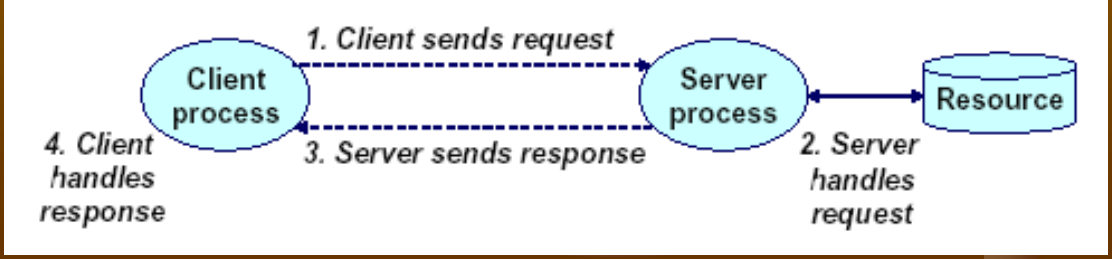
Socket Programming

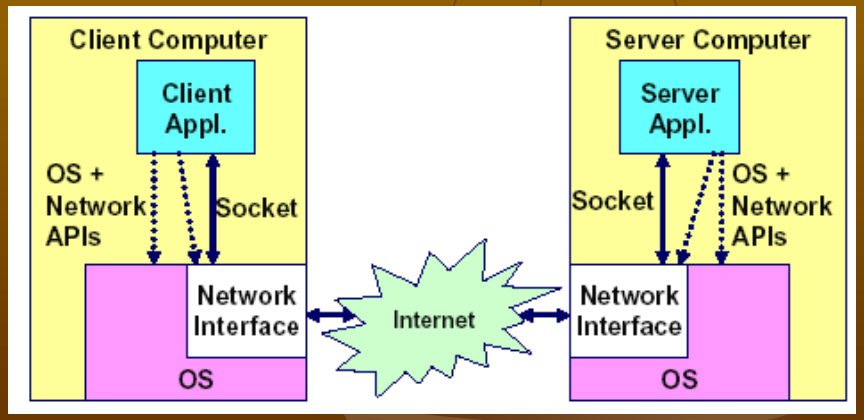
Pendahuluan

- Aplikasi di jaringan, transaksinya didasarkan pada konsep *client-server*. Sebuah atau beberapa *client* meminta/*request* pelayanan ke server.
- Aplikasi *client-server* menggunakan protokol *tranport* untuk saling berinteraksi.
- Ketika proses interaksi terjadi, suatu aplikasi harus memberikan informasi-informasi secara detail tentang :
  - Informasi tentang apakah dia *client* atau *server.*
  - Pengirim memberikan informasi tentang data yang dikirim.
  - Penerima memberikan informasi tentang dimana data diletakkan, dll



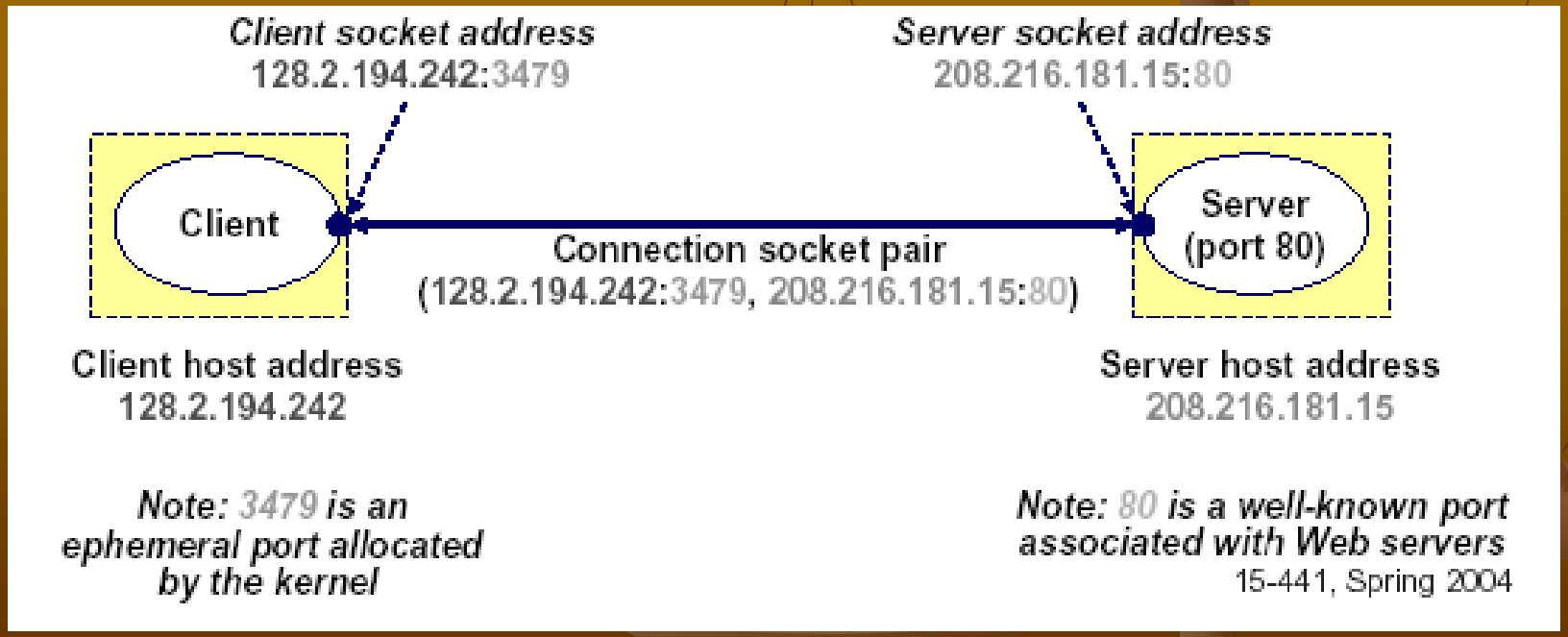
Konsep Socket

- Antarmuka (*interface*) antara program aplikasi dengan protokol komunikasi pada suatu sistem operasi disebut *Application Program Interface (API)*. *API* didefinisikan sebagai suatu kumpulan instruksi yang mendukung proses interaksi antara suatu perangkat lunak dengan suatu protokol yang digunakan.
- Pada mesin keluarga *Linux*, *socket* terintegrasi dengan *I/O* sehingga aplikasi yang berkomunikasi dengan *socket,* cara kerjanya sama dengan suatu aplikasi yang mengakses peralatan *I/O*. Oleh karena itu untuk memahami cara kerja socket pada *Linux*, sebelumnya harus juga memahami fasilitas *I/O* pada *Linux*.



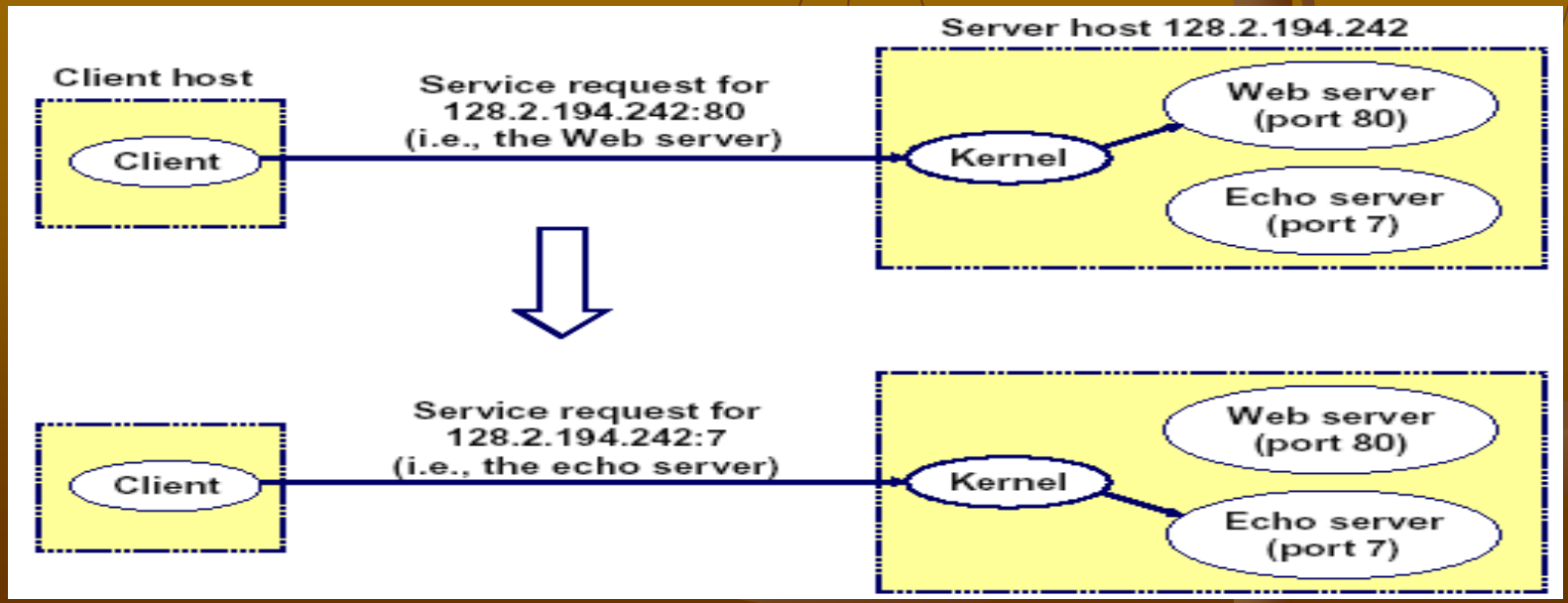
Konsep Socket

- Pada saat suatu aplikasi berkomunikasi, awalnya aplikasi membuat *socket* baru, maka pada aplikasi tersebut akan diberikan nomer yang digunakan sebagai referensi *socket*. Jika ada suatu sistem yang menggunakan nomer referensi *socket* tersebut, maka akan terjalin suatu jaringan komunikasi antar komputer sebaik transfer data lokal



Konsep Socket…

- Untuk berkomunikasi dengan *server*, *client* harus tahu nomor *IP server* begitu juga nomor *port* yang dituju, nomor port menunjukkan service yang dijalankan. Contoh port 23 untuk *Telnet Server*, port 25 untuk *Mail Server* dan port 80 untuk *Web Server*. Dalam hal ini aplikasi di client sudah mengetahui port yang akan dituju. Contoh program aplikasi di client yang meminta service di server ada;ah ftp, telnet, ssh. Untuk melihat service bisa dilihat pada file */etc/services*.
- Program yang berjalan di server, akan berjalan sepanjang waktu (disebut sebagai daemon) sampai mesin/service dimatikan, menunggu request dari *client* sesuai service yang diminta.



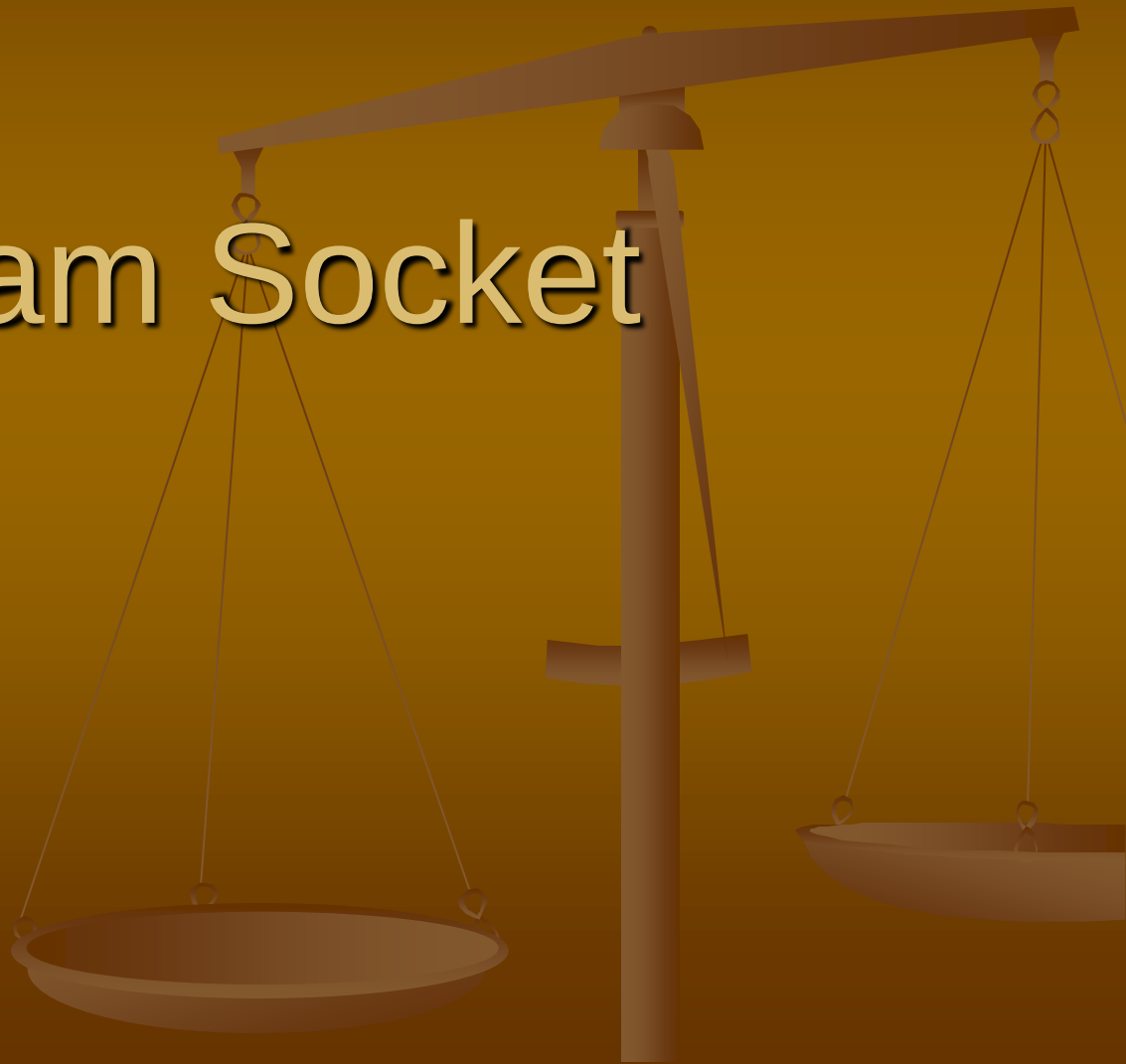
Procedure Utama Socket

- Client, meminta layanan, langkah :
  - Membuka koneksi client ke server, yang di dalamnya adalah :
    - Membuat socket dengan perintah socket()
    - melakukan pengalamatan ke server.
    - Menghubungi server dengan *connect()*
  - Melakukan komunikasi (mengirim dan menerima data), dengan menggunakan perintah *write()* dan *read()*
  - Menutup hubungan
- Server, menyediakan layanan, langkah:
  - Melakukan prosedur pembukaan koneksi yang di dalamnya berupa langkah – langkah : membuat socket, mengikat socket, menyiapkan socket menerima koneksi, pengalamatan socket
  - Looping utama adalah menerima koneksi, dan melakukan komunikasi data (mengirim dan menerima).

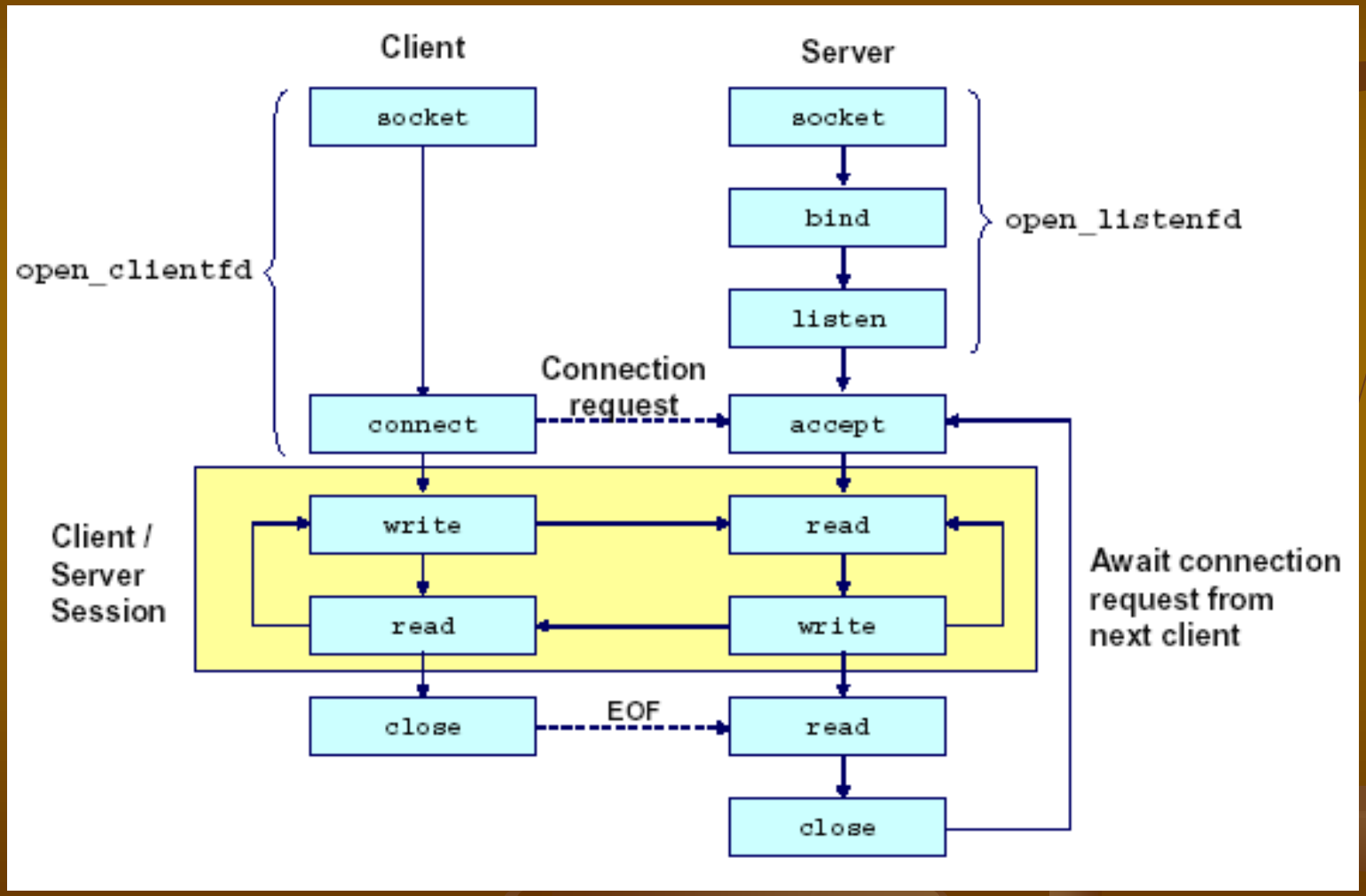
Tipe Socket

- User Datagram Socket
  - connectionless socket
  - Jika client mengirimkan data ke server, data tersebut ada kemungkinan sampai ke server atau tidak. Untuk itu client menunggu sinyal 'error free' dari client.
  - Jika client tidak menerima sinyal 'error free' dalam suatu kurun waktu, maka client akan mengirimkan lagi data tersebut.
  - Contoh aplikasi yang menggunakan datagram socket adalah **tftp** dan **bootp**.
- Stream Socket
  - Connection oriented socket
  - Client dan server harus saling berkomunikasi sebelum transfer data.
  - Pada stream socket dalam melakukan koneksi salah satu prosedur yang penting adalah *accept()*, yang dipakai untuk menerima koneksi dari client

Stream Socket



Mekanisme Stream Socket

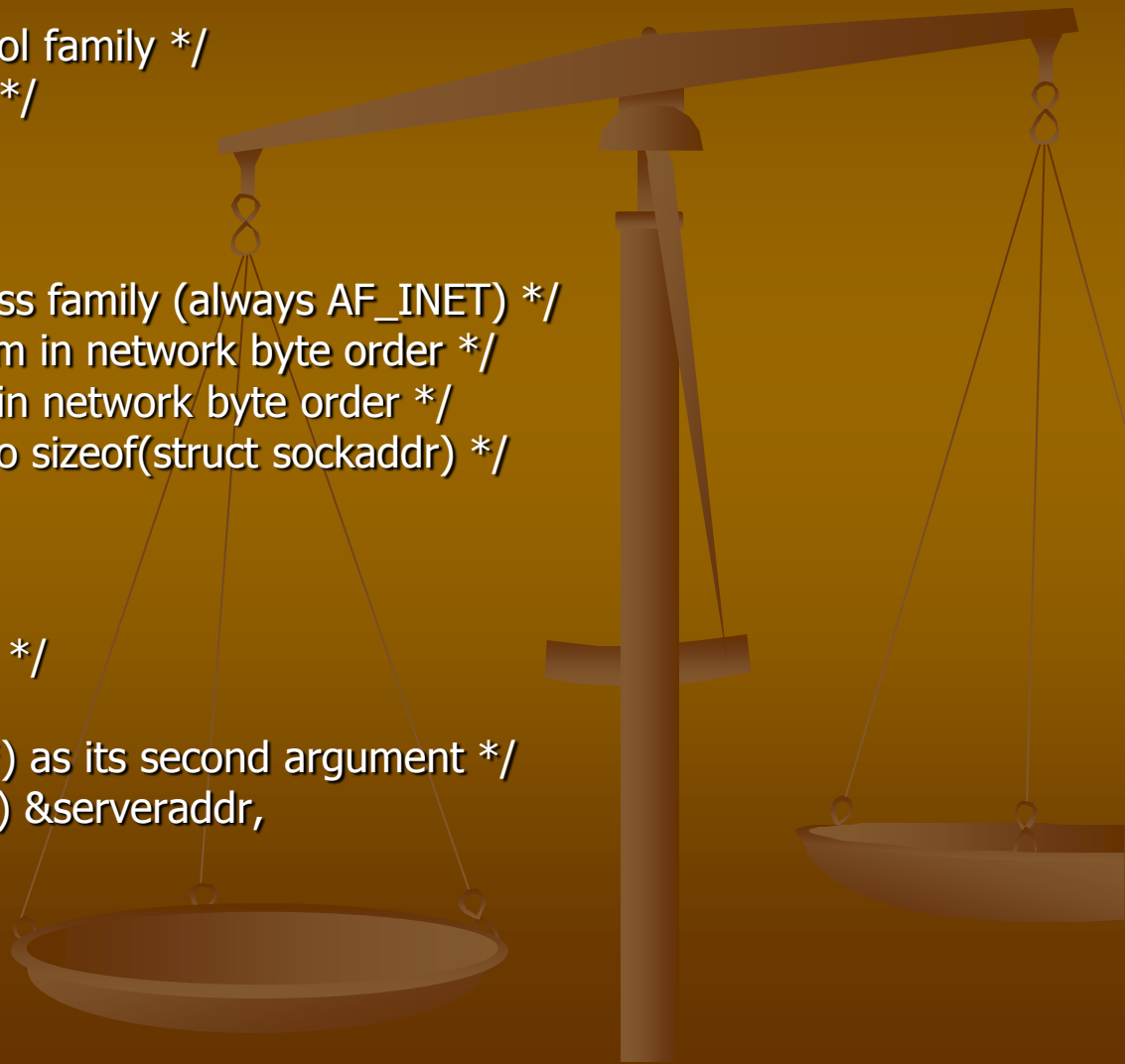


Socket Programming

- Langkah – langkah dasar di client :
  - Membuka koneksi client ke server, yang di dalamnya adalah :
    - Membuat socket dengan perintah socket()
    - melakukan pengalamatan ke server.
    - Menghubungi server dengan *connect()*
  - Melakukan komunikasi (mengirim dan menerima data), dengan menggunakan perintah *write()* dan *read()*
  - Menutup hubungan dengan perintah *close()*;
- Langkah – langkah dasar di server :
  - Membuat socket dengan perintah *socket()*
  - Mengikatkan socket kepada sebuah alamat network dengan perintah *bind()*
  - Menyiapkan socket untuk menerima koneksi yang masuk dengan perintah *listen()*
  - Menerima koneksi yang masuk ke server dengan perintah *accept()*
  - Melakukan komunikasi (mengirim dan menerima data), dengan menggunakan perintah *write()* dan *read()*

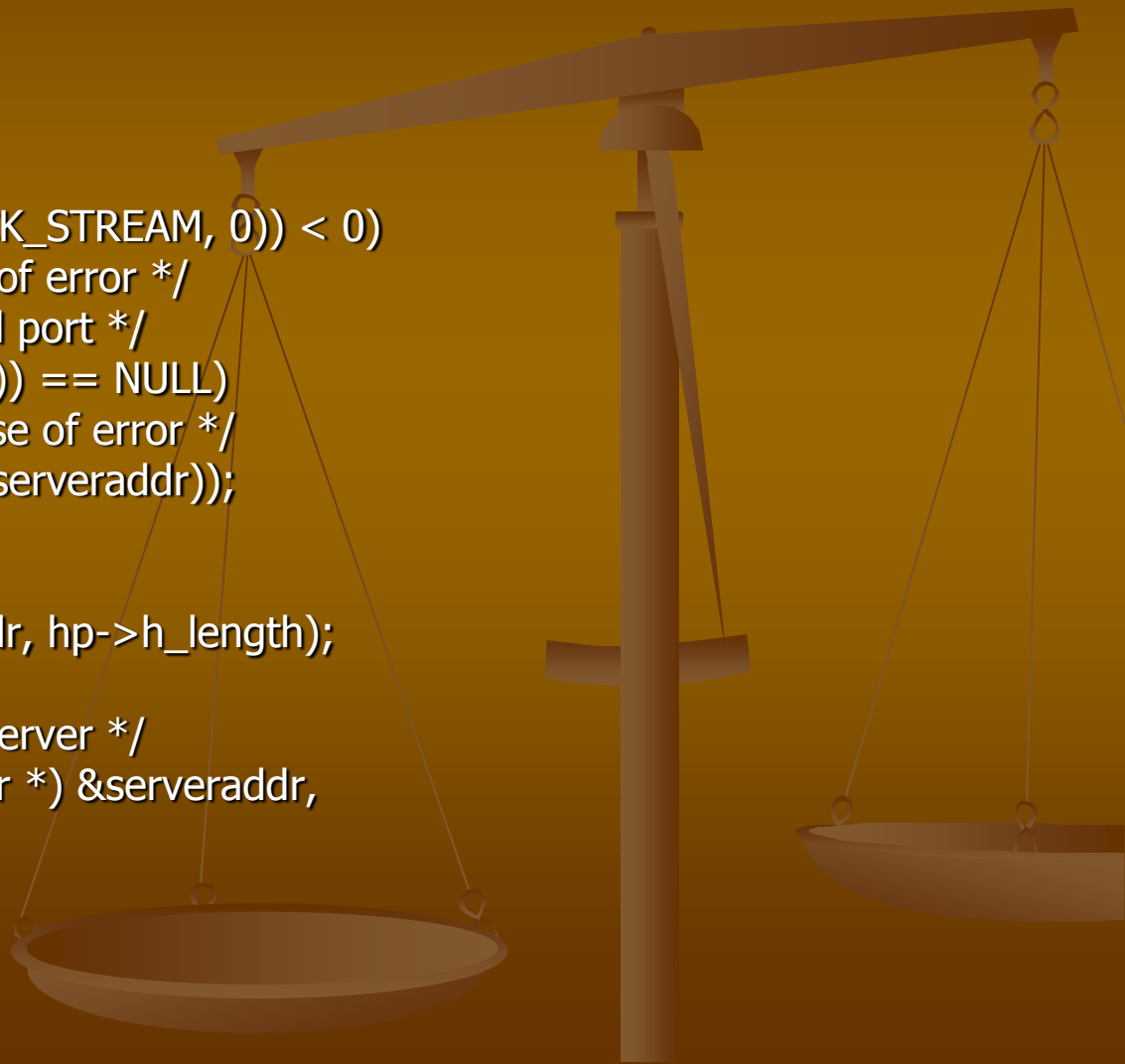
Struktur Pengalamatan

```
struct sockaddr {
unsigned short sa_family; /* protocol family */
char sa_data[14]; /* address data. */
};

struct sockaddr_in {
unsigned short sin_family; /* address family (always AF_INET) */
unsigned short sin_port; /* port num in network byte order */
struct in_addr sin_addr; /* IP addr in network byte order */
unsigned char sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};

struct sockaddr_in serveraddr;
/* fill in serveraddr with an address */
…
/* Connect takes (struct sockaddr *) as its second argument */
connect(clientfd, (struct sockaddr *) &serveraddr,
sizeof(serveraddr));
```



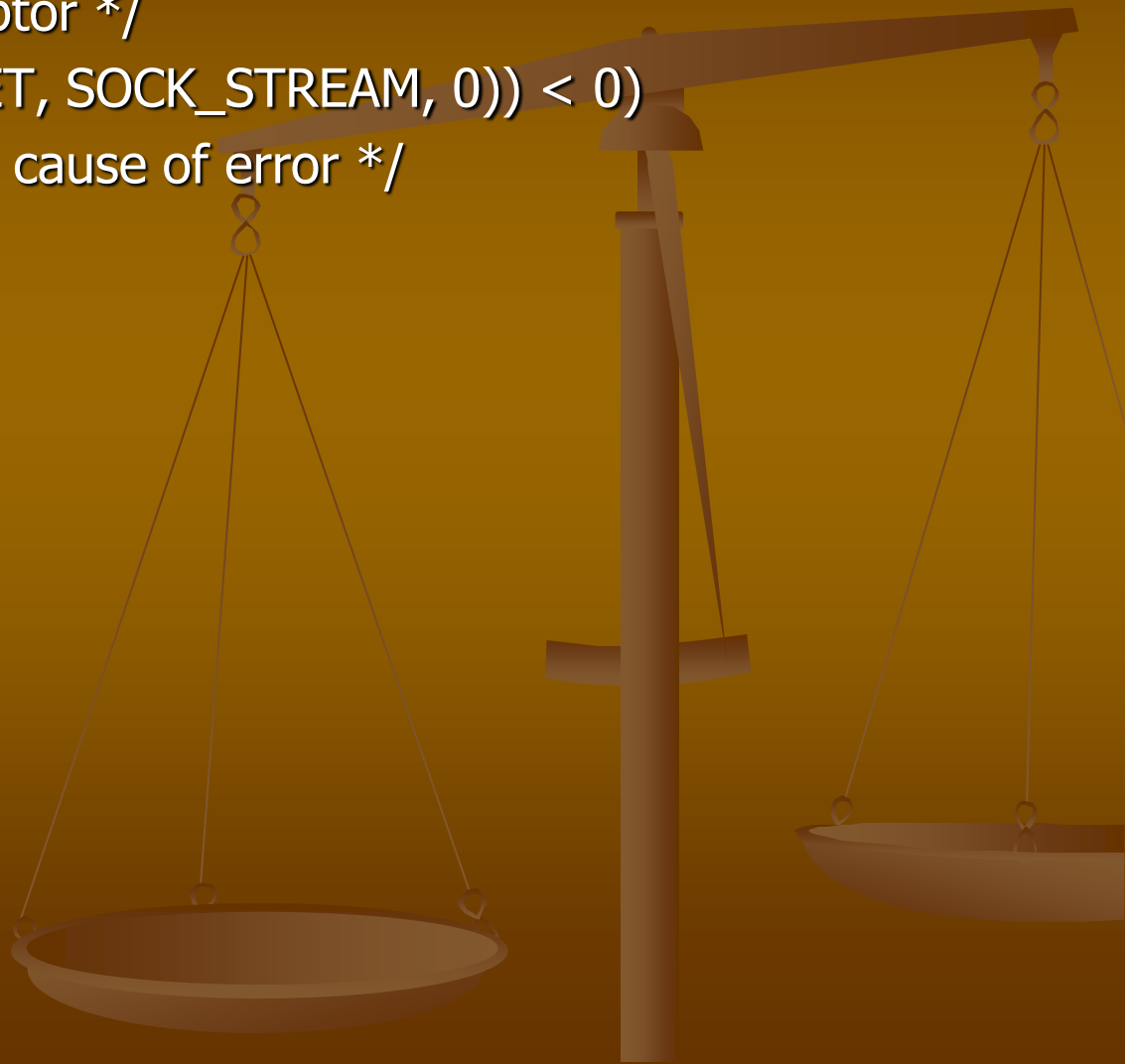
Prosedure Pembukaan Koneksi

```
int open_clientfd(char *hostname, int port)
{
int clientfd;
struct hostent *hp;
struct sockaddr_in serveraddr;
if ((clientfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
return -1; /* check errno for cause of error */
/* Fill in the server's IP address and port */
if ((hp = gethostbyname(hostname)) == NULL)
return -2; /* check h_errno for cause of error */
bzero((char *) &serveraddr, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
bcopy((char *)hp->h_addr,
(char *)&serveraddr.sin_addr.s_addr, hp->h_length);
serveraddr.sin_port = htons(port);
/* Establish a connection with the server */
if (connect(clientfd, (struct sockaddr *) &serveraddr,
sizeof(serveraddr)) < 0)
return -1;
return clientfd;
}
```



Prosedure Pembuatan socket

```
int clientfd; /* socket descriptor */
if ((clientfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
return -1; /* check errno for cause of error */
... (more)
```



Prosedure Pengalamatan

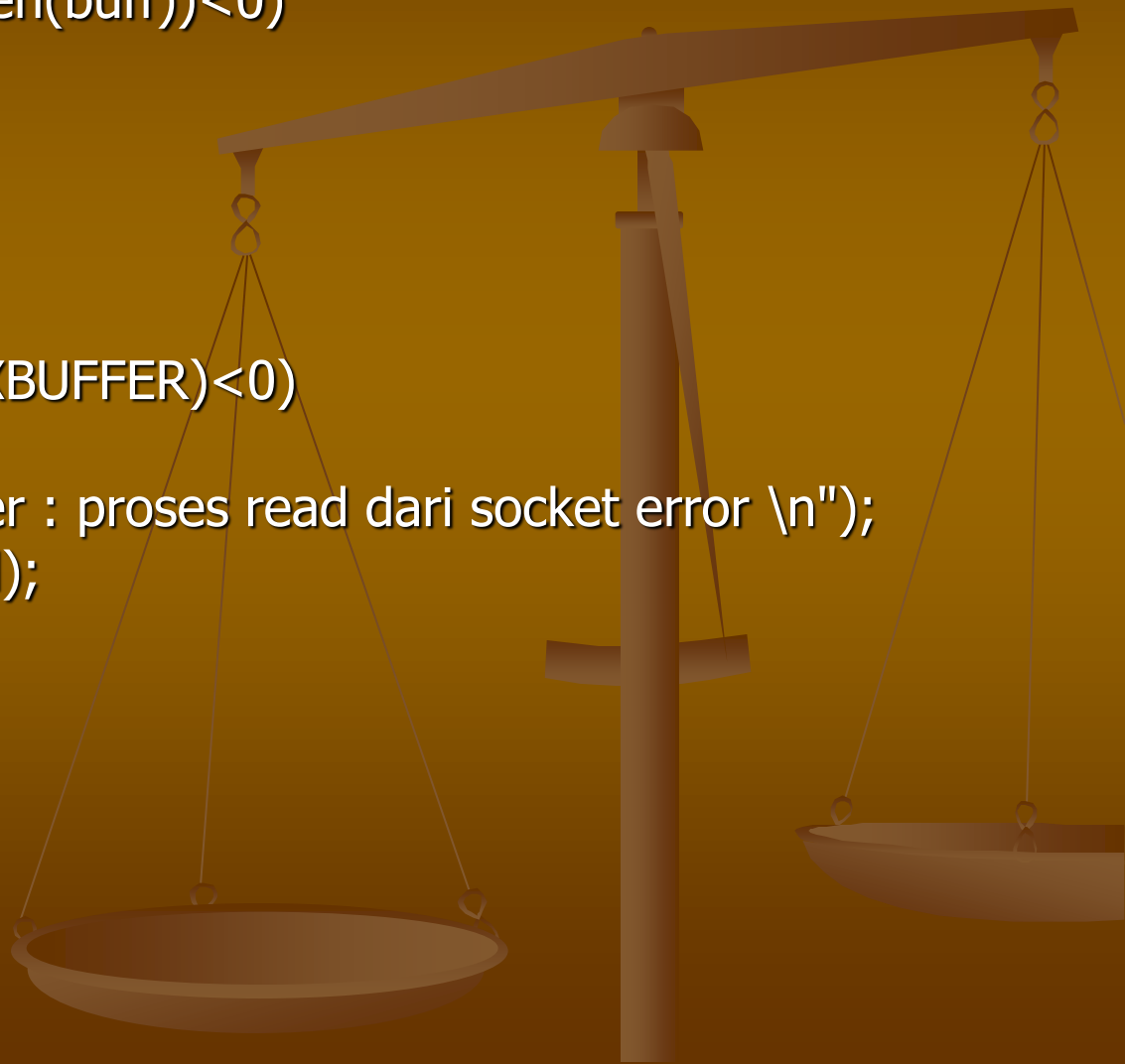
```
int clientfd; /* socket descriptor */
struct hostent *hp; /* DNS host entry */
struct sockaddr_in serveraddr; /* server's IP address */
...
/* fill in the server's IP address and port */
if ((hp = gethostbyname(hostname)) == NULL)
return -2; /* check h_errno for cause of error */
bzero((char *) &serveraddr, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
bcopy((char *)hp->h_addr,
(char *)&serveraddr.sin_addr.s_addr, hp->h_length);
serveraddr.sin_port = htons(port);
```

Prosedure Koneksi ke Server

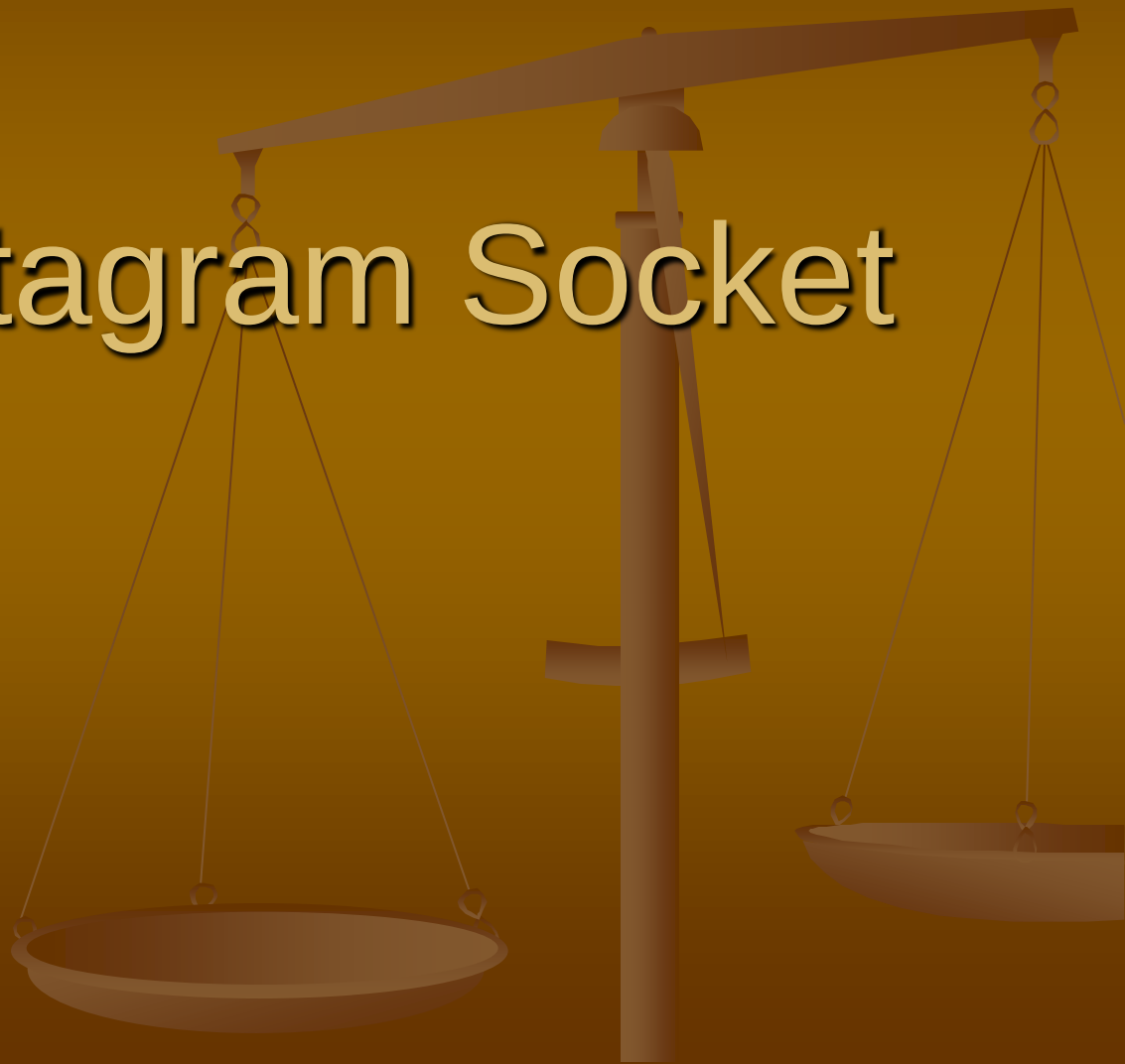
```
int clientfd; /* socket descriptor */
struct hostent *hp; /* DNS host entry */
struct sockaddr_in serveraddr; /* server's IP address */
...
/* fill in the server's IP address and port */
if ((hp = gethostbyname(hostname)) == NULL)
return -2; /* check h_errno for cause of error */
bzero((char *) &serveraddr, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
bcopy((char *)hp->h_addr,
(char *)&serveraddr.sin_addr.s_addr, hp->h_length);
serveraddr.sin_port = htons(port);
```

Prosedure Pengiriman dan Penerimaan Data

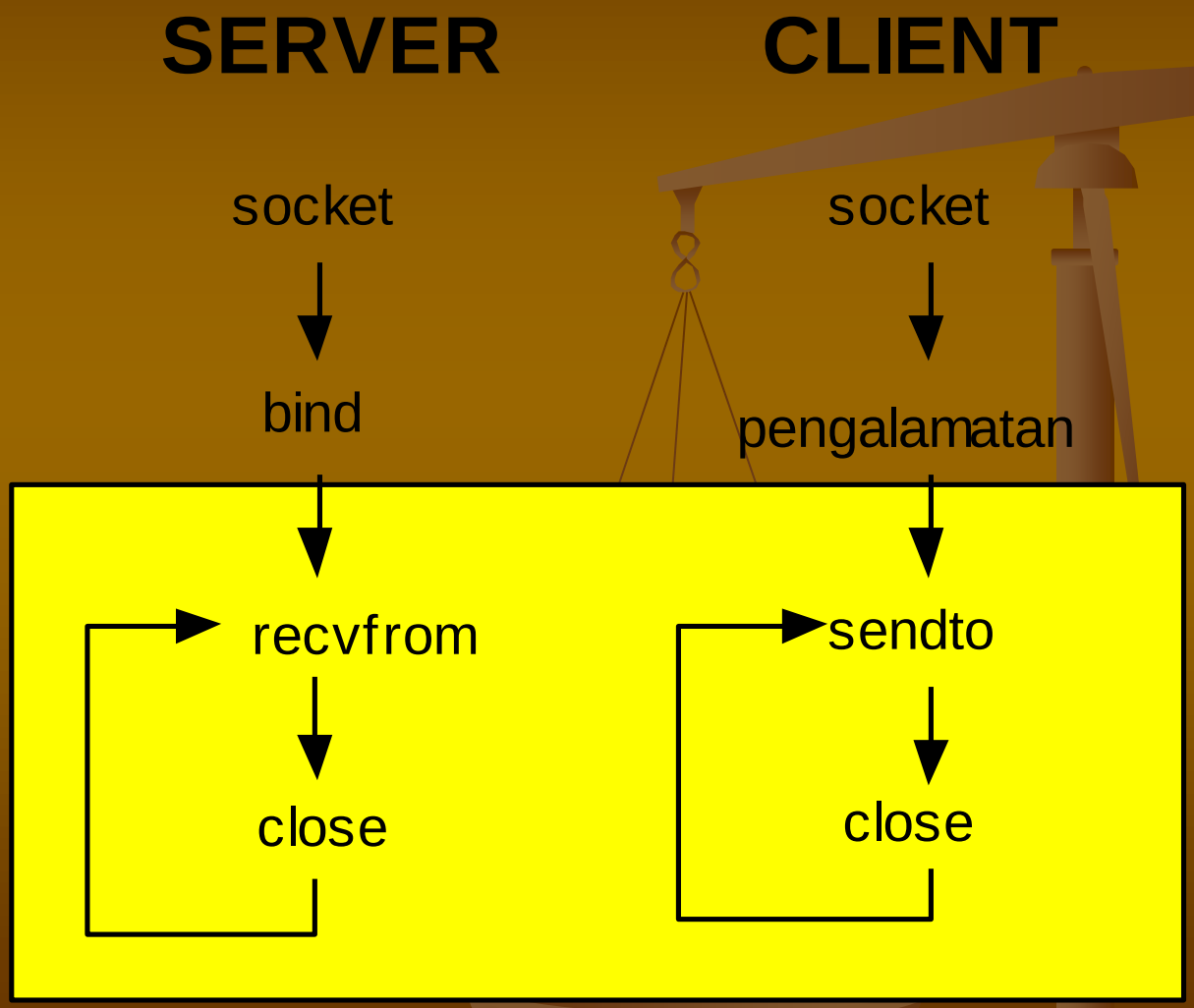
```
if (write(sockfd, buff, strlen(buff))<0)
{
        close(sockfd);
        exit(1);
}

if (read(sockfd, buff, MAXBUFFER)<0)
    {
                printf("server : proses read dari socket error \n");
                close(sockfd);
                exit(1);
    }
```



User Datagram Socket



Algoritma Datagram Socket



Procedure Datagram Socket

- Langkah – langkah dasar di *client* :
  - Membuka koneksi *client* ke *server*, yang di dalamnya adalah :
  - Membuat socket dengan perintah socket().
  - melakukan pengalamatan ke server.
  - Melakukan komunikasi (mengirimkan data), dengan menggunakan perintah *sendto()*
  - Menutup hubungan dengan perintah *close()*;
- Langkah – langkah dasar di server :
  - Membuat socket dengan perintah *socket()*
  - Mengikatkan socket kepada sebuah alamat network dengan perintah *bind()*
  - Melakukan komunikasi (menerima data), dengan menggunakan perintah *recvfrom()*

Struktur Pengalamatan

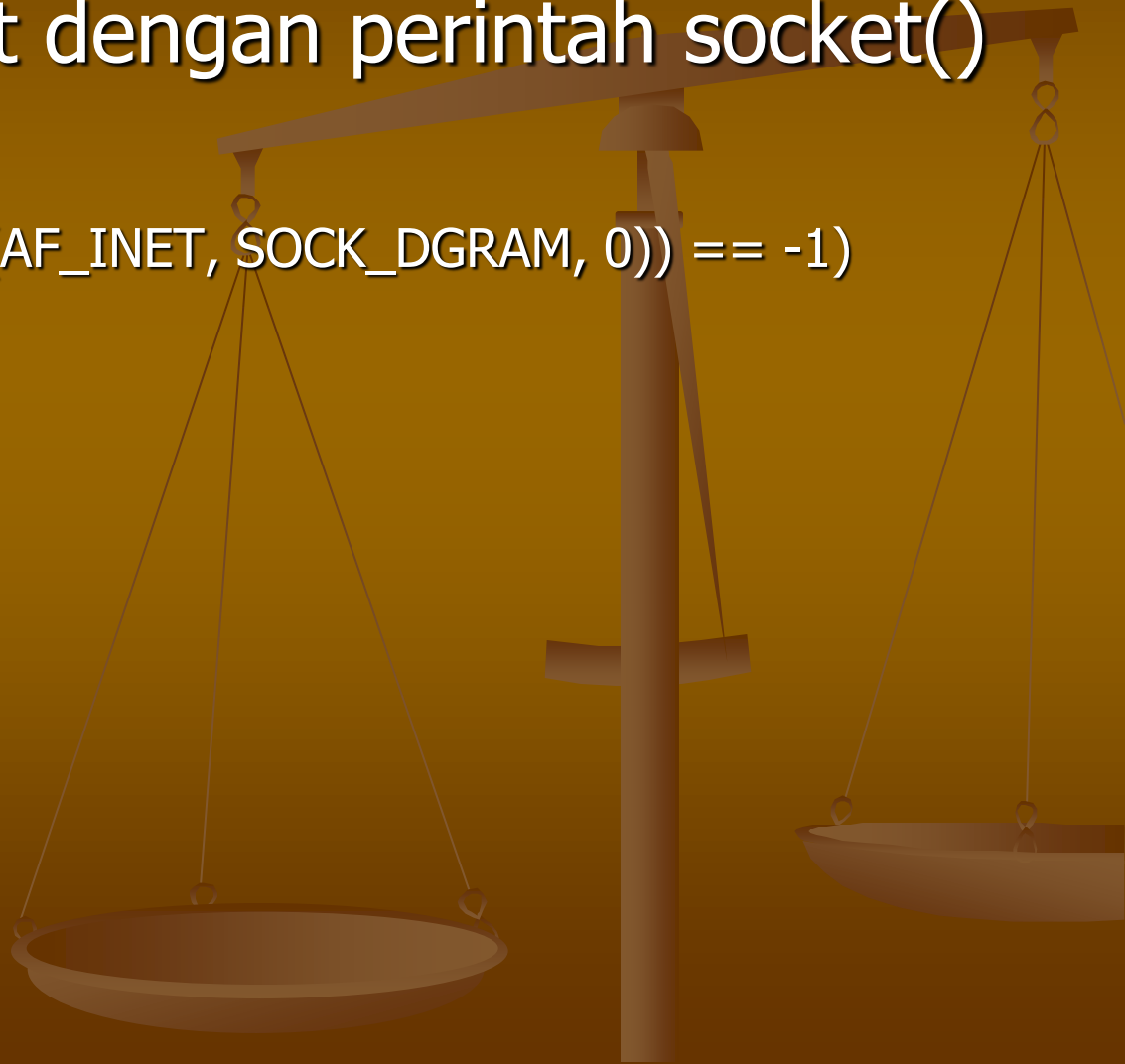
- Struktur pengalamatan yang dipakai antara stream socket dan datagram socket tidak ada perbedaan

```
struct sockaddr_in {
unsigned short sin_family; /* address family (always AF_INET)
   */
unsigned short sin_port; /* port num in network byte order */
struct in_addr sin_addr; /* IP addr in network byte order */
unsigned char sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

Procedure Client…

- Membuat socket dengan perintah socket()

```
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
{
    perror("socket");
   exit(1);
{
```



Procedure Client…

- Selanjutnya setelah membuat socket melakukan pengalamatan ke server

```
their_addr.sin_family = AF_INET;     // host byte order
        their_addr.sin_port = htons(MYPORT); // short, network byte order
        their_addr.sin_addr = *((struct in_addr *)he->h_addr);
        memset(&(their_addr.sin_zero), '\0', 8); // zero the rest of the struct
```

Procedure Client…

- Melakukan komunikasi (mengirim data), dengan menggunakan perintah *sendto()*

```
if ((numbytes=sendto(sockfd, argv[2], strlen(argv[2]), 0,
                (struct sockaddr *)&their_addr, sizeof(struct
    sockaddr))) == -1) {
     perror("sendto");
     exit(1);
}
```

- Menutup hubungan dengan perintah *close()*;

Procedure Server

- Melakukan prosedur pembukaan koneksi yang di dalamnya berupa langkah:
  - Membuat socket,

```
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
perror("socket");
exit(1); }
```

  - Mengikat socket

```
if (bind(sockfd, (struct sockaddr *)&my_addr,
   sizeof(struct sockaddr)) == -1) {
    perror("bind");
    exit(1);}
```

  - Pengalamatan socket

```
my_addr.sin_family = AF_INET;         // host byte order
my_addr.sin_port = htons(MYPORT);     // short, network byte order
my_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill with my IP
memset(&(my_addr.sin_zero), '\0', 8); // zero the rest of the struct
```

Procedure Server…

- Menerima koneksi dengan perintah *recvfrom()*

```
addr_len = sizeof(struct sockaddr);
if ((numbytes=recvfrom(sockfd,buf, MAXBUFLEN-1, 0,
                        (struct sockaddr *)&their_addr, &addr_len)) == -
   1) {
      perror("recvfrom");
      exit(1);
}
```