

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

## *Utilizzo dei socket: socket options*

Claudio Vicari

# *Funzioni per socket options*

```
int getsockopt(int s, int level, int optname,  
               void *optval, socklen_t *optlen);  
int setsockopt(int s, int level, int optname,  
               const void *optval, socklen_t optlen);
```

- ➔ queste funzioni ci consentono di impostare una lunga serie di opzioni a disposizione per i socket
- ➔ alcune di queste opzioni dipendono puramente dal kernel, altre sono opzioni specifiche del protocollo
- ➔ *s* è un file descriptor di un socket aperto
- ➔ *level* specifica come interpretare l'opzione: generica o specifica di un qualche protocollo
- ➔ *optval* è una variabile che indica il valore dell'opzione
- ➔ *optlen* indica la lunghezza di questa variabile

# *Esempio: controllare le opzioni di default*

checkopts.c

- ➔ questo programma contiene definizioni per molte opzioni, e mostra a video i valori di default
- ➔ per esempio:
  - SO\_RCVBUF mostra il buffer di ricezione dei socket
  - IP\_TTL è il time to live dei pacchetti IP
  - TCP\_MAXSEG è il massimo numero di byte inviati in un singolo segmento TCP

# *Opzioni interessanti: generiche*

## ➡ SO\_KEEPALIVE:

- è valida nel caso di TCP. Se nessun dato viene scambiato per un tempo lungo (2 ore di default), TCP invia un pacchetto al puro scopo di controllare se l'altro peer è ancora vivo. Il peer risponderà con un ACK in caso affermativo, con un RST in caso negativo.
- questa opzione fornisce ai server una possibilità in più di continuare quando ci si blocca in attesa di dati da un client non più attivo

## ➡ SO\_LINGER:

- può essere usata per far sì che la close() ritorni solamente dopo avere inviato tutti i dati rimasti nei buffer di invio

# *Opzioni interessanti: generiche*

- ➡ SO\_TYPE:
  - il tipo del socket: SOCK\_DGRAM o SOCK\_STREAM
- ➡ SO\_RCVTIMEO, SO\_SNDTIMEO:
  - queste opzioni ci permettono di specificare il timeout per la ricezione e per la spedizione. Usano una struttura di tipo timeval, che permette di specificare secondi e microsecondi

# *Opzioni interessanti: generiche*

- ➡ SO\_REUSEADDR permette di usare una porta per bind, anche se c'è un altro processo che la usa come porta locale. Questo può accadere quando un processo sta ancora gestendo una richiesta su quella stessa porta.
- quindi, ci permette di non avere problemi in un caso di questo tipo: un server si mette in ascolto su una porta, poi arriva una richiesta, crea un child, e termina senza che il child abbia finito. In questo caso, non potremmo far ripartire il server
- possiamo far partire due processi server, su due indirizzi IP ma con la stessa porta – non possiamo comunque avere stesso IP e stessa porta
- quindi, **questa opzione è consigliata** nei server

# *Opzioni interessanti: IP*

## ⇒ IP\_OPTIONS

- consente di impostare opzioni dei pacchetti IP – e richiede una conoscenza dettagliata del protocollo
- per esempio, per impostare le possibilità di frammentazione dei pacchetti IP

## ⇒ IP\_TOS

- permette di impostare il campo TOS, una sorta di qualità del servizio implementata in IP – comunemente ignorata dai router!

## ⇒ IP\_TTL

- per impostare il time to live (massimo numero di hop da percorrere) dei pacchetti spediti sulla rete

# *Opzioni interessanti: TCP*

## ➡ TCP\_KEEPALIVE

- non su tutti i sistemi. Specifica un tempo di inattività in secondi, prima che TCP inizi a mandare pacchetti di probe (similmente a SO\_KEEPALIVE, che deve comunque essere attiva)

## ➡ TCP\_NODELAY

- blocca l'uso dell'algoritmo *Nagle* di TCP, che impedisce l'invio di pacchetti di piccole dimensioni. Per esempio, in telnet normalmente si invia un pacchetto per ogni lettera digitata! In reti con un RTT alto, *Nagle* rallenta il tasso d'invio, e telnet sembrerebbe molto rallentato...



# *Opzioni interessanti: ancora sulle write non confermate*

tsigpipe-so\_linger.c