

Sviluppo di software di rete con l'utilizzo di sistemi UNIX

Utilizzo dei socket: socket locali

Claudio Vicari

Definizioni

- ➔ i socket locali sono anche detti socket **Unix Domain Protocol**
- ➔ sono un metodo per comunicare all'interno di un singolo host, alternativo alle IPC
- ➔ possiamo usarli sia come socket stream che datagram, analogamente a TCP e UDP per la rete

Vantaggi

- ⇒ questo metodo di comunicazione si usa in modo simile agli altri socket
- ⇒ sono più veloci di TCP (almeno il doppio)
- ⇒ permettono di fare operazioni particolari:
 - passaggio di file descriptor fra processi differenti
 - comunicazione di credenziali per controlli sulla sicurezza
- ⇒ per esempio, i server X tipicamente usano i socket locali e anche quelli di rete. In questo modo, su una singola macchina si possono visualizzare applicazioni locali e remote. Le applicazioni locali possono beneficiare della maggior velocità

Indirizzamento nei socket locali

```
struct sockaddr_un {  
    uint8_t      sun_len;  
    sa_family_t  sun_family;    /* AF_LOCAL */  
    char          sun_path[104]; /* pathname */  
};
```

- ➡ sun_family deve essere AF_LOCAL (o anche AF_UNIX)
- ➡ sun_path deve essere terminata con un NULL. Si riferisce ad un vero file sul disco
- ➡ un indirizzo non specificato si esprime con sun_path[0]=0

Esempi

- ⇒ cerchiamo i socket locali nel sistema:
 - `find / -type s` trova tutti i socket
 - `file /tmp/.X11-unix` mostra il socket di X

unixbind.c

- ⇒ controllare che il socket sia stato creato!
- ⇒ notare che la lunghezza restituita è variabile a seconda del nome del file...

La funzione socketpair

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

- ➡ questa funzione crea una coppia di socket, fra loro connessi
- ➡ nella maggior parte delle implementazioni, supporta solo AF_UNIX
- ➡ il tipo può essere SOCK_STREAM (e si parla di *stream pipe*) oppure SOCK_DGRAM
- ➡ questi socket non hanno un nome identificativo: come se non si fosse fatto bind
- ➡ unica differenza rispetto alle pipe: sono full-duplex!

Note sparse

- ➔ i permessi di accesso di un socket, per default sono 777, modificati da umask
- ➔ il pathname deve essere specificato come percorso assoluto
- ➔ se la coda di ascolto è piena, chi fa connect riceve un errore immediatamente (diversamente da TCP)
- ➔ i socket Unix possono essere anche usati per passare file descriptor fra processi fra loro non correlati – ma è una tecnica che non esamineremo (lib/writefd.c e lib/readfd.c contengono una possibile implementazione)

Esempio di client e server

unixstrcli01.c

unixstrserv01.c

- ➔ possiamo lanciare anche client multipli
- ➔ il socket creato è visibile in /tmp/unix.str
- ➔ notare come le differenze rispetto ai socket di rete siano veramente minime!