------------
 * TCPdaytimed - do TCP DAYTIME protocol
 *------------------------------------------------------------------
 */
void TCPdaytimed(SOCKET fd)
{
    char *          pts;                /* pointer to time string   */
    time_t           now;               /* current time             */

    (void) time(&now);
    pts = ctime(&now);
    (void) send(fd, pts, strlen(pts), 0);
    (void) closesocket(fd);
}
```

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场

谢 谢！