IPC: Sockets

• Point-to-point, bidirectional communication between two processes.
• End-point of a communication to which a name can be bound.
• Type and one or more associated processes.
• Domains (>23 families):
  UNIX
  INET
  Others (SNA, DECnet, APPLETALK, X25, IPX, ROUTE)
• Socket types:

  **Stream socket**  
  Bidirectional, sequenced, reliable, unduplicated flow. No record. \((SOCK\_STREAM)\). TCP in INET domain.

  **Datagram socket**  
  Bidirectional, record boundaries, not reliable, not sequenced. \((SOCK\_DGRAM)\). UDP in INET domain

  **Sequential packet socket**  
  Bidirectional, sequenced, reliable, connection, for datagrams with max length. \((SOCK\_SEQPACKET)\).

  **Raw socket**  
  For accessing underlying protocols
Creation and Naming

- `int socket(int domain, int type, int protocol)` is called to create a socket.

- A socket should be bound to an address for another process to identify it:
  ```c
  int bind(int s, const struct sockaddr *name, int name-len)
  ```
  - UNIX domain (creates a named socket on filesystem):
    ```c
    #include<sys/un.h>
    ...
    bind(sd, (struct socaddr_un *) &addr, length);
    ```
  - Internet domain:
    ```c
    #include<netinet/in.h>
    ...
    bind(sd, (struct socaddr_in *) &addr, length);
    ```
Connecting a socket

Usually not symmetric. A server should “listen” for connections.

Server:

```c
int listen(int s, int backlog)
```

Client:

```c
int connect(int s, struct sockaddr *name, int namelen)
or
int connect(int s, struct sockaddr_in *name, int namelen).
```

Server:

```c
int accept(int s, struct sockaddr *addr, int *addrlen)
```

returns a new socket for the current connection instance.
Data transfer

• Several functions:
  read(), write(), int send(int s, const char *msg, int len, int flags), recv(int s, char *buf, int len, int flags)

• send() and recv similar to read() and write() but have some flags.
  MSG_OOB     Out-of-band data
  MSG_DONTROUTE Only directed networks
  MSG_DONTWAIT Non-blocked mode
  MSG_NOSIGNAL No SIGPIPE
  MSG_PEEK     Read but not consume

• Closing a socket is just calling close() on file descriptor.
Datagram socket

- No connection required
- Each message carries destination address (Bind and send)
- `sendto()`, `sendmsg()`, `recvfrom()`, `recvmsg()`
- If `connect()` is used to specify a destination socket, `send()` and `recv()` can be used.