Laboratorio di Programmazione in Rete

a.a. 2005/2006

http://www.di.uniba.it/~lisi/courses/prog-rete/prog-rete0506.htm

dott.ssa Francesca A. Lisi
lisi@di.uniba.it

Orario di ricevimento: mercoledì ore 10-12

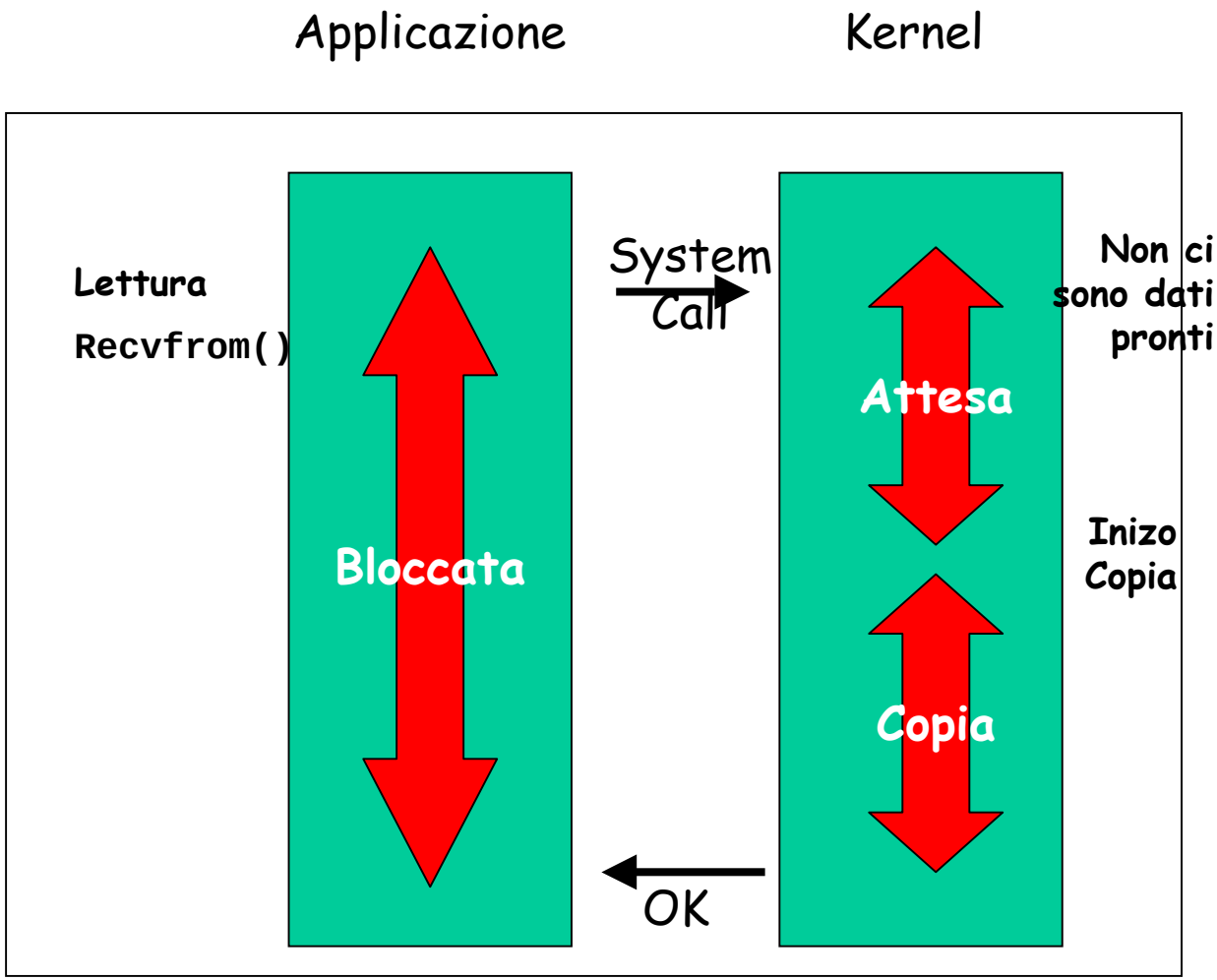
N.B. Il presente materiale didattico è stato
prodotto da: dott.ssa V. **Carofiglio**
rielaborato da: dott.ssa F.A. **Lisi**

dib

Socket non bloccanti: modelli di I/O

Per default una socket e' bloccante:

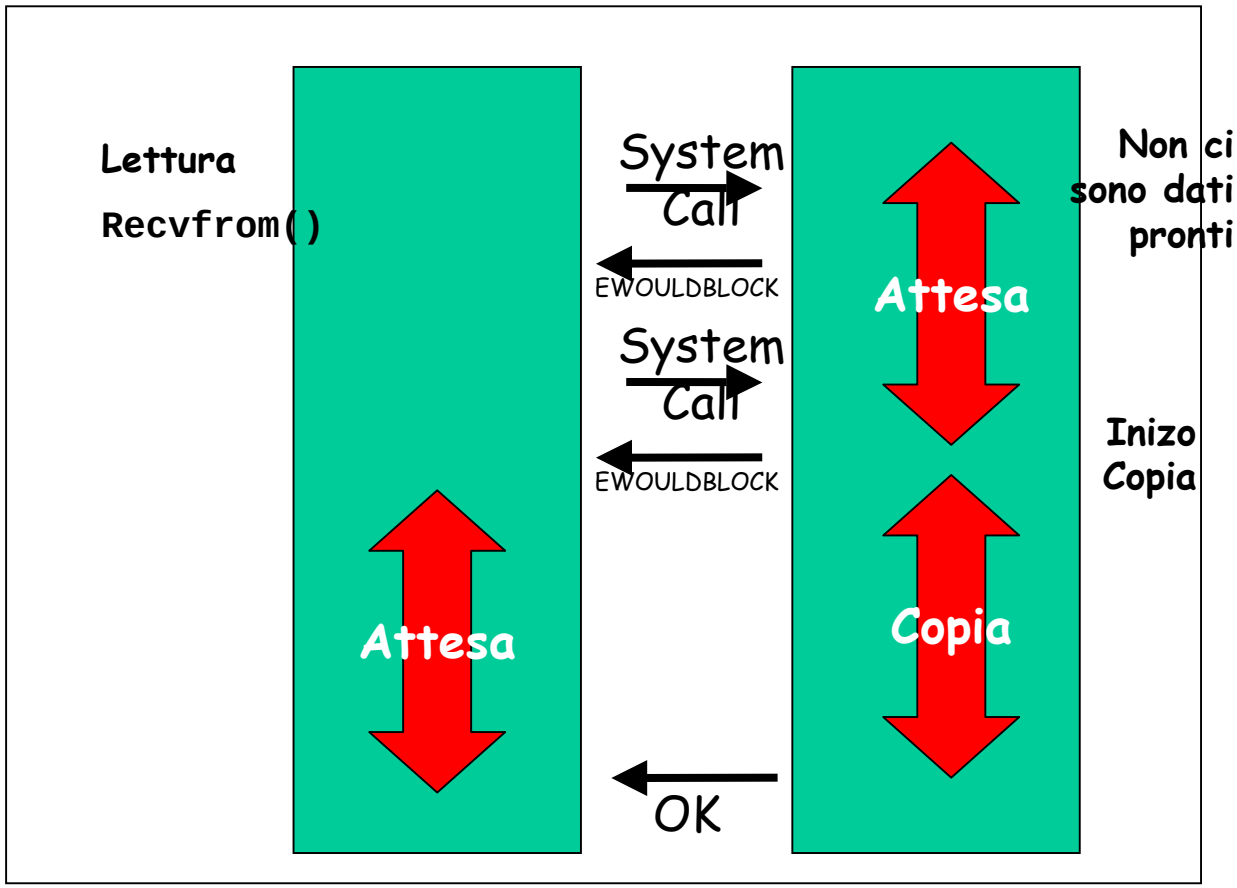
quando la socket deve effettuare un'operazione di I/O,
se questa non puo' essere terminata immediatamente
il processo si mette in attesa aspettando la fine dell'operazione.



Socket non bloccanti: modelli di I/O (cont.)

Input con una socket Non Bloccante:

se l'operazione di input non puo' essere terminata
(non c'e' nemmeno un byte per la socket TCP o nemmeno un datagram per la socket UDP)
la funzione ritorna immediatamente al chiamante restituendo il codice d'errore EWOULDBLOCK (polling).



Socket non bloccanti: modelli di I/O (cont.)

Output in una socket Non Bloccante:

• Con una **socket Non Bloccante di tipo TCP**, se l'operazione di scrittura non puo' essere effettuata nemmeno in parte perche' manca spazio nei buffer del TCP in cui andare a scrivere i dati, la funzione ritorna immediatamente al chiamante restituendo il codice d'errore EWOULDBLOCK.

Se invece e' rimasto spazio, **viene effettuata una scrittura di una porzione di dati minore o uguale alla dimensione del buffer libero**, e la `write` restituisce il numero di byte scritti.

• Con una **socket Non Bloccante di tipo UDP** invece il problema non sussiste perchè nessun datagram viene mantenuto in un buffer in quanto non c'è ritrasmissione, quindi il datagram UDP viene immediatamente spedito e vada come vada. Quindi una primitiva di output di una socket UDP non blocca mai il processo che l'effettua.

Socket non bloccanti: modelli di I/O (cont.)

Cambiare il comportamento delle socket per rendere le chiamate non bloccanti:

- Operazioni richieste possono essere completate immediatamente:
  - il valore di ritorno della funzione indica successo
- altrimenti:
  - il valore di ritorno della funzione indica fallimento (-1) (attenzione a distinguere)

Winsock fa uso della funzione `ioctlsocket()` per controllare l'I/O di una socket

Unix fa uso della funzione `fcntl()` per controllare l'I/O di una socket

LE DUE FUNZIONI NON SONO EQUIVALENTI

La funzione fcntl()

Consente (tra l'altro) di settare i flag per il controllo del bloccaggio di una socket valida

```
int fcntl (int socket, int command,  long argument);
```

descrittore di socket

Operazione da effettuare (F_GETFL e F_SETFL)

Specifica dei flag (O_NONBLOCK)

la procedura corretta per settare un flag consiste nel:

- leggere i flag del descrittore mediante il comando F_GETFL,
- modificare solo il flag che interessa,
- settare i nuovi flag per il socket mediante il comando F_SETFL

La funzione `ioctlsocket()`

Consente (tra l'altro) di settare i flag per il controllo del bloccaggio di una socket valida

```
int ioctlsocket (int socket, int command,  u_long *argument);
```

descrittore di socket

Operazione da effettuare (FIONBIO FIONREAD ….)

Non-0 per abilitare la modalità non bloccante

FIONBIO: I/O bloccante o no

FIONREAD: per determinare l'ammontare di dati in pendenza nel buffer che puo' essere letto dalla socket s

…

Socket non bloccanti: esempio di I/O non bloccante

```
//-------------------------
// Initialize Winsock WSADATA wsaData;
int iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
if (iResult != NO_ERROR)
printf("Error at WSAStartup()\n");
//-------------------------
// Create a SOCKET object. SOCKET m_socket;
m_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (m_socket == INVALID_SOCKET)
{ printf("Error at socket(): %ld\n", WSAGetLastError());
WSACleanup();
 return; }
//-------------------------
// Set the socket I/O mode: In this case FIONBIO
// enables or disables the blocking mode for the
// socket based on the numerical value of iMode.
// If iMode = 0, blocking is enabled;
// If iMode != 0, non-blocking mode is enabled.
u_long iMode = 1;
ioctlsocket(m_socket, FIONBIO, &iMode);
```

Socket non bloccanti: richiesta di connessione

Operazioni di Richiesta Connessione (connect).

Con **una socket UDP** la funzione `connect()` non e' realmente bloccante, perche' deve solo scrivere nel kernel l'indirizzo IP e il numero di porta dell'altro end-system, quindi anche nel caso di socket Non Bloccante la connect per l'UDP effettua sempre l'operazione interamente e restituisce il controllo al chiamante.

Invece **per il TCP** la chiamata del client alla `connect()` di una socket bloccante deve aspettare che, all'interno del *three way handshake*, il client abbia ricevuto il primo ACK. Solo allora puo' veramente bloccare il processo.

Se la socket TCP e' di tipo Non Bloccante, la chiamata del client alla `connect()` fa iniziare il procedimento di inizio connessione, cioe' fa spedire il primo ACK, ma se la connessione non puo' immediatamente essere instaurata la `connect()` termina con un codice d'errore **EINPROGRESS** ad indicare che l'operazione continua a livello inferiore.

Socket non bloccanti: accettazione di una richiesta di connessione

Operazioni di Accettazione Richiesta Connessione (accept).
Con una socket Non Bloccante, se l'operazione di accept non puo' restituire immediatamente una nuova connessione perche' la coda delle connessioni accettate e' vuota, la funzione ritorna immediatamente al chiamante restituendo il codice d'errore EWOULDBLOCK.

- Per i modelli di I/O, vedi par. 5.3, in particolare par. 5.3.1 di Donahoo & Calvert