

Laboratorio reti AA 2008/2009

Dott. Matteo Roffilli

roffilli@csr.unibo.it

**Ricevimento in ufficio
dopo la lezione**

Laboratorio reti AA 2008/2009

Per esercitarvi fate SSH su:

alfa.csr.unibo.it

si-tux00.csr.unibo.it

....

si-tux15.csr.unibo.it

Eventuali variazioni di orario/giorno verranno comunicate in anticipo via mail.

Laboratorio reti AA 2008/2009

- **Marzo**

- 5 Intro,SSH,VI/VIM,GCC base
- 12 Richiami di C e Compilazione
- 19 Socket e Co.
- 26 Socket e Co. parte seconda

- **Aprile**

- **2 Utilizzo di IPC: Client**

- Si riprende il 16 Aprile. Controllate sempre il portale di CSR.

Inter Process Communication

La comunicazione tra processi può essere effettuata in molti modi.

Per ora abbiamo visto le **socket** ma si possono usare anche:

- **File**
- **Signal**
- **Pipe**
- **Named pipe, Semaphore, Shared memory, Message queue, Mailbox**

Le socket si appoggiano al meccanismo dei file descriptor: “Nei sistemi operativi Unix e Unix-like un descrittore di file (o file descriptor) è un numero intero non negativo che rappresenta un file, una pipe o un socket aperto da un processo e sul quale il processo può effettuare operazioni di input/output.” [Wikipedia]

In particolare si possono usare le funzioni appositamente sviluppate per gestire l'I/O su file descriptor: **read** e **write**

Write

```
#include <unistd.h>
```

```
ssize_t write(  
    int fd,  
    void* buf,  
    size_t count)
```

Richiede di scrivere count byte dal buffer buf sul file fd.

La funzione ritorna il numero di byte scritti in caso di successo e -1 in caso di errore,
nel qual caso errno assumerà uno dei valori:

EINVAL fd è connesso ad un oggetto che non consente la scrittura.

EFBIG si è cercato di scrivere oltre la dimensione massima consentita dal filesystem o il limite per le dimensioni dei file del processo o su una posizione oltre il massimo consentito.

EPIPE fd è connesso ad una pipe il cui altro capo è chiuso in lettura; in questo caso viene anche generato il segnale SIGPIPE, se questo viene gestito (o bloccato o ignorato) la funzione ritorna questo errore.

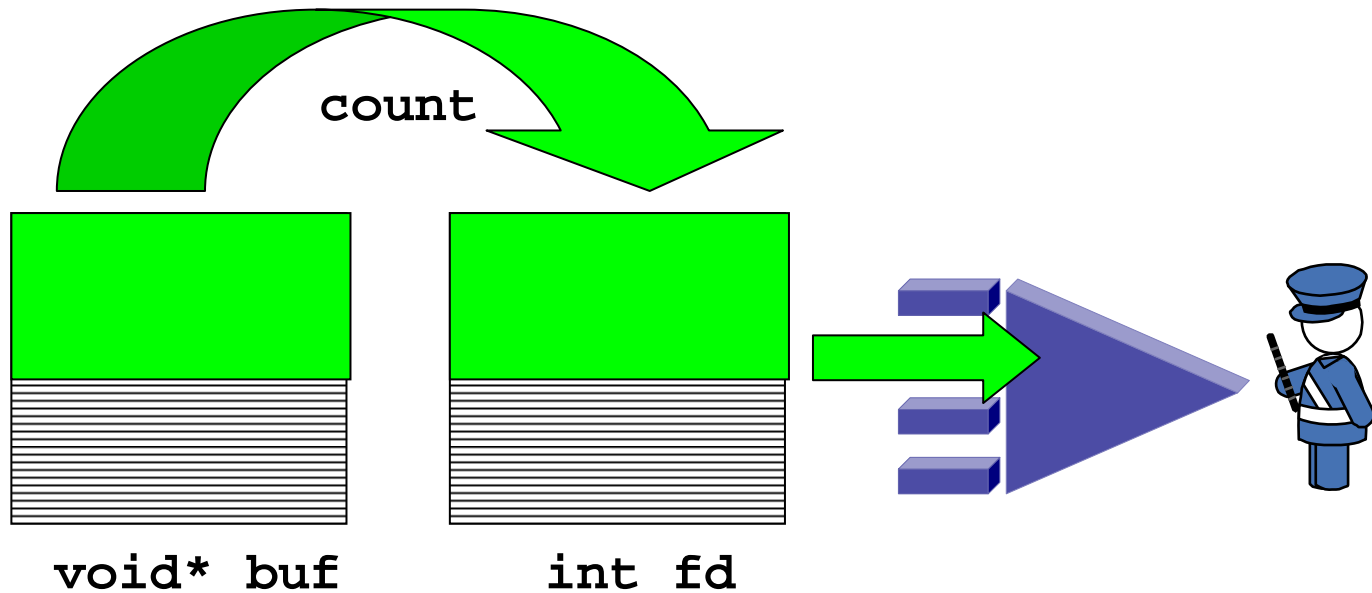
EINTR si è stati interrotti da un segnale prima di aver potuto scrivere qualsiasi dato.

EAGAIN ci si sarebbe bloccati, ma il file era aperto in modalità O_NONBLOCK.

Write

```
#include <unistd.h>
```

```
ssize_t write(  
    int fd,  
    void* buf,  
    size_t count)
```



Read

```
#include <unistd.h>
```

```
ssize_t read(  
    int fd,  
    void* buf,  
    size_t count)
```

Richiede di leggere count byte dal file fd al buffer buf.

La funzione ritorna il numero di byte letti in caso di successo e -1 in caso di errore,
nel qual caso errno assumerà uno dei valori:

EINTR la funzione è stata interrotta da un segnale prima di aver potuto leggere qualsiasi dato.

EAGAIN la funzione non aveva nessun dato da restituire e si era aperto il file in modalità
O_NONBLOCK.

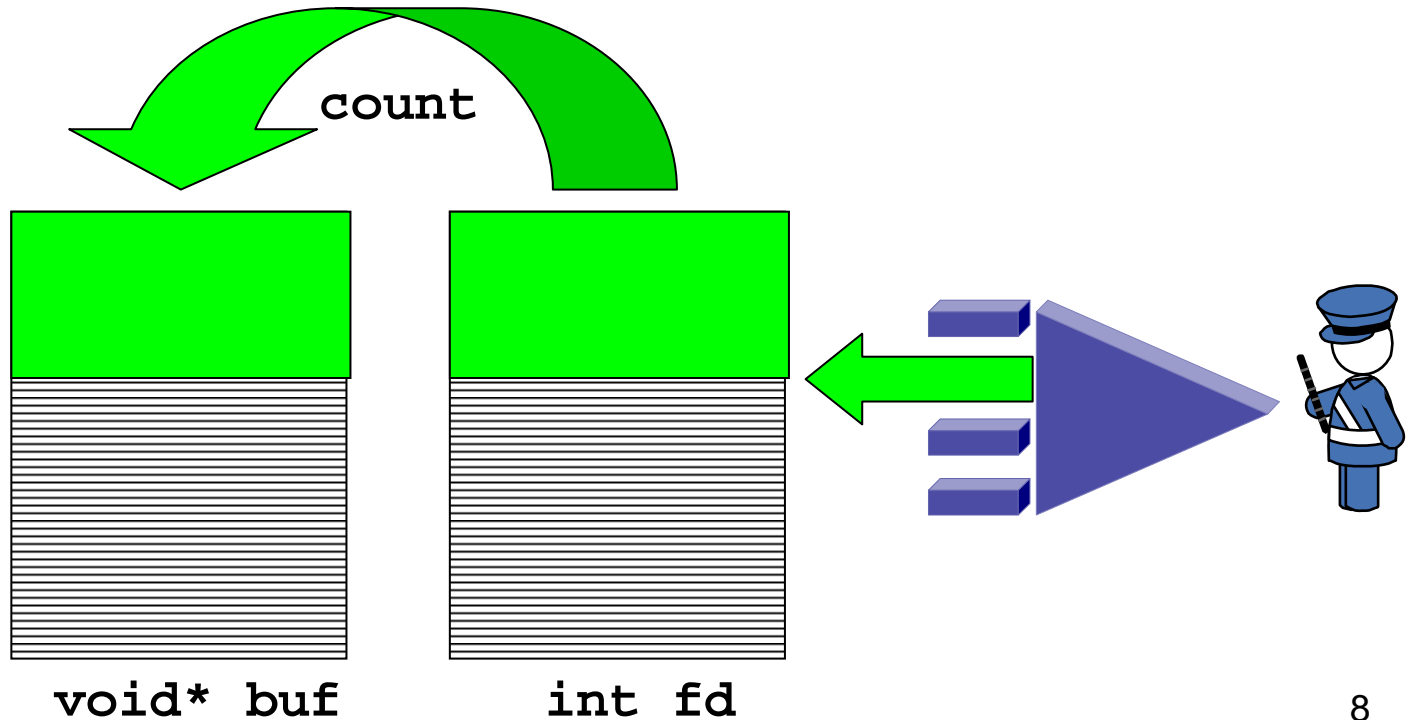
ed inoltre EBADF, EIO, EISDIR, EBADF, EINVAL e EFAULT ed eventuali altri errori
dipendenti dalla natura dell'oggetto connesso a fd.

Se viene restituito 0 (zero) significa end-of-file (fine stream), ovvero significa che l'altro end system
ha volutamente chiuso la connessione e quindi il socket non potrà essere più utilizzato per
leggere.

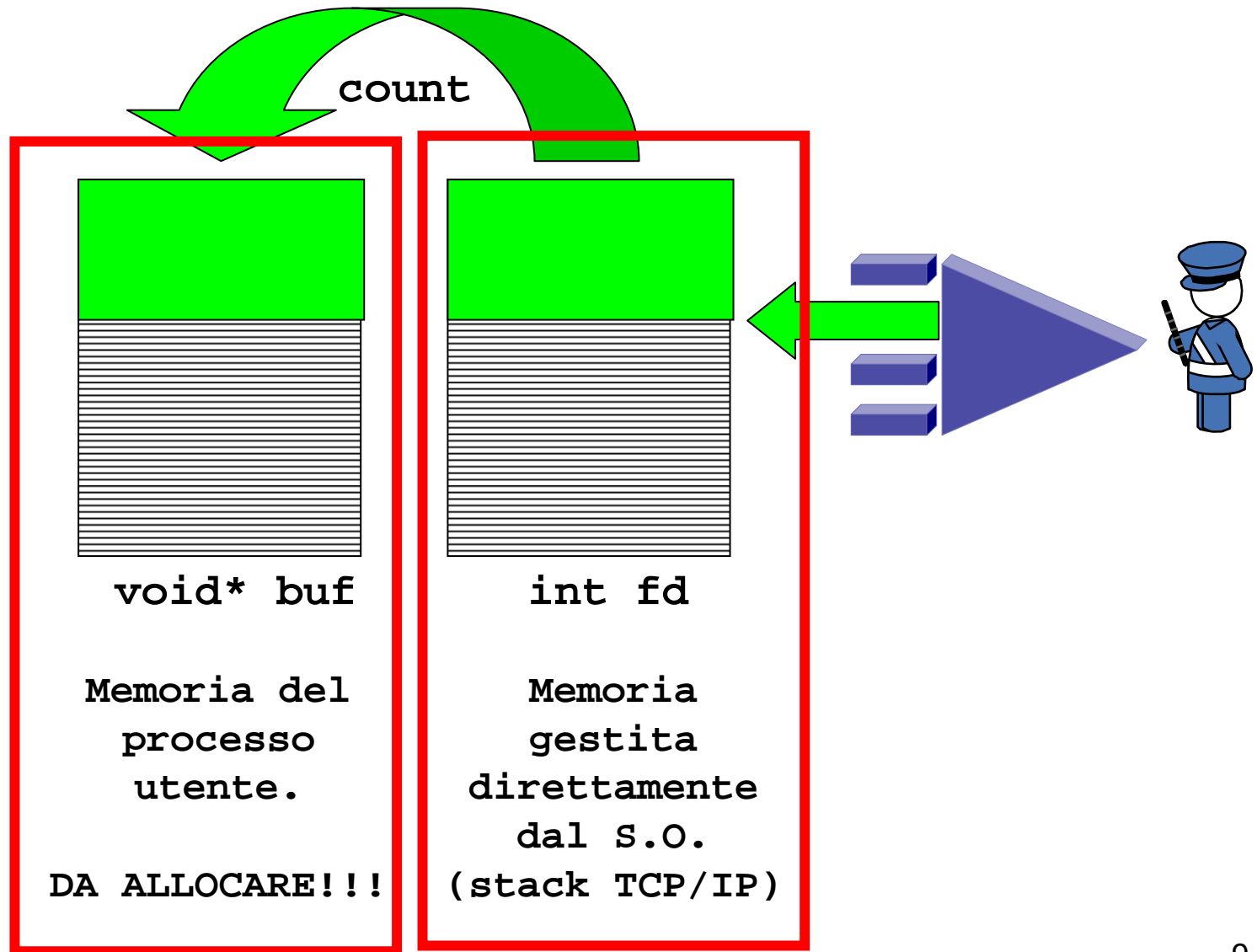
Read

```
#include <unistd.h>
```

```
ssize_t write(  
    int fd,  
    void* buf,  
    size_t count)
```



Memoria in Read/Write



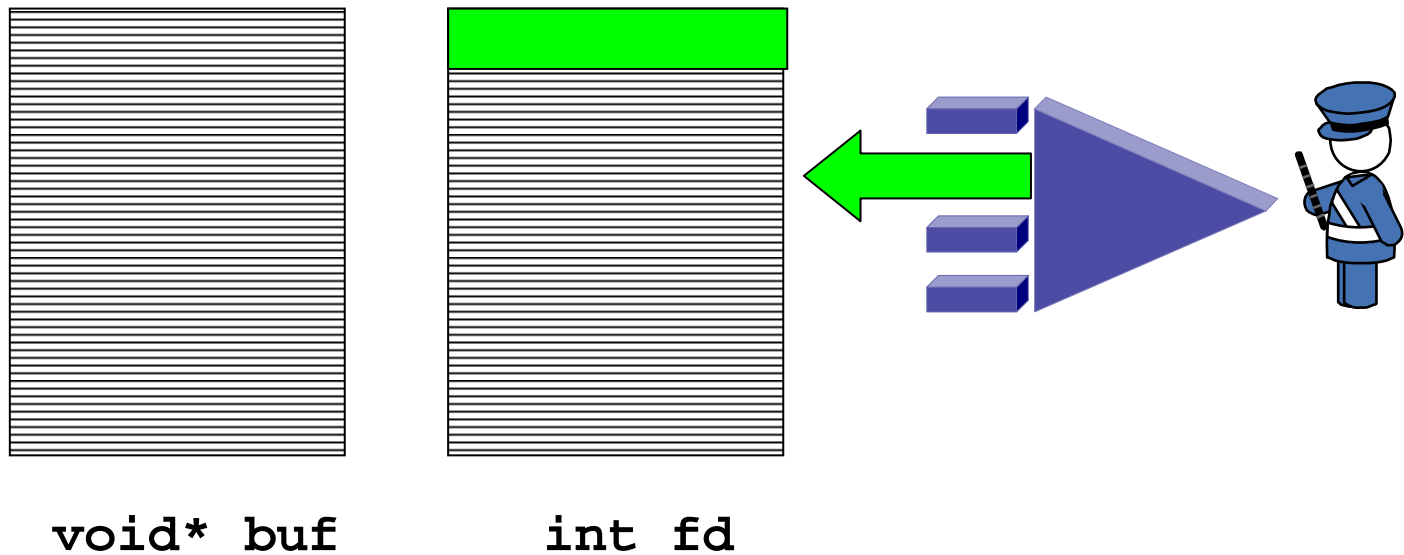
write/read

```
while( (ho scritto/letto tutti i caratteri prefissati) &&
      (la write/read scrive/legge correttamente nel buffer) )
{
    incrementa la posizione di scrittura/lettura nel buffer;
}

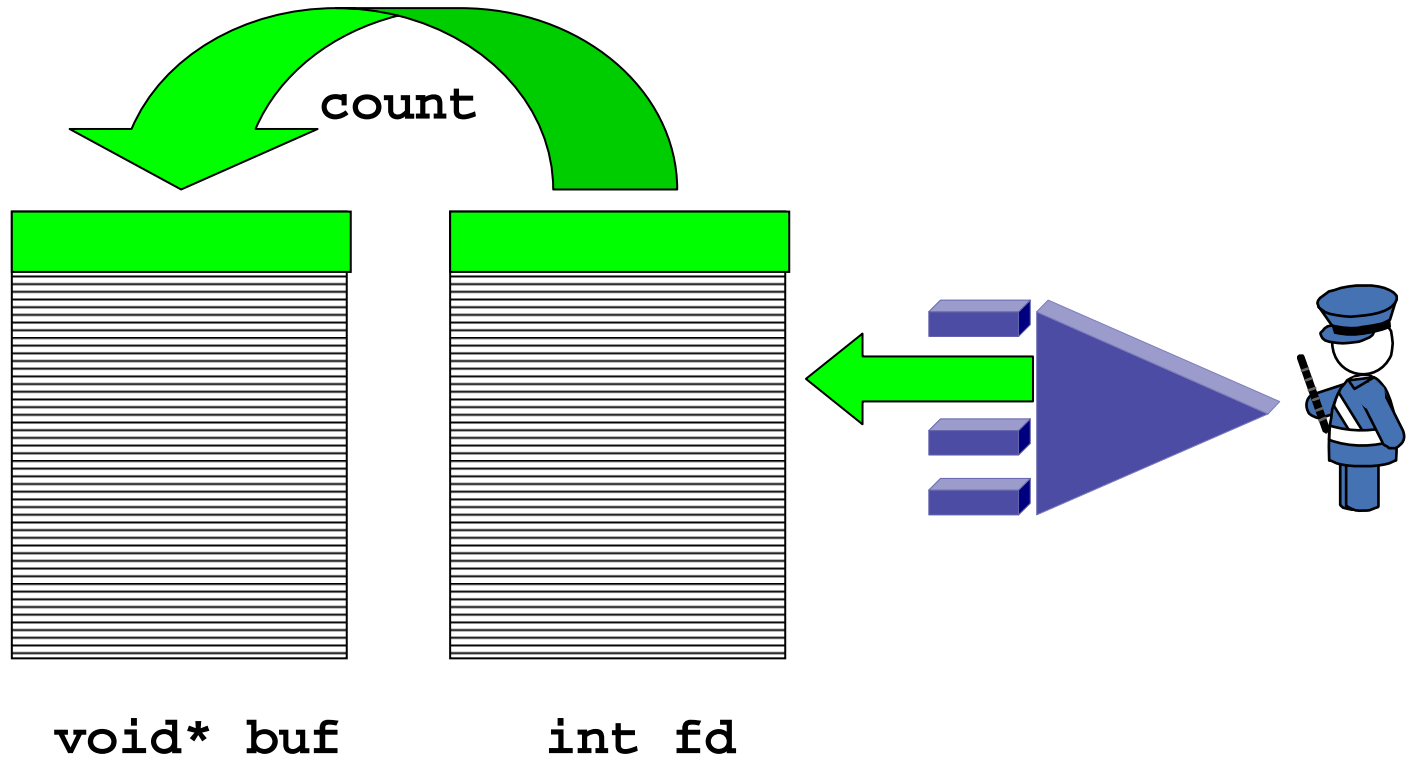
/* write */
len = number_of_char_to_write;
nwrite=0;
while((nwrite<len) && ((n=write(socketfd,&(msg[nwrite]),len-write))>0))
{
    nwrite+=n;
}

/* read */
nread=0;
len = number_of_char_to_read;
while((nread<len) && ((n=read(socketfd,&(buf[nread]),len-nread))>0))
{
    nread+=n;
}
```

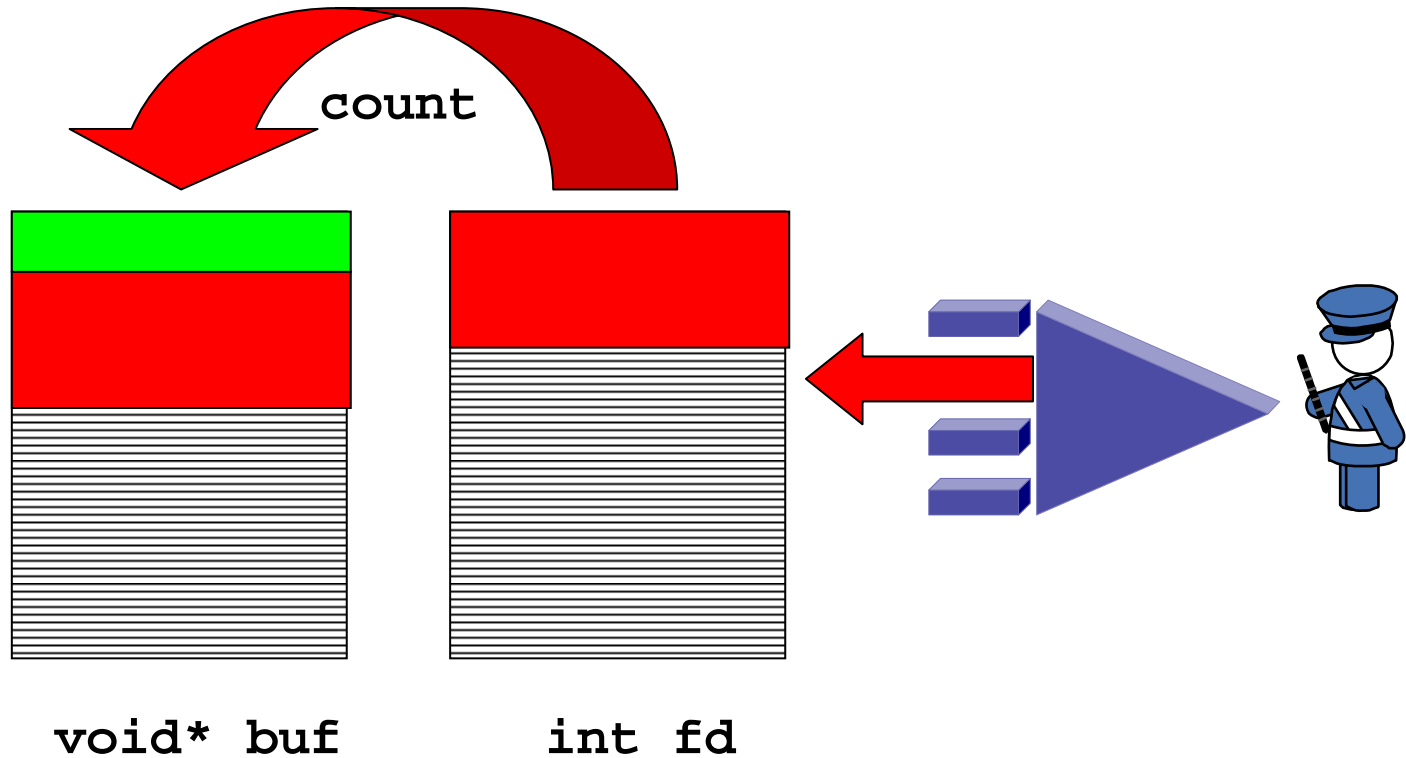
Read sequenza #0



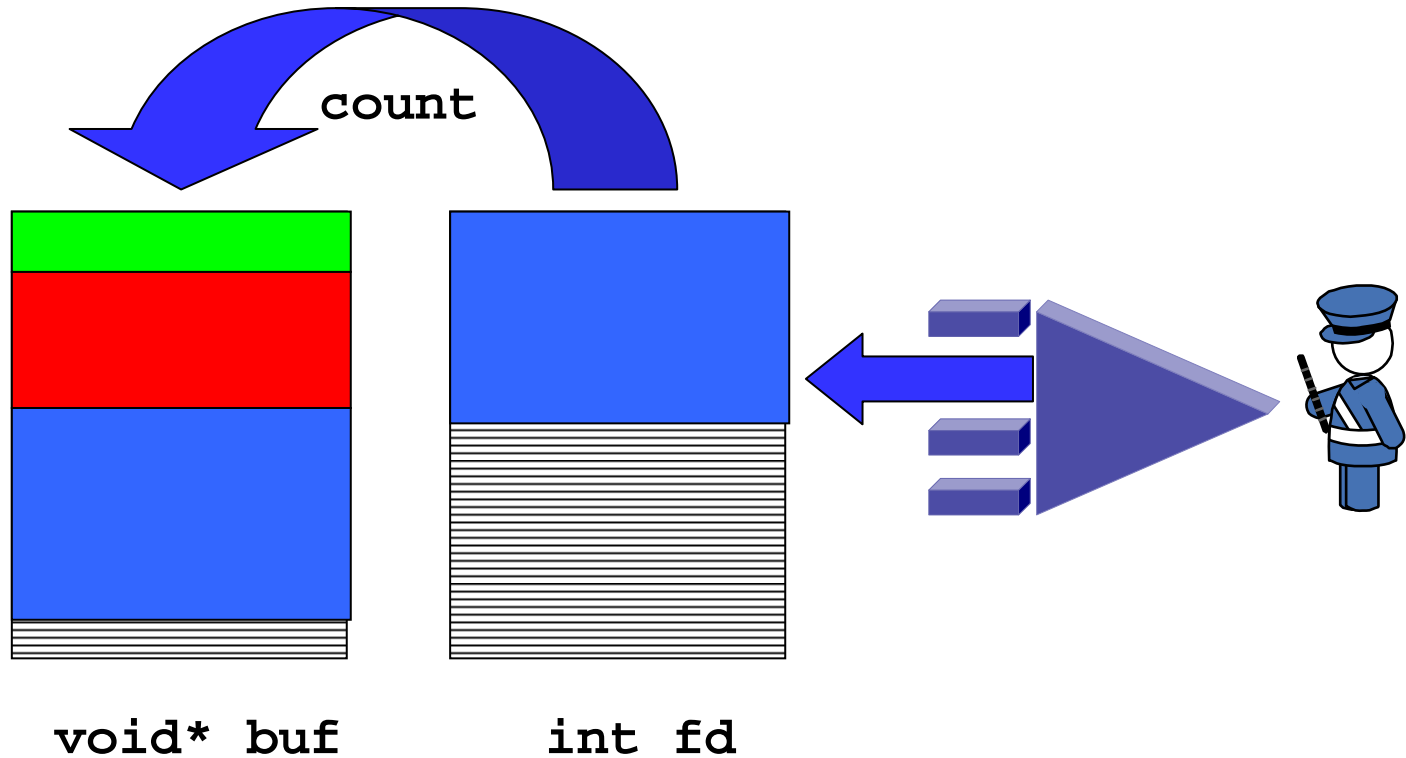
Read sequenza #1 start



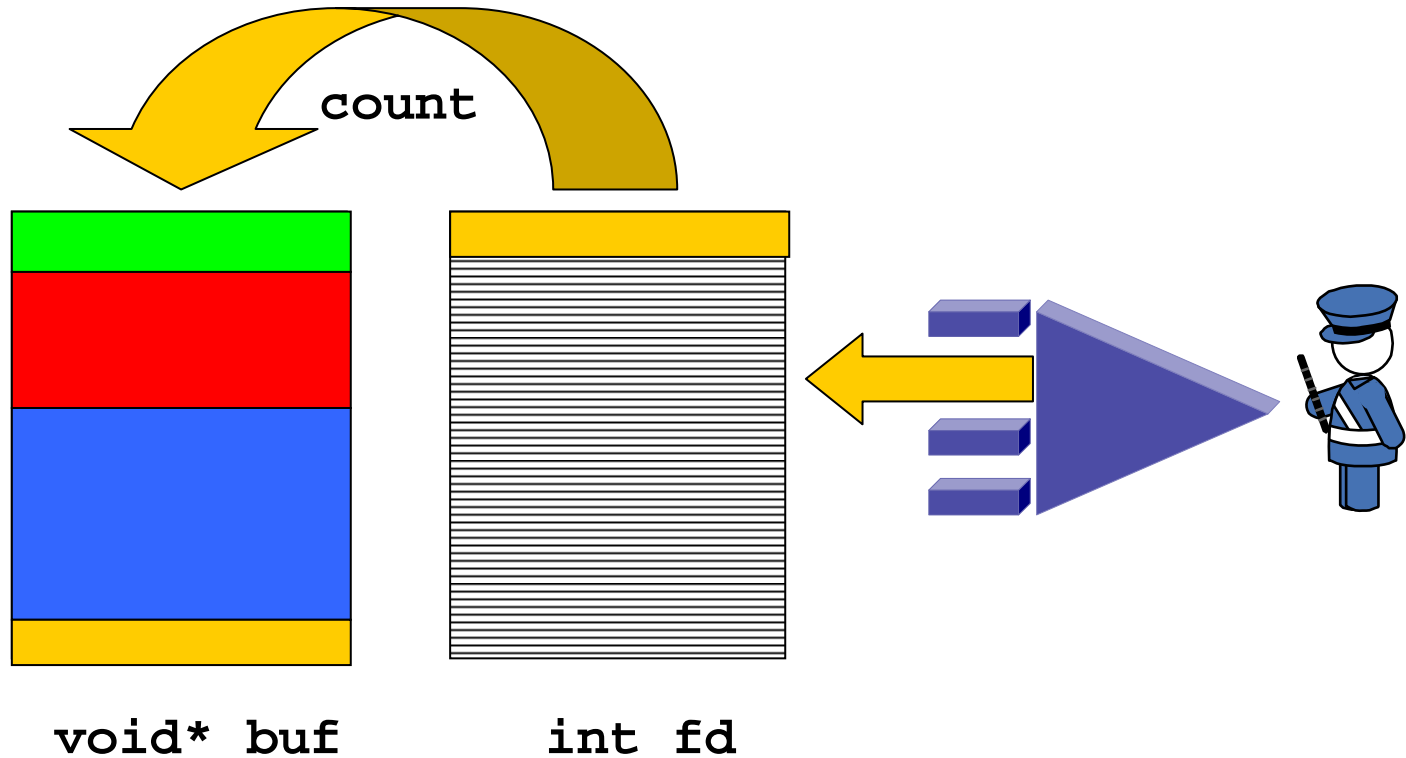
Read sequenza #2



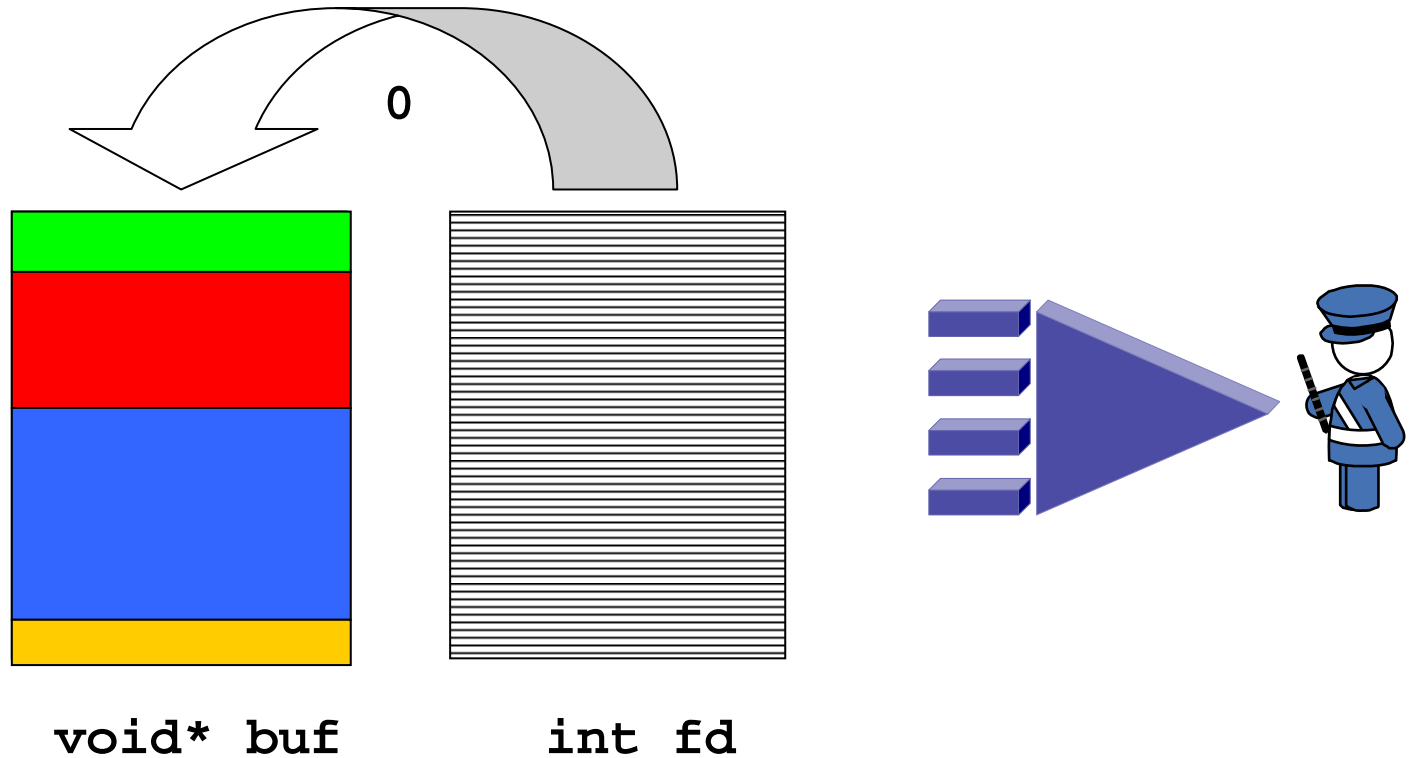
Read sequenza #3



Read sequenza #4



Read sequenza #5 end



Esercizio: il primo client

Goal:

- Scrivere un client utilizzando le informazioni dei lab. precedenti
- Controllare gli errori
- Accettare indirizzi testuali
- Per testarlo scaricare il server [ELF-server] e lanciarlo da un'altra shell (per renderlo eseguibile: `chmod 777 server`)

Requisiti:

1. Testare il client su **si-tux00.csr.unibo.it** alla porta **8888**
2. Cercare di capire cosa fa il server.

Tempo a disposizione:

60 minuti

Soluzione base senza controlli

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>
#define MAXSIZE 100

int main(int argc, char *argv[])
{
    struct sockaddr_in Serv;
    char string_remote_ip_address[100];
    short int remote_port_number, local_port_number;
    int sockfd, msglen, ris;
    int n, i, nread, nwrite, len;
    char buf[MAXSIZE], msg[MAXSIZE];    // ="012345ABCD";

    for(i=0;i<MAXSIZE;i++) msg[i]='a';
    msg[MAXSIZE-1]='\0';

    /* set remote host */
    strncpy(string_remote_ip_address, argv[1], 99);
    remote_port_number = atoi(argv[2]);

    /* assign our destination address */
    memset ( &Serv, 0, sizeof(Serv) );
    Serv.sin_family      =      AF_INET;
    Serv.sin_addr.s_addr =      inet_addr(string_remote_ip_address);
    Serv.sin_port        =      htons(remote_port_number);
```

Soluzione base senza controlli

```
/* get a datagram socket */
socketfd = socket(PF_INET, SOCK_STREAM, 0);

/* connection request */
ris = connect(socketfd, (struct sockaddr*) &Serv, sizeof(Serv));

/* scrittura */
len = strlen(msg)+1;
nwrite=0;
while( (n=write(socketfd, &(msg[nwrite]), len-nwrite)) >0 )
    nwrite+=n;

/* lettura */
nread=0;
while( (len>nread) && ((n=read(socketfd, &(buf[nread]), len-nread )) >0))
{
    nread+=n;
    printf("read effettuata, risultato n=%d  len=%d nread=%d len-
nread=%d\n", n, len, nread, len-nread );
    fflush(stdout);
}

/* stampa risultato */
printf("\nstringa ricevuta: %s\n", buf);fflush(stdout);

/* chiusura */
close(socketfd);

return(0);
}
```