Unix Domain Protocols

when client and server are on the same host

- Unix domain socket address structure
- Socket functions
- Stream client-server
- Datagram client-server
- Passing descriptors
- Receiving sender credentials

Unix Domain Socket Address Structure

```
#include <sys/un.h>
struct sockaddr_un {
   uint8_t     sun_len;
   sa_family_t   sun_family;   /* AF_LOCAL */;
   char     sun_path[104];   /* null-terminated pathname */
};
```

Socket Functions

#include <sys/socket.h>

int socketpair (int family; int type, int protocol, int sockfd[2]);

returns: nonzero if OK, -1 on error

creates two sockets that are connected together

family: AF_LOCAL, protocol: 0, type: SOCK_STREAM or SOCK_DGRAM

- All socket functions for TCP and UDP sockets can be used, but several restrictions apply.

Passing Descriptors between Related/Unrelated Processes

- Create a Unix domain socket, either stream or datagram
- One process opens a descriptor
- The sending process builds a msghdr structure containing the descriptor to be passed, calls sendmsg
- The receiving process calls recvmsg

Receiving Sender Credentials through a Unix domain socket

```
Include <sys/ucred.h>
Struct fcred{
        uid_t     fc_ruid;                    /* real user ID */
        gid_t     fc_rgid;                    /* real group ID */
        char      fc_login[MAXLOGNAME];       /* setlogin() name */
        uid_t     fc_uid;                     /* effective user ID */
        fc_ngroups;                           /* number of group */
        gid_t     fc_groups[NGROUPS];         /* supplementary group IDs */
};
#define   fc_gid      fc_groups[0]       /* effective group ID */
```