

Laboratorio reti AA 2007/2008

Dott. Matteo Roffilli

roffilli@csr.unibo.it

**Ricevimento in ufficio
dopo la lezione**

Laboratorio reti AA 2007/2008

Per esercitarvi fate SSH su:

alfa.csr.unibo.it

si-tux00.csr.unibo.it

....

si-tux15.csr.unibo.it

Lo username dovrebbe essere del tipo:

STUDENTI/matteo.roffilli

STUDENTI\matteo.roffilli

Se avete problemi di login contattate i tecnici

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

Laboratorio reti AA 2007/2008

- **Marzo**
- 13 Intro,SSH,VI/VIM,GCC base
- 19 Richiami di C e Compilazione
- **Aprile**
- 3 Socket e Co.
- 10 Socket e Co. parte seconda
- **17 Client**
- **Mercoledì' 23/4 11:00-13:00 Client parte II**

Write

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

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

Scrive 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:

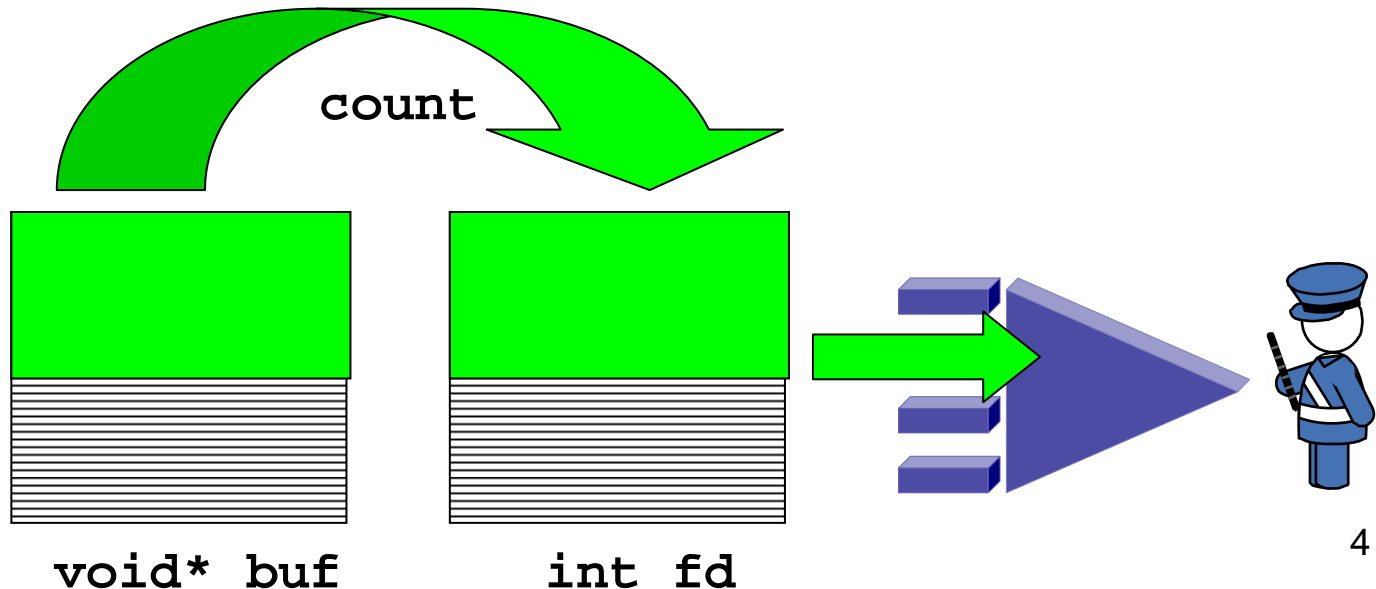
EINVAL fd è connesso ad un oggetto che non consente la scrittura.

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.



Read

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

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

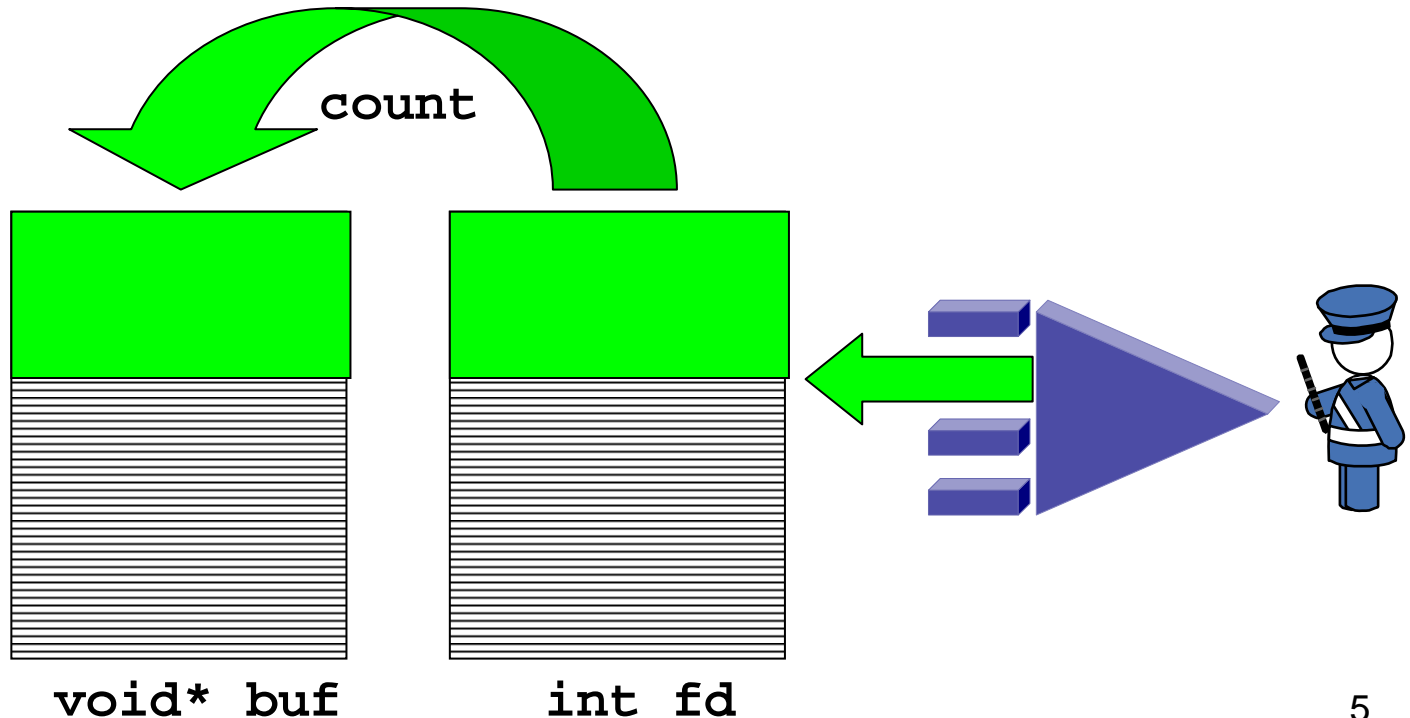
Cerca di leggere count byte dal file fd al buffer buf.

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:

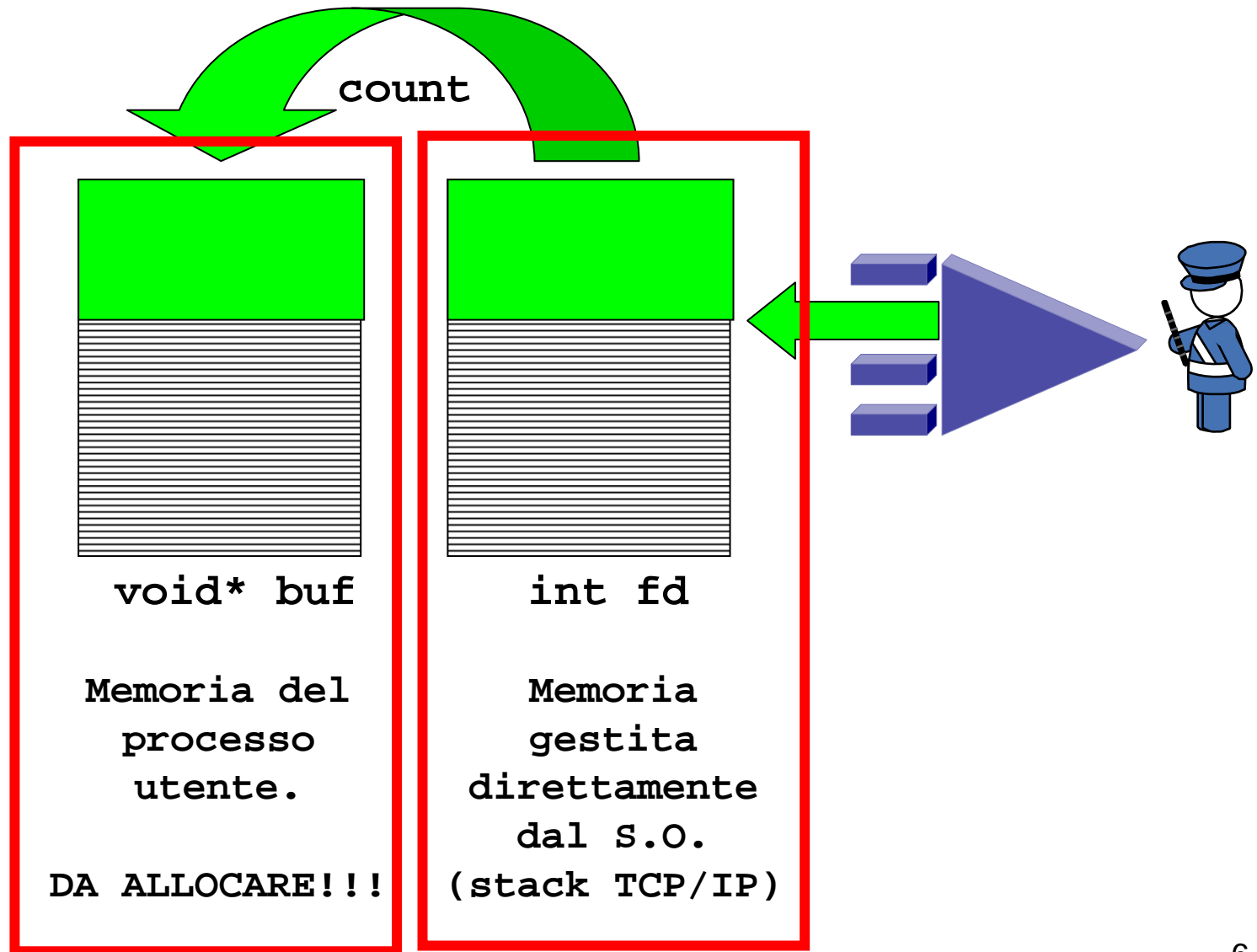
EINTR la funzione è stata interrotta da un segnale prima di aver potuto leggere qualsiasi dato.

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.



Memoria in Read/Write



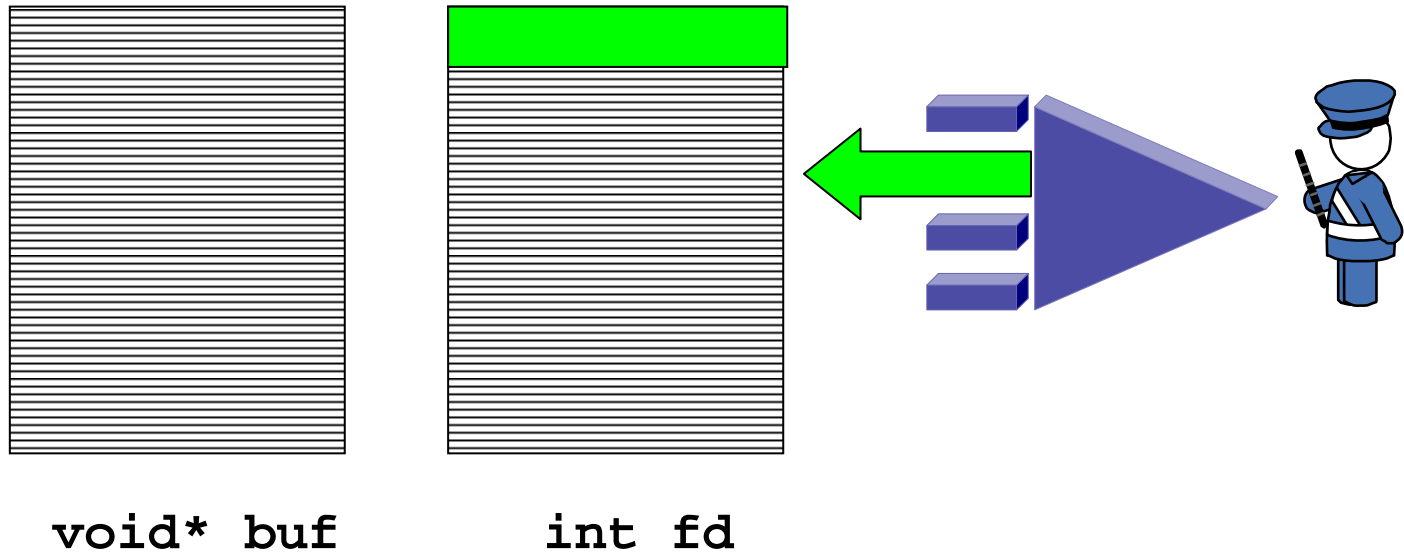
write/read

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

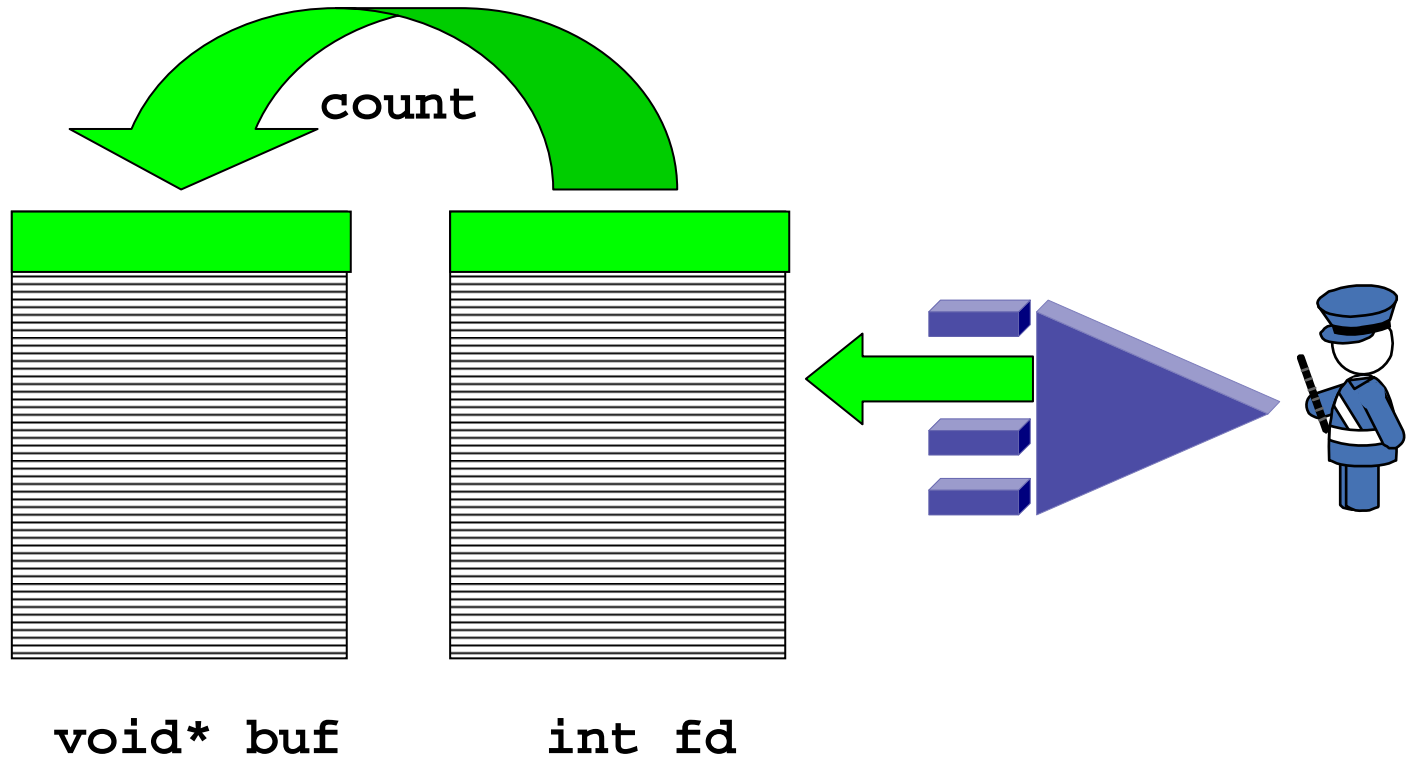
/* write */
len = number_of_char_to_write;
nwrite=0;
while((nwrite<len) && ((n=write(socketfd,&(msg[nwrite]),len-write))>0))
{
    nwrite+=n;
}

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

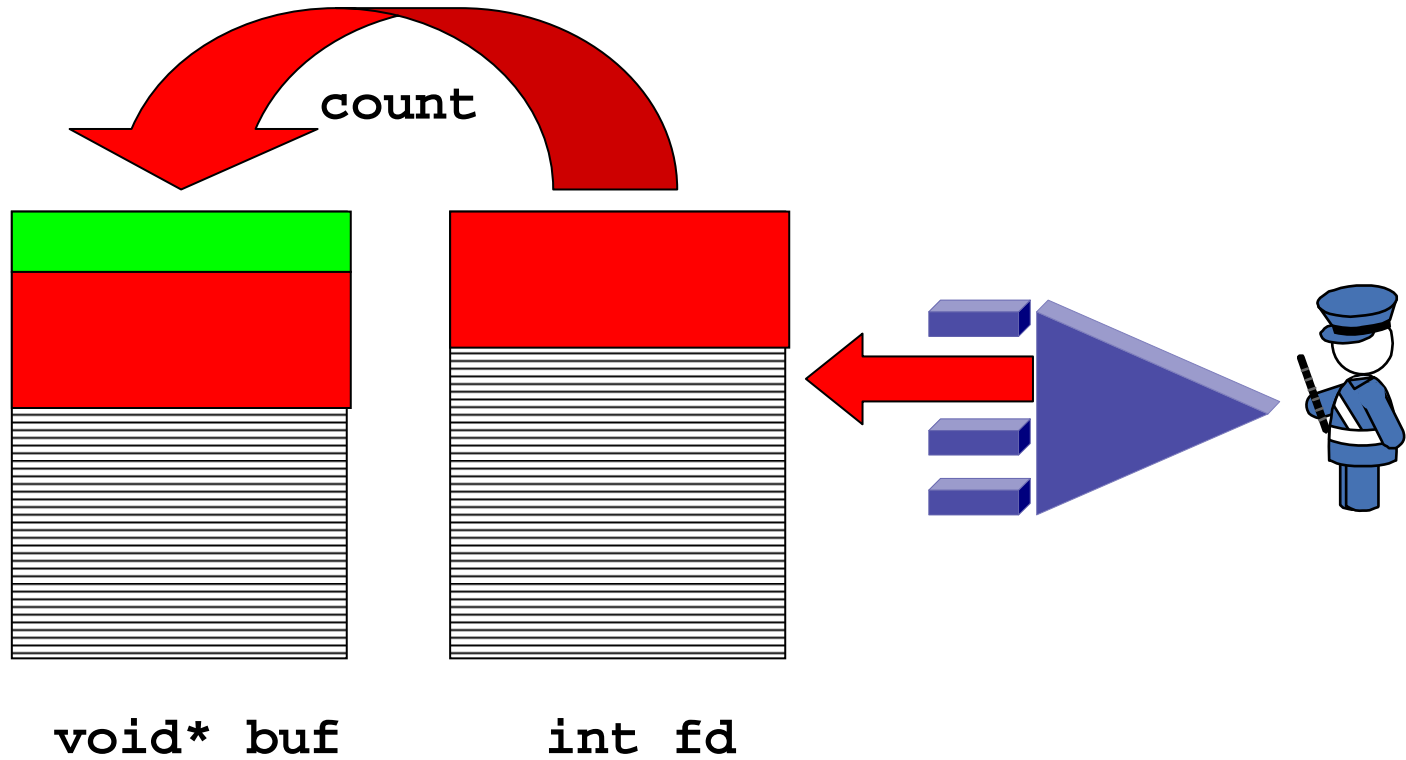
Read sequenza #0



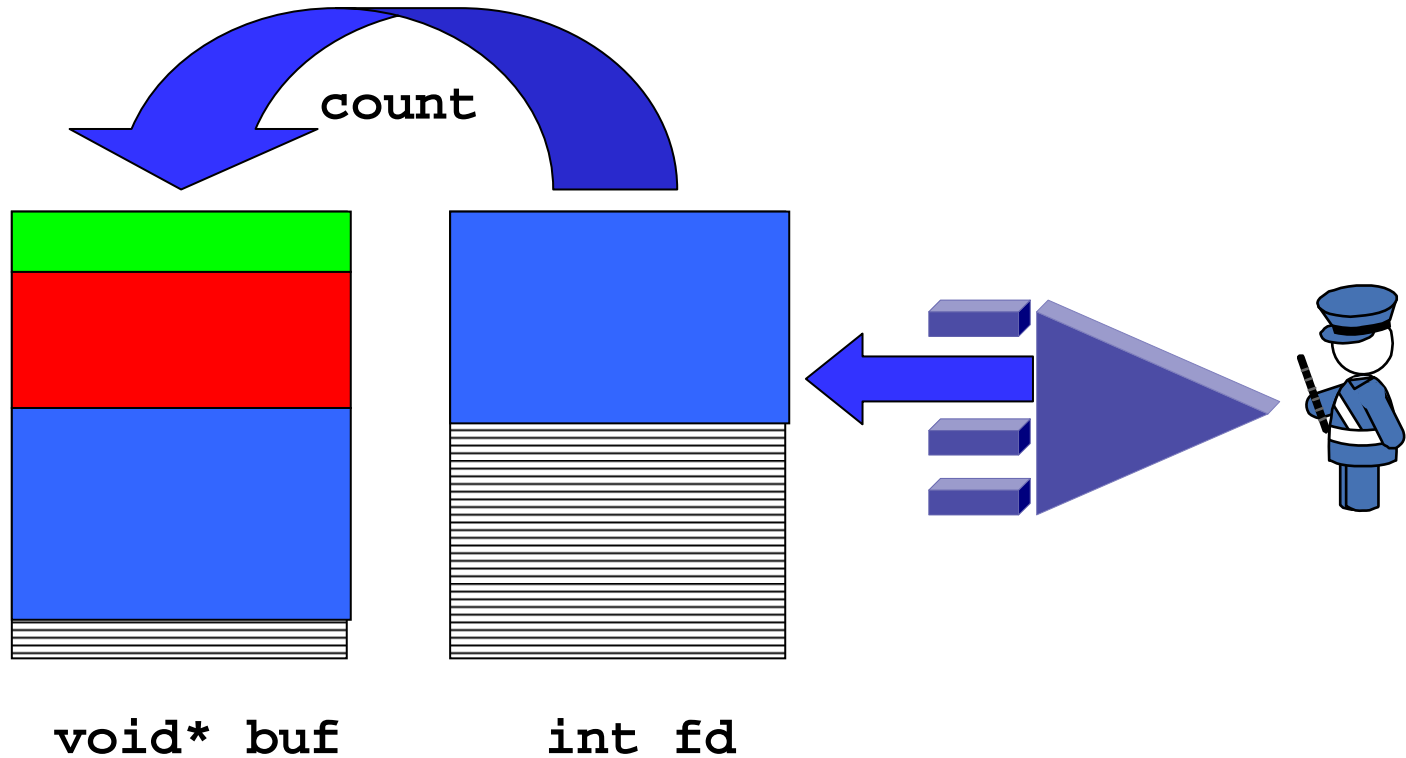
Read sequenza #1 start



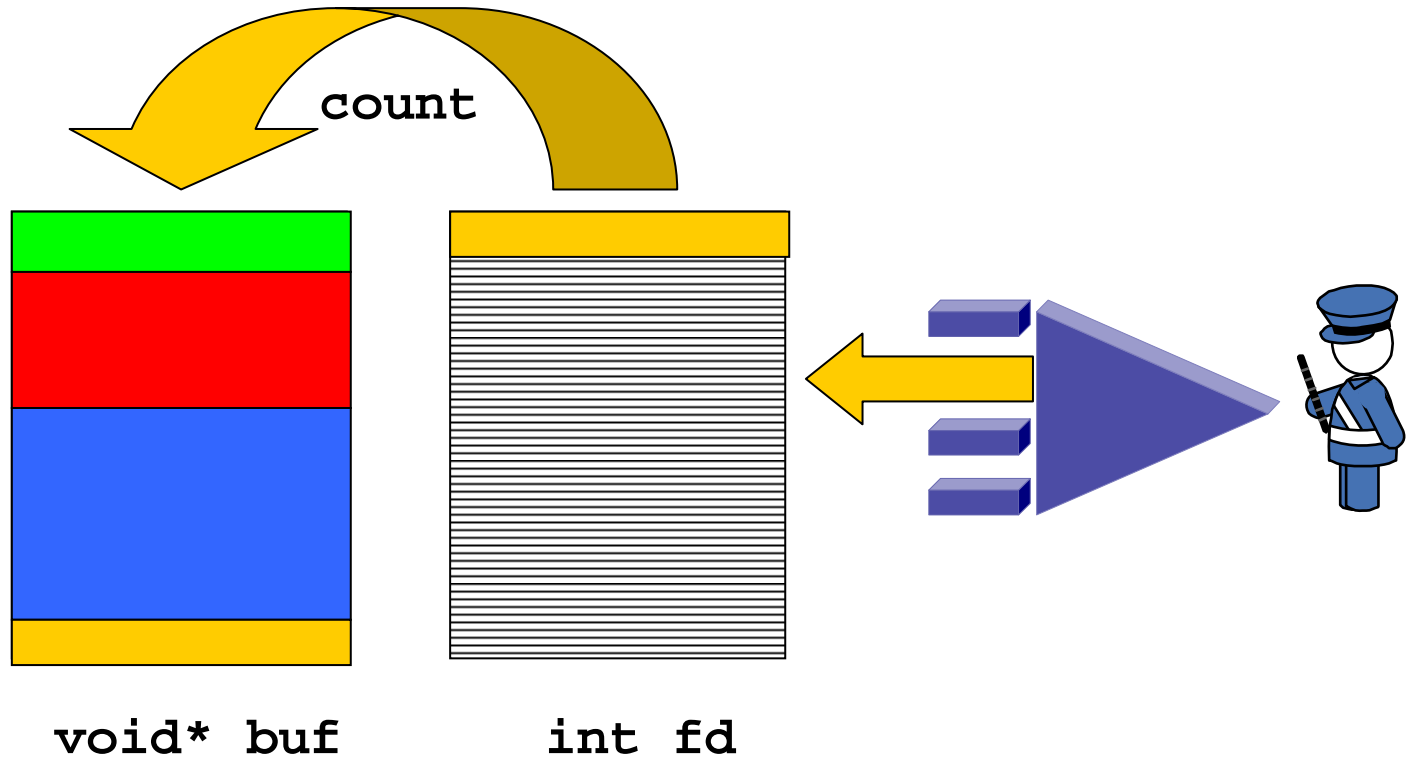
Read sequenza #2



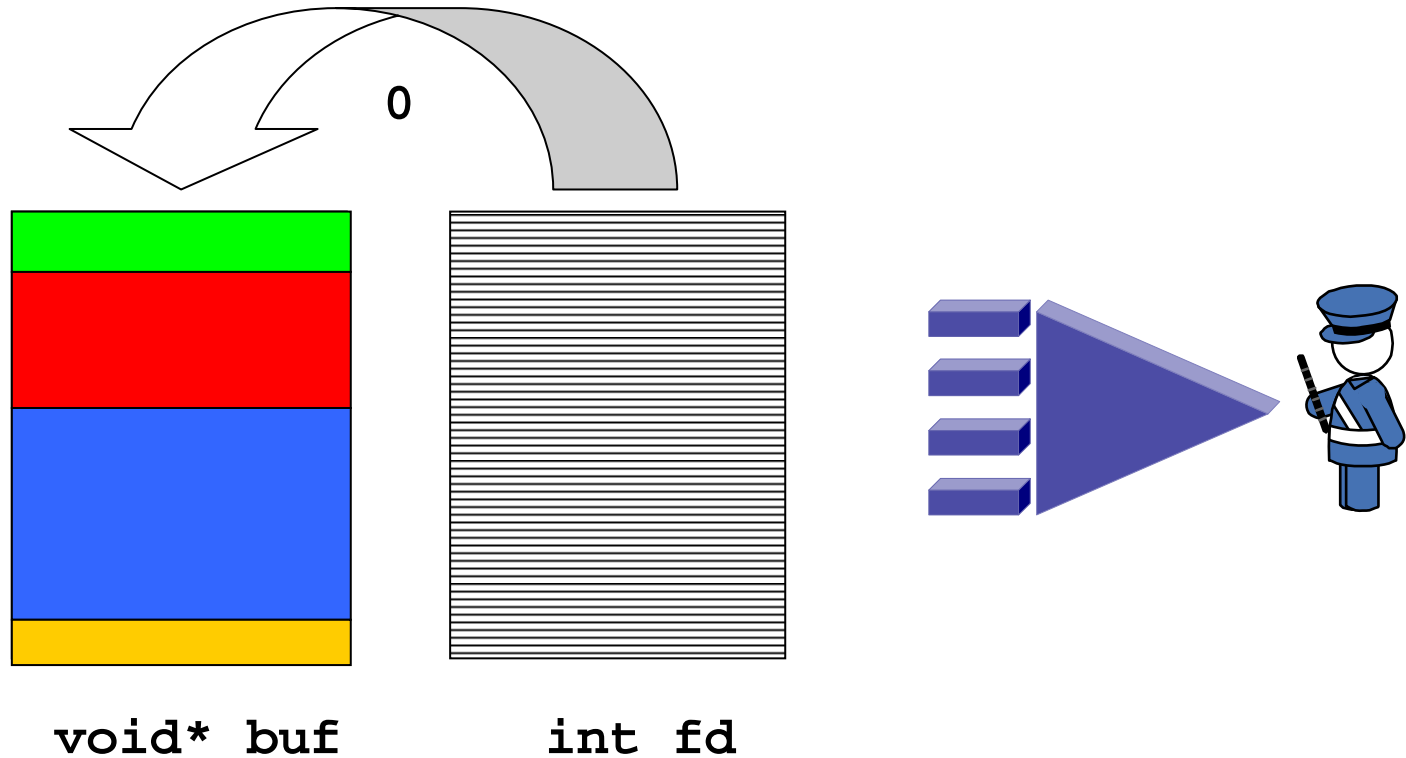
Read sequenza #3



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

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(AF_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);
}
```