

UNIX Network Programming

Andrea Passarella
a.passarella@iet.unipi.it

OSOR
Anno Accademico 2003/2004



C/C++: un paio di differenze



- Le variabili possono essere definite SOLO ALL'INIZIO DI UN BLOCCO

Stile C++

```
int main(void)
{
    int a=5, i, b;
    a=func(1000);
    int b=f(a);
    ...
    for(i=0; a<100; i++){
        b=f(a);
        int c=0;
        ...
    }
}
```

Stile C

```
int main(void)
{
    int a=5, i, b;
    int b=f(a);
    a=func(1000);
    ...
    for(i=0; a<100; i++){
        int c=0;
        b=f(a);
        ...
    }
}
```

C/C++: un paio di differenze



- Le strutture vanno sempre riferite con la parola chiave `struct`

Stile C++

```
struct Complesso{
    double re;
    double im;
}

int main(void)
{
    int a=5;
    Complesso c;
    ...
}
```

Stile C

```
struct Complesso{
    double re;
    double im;
}

int main(void)
{
    int a=5;
    struct Complesso c;
    ...
}
```

Gestione della memoria dinamica



- Allocazione
- Deallocazione

```
int main(void)
{
    int mem_size=5;
    void *ptr;
    ptr = malloc(mem_size);
    if(ptr == NULL){
        /* gestione condizione
        di errore */
    }
    ...
}
```

```
int main(void)
{
    int mem_size=5;
    void *ptr;
    ptr = malloc(mem_size);
    if(ptr == NULL){
        /* gestione condizione
        di errore */
    }
    ...
    free(ptr);
}
```

I/O

□ Output su standard output (video)

```
int main(void)
{
    int i=5;
    char *str="ciao ciao\n";
    printf(str);
    printf("non vado a capo");
    printf("ora si\n");
    printf("i=%d\n", i);
    ...
}
```

*> ./prog
ciao ciao
non vado a capoora si
i=5
>*

□ Input da standard input (tastiera)

```
int main(void)
{
    char *str = (char *)malloc(100);
    fgets(str, 100, stdin);
    ...
}
```

Gestione delle stringhe

□ Lunghezza

```
int main(void)
{
    char *str="ciao ciao\n";
    int len;
    ...
    len = strlen(str);
    ...
}
```

str
↓
11 bytes allocati in memoria
c i a o c i a o \n \0
len = 10 !!!

□ Comparison

```
int main(void)
{
    char *str1="ciao", *str2="bye";
    int i = strcmp(str1, str2);
    ...
}
```

i<0 str1 alfabeticamente
minore di str2
i>0 str1 alfabeticamente
maggiore di str2
i=0 str1 uguale a str2

Gestione delle stringhe

□ Copia

```
int main(void)
{
    char str1[100];
    strncpy(str1, "ciao\n", sizeof(str1)-1);
    str[99] = '\0';
    ...
}
```

□ Concatenazione

```
int main(void)
{
    char str1[100];
    char *str2 = "bye\n";
    strncpy(str1, "ciao\n", sizeof(str1)-1);
    str[99] = '\0';
    strncat(str1, str2, sizeof(str1)-strlen(str1)-1);
    str[99] = '\0';
    ...
}
```

Gestione dei files

□ Apertura

```
int main(void)
{
    FILE *fp;
    fp = fopen("/tmp/prova.txt", "r");
    if(fp == NULL){
        /* gestione errore */
    }
    ...
}
```

path relativo o assoluto
↑
Modalita' di apertura:
"r": read-only
"w": write-only
"r+": read and write
"a": append
"a+": append and read

*Aperture in write/append di files inesistenti causano creazione del file
(a patto di avere permessi sufficienti sulle directory del path)*

Gestione dei files

□ Lettura

```
int main(void)
{
    int ret;
    char str[1024];
    FILE *fp;
    fp = fopen("/tmp/prova.txt", "r");
    ret = fread(str, 1, sizeof(str)-1, fp);
}
```

□ Scrittura

```
int main(void)
{
    int ret;
    char *str="ciao ciao";
    FILE *fp;
    fp = fopen("/tmp/prova.txt", "w");
    ret = fwrite(str, 1, strlen(str), fp);
}
```

Gestione dei files

□ Dimensione di un file

```
int main(void)
{
    int ret, size;
    struct stat info;
    ret = stat("/tmp/prova.txt", &info);
    size = info.st_size;
}
```

□ Chiusura

```
int main(void)
{
    FILE *fp;
    fp = fopen("/tmp/prova.txt", "r");
    fclose(fp);
}
```

Esecuzione di un file eseguibile

```
int main(void)
{
    char *str = "/bin/ls -l > file1";
    system(str);
    system("/bin/ls -l > file1");
}
```

Esattamente lo stesso risultato

Lo standard output va redirezionato su un file o viene perso

Includes

□ Headers da includere

```
#include <stdlib.h> per malloc(), free(), system()
#include <stdio.h> per printf(), fgets(), fopen(), fclose(), fread(), fwrite()
#include <string.h> per strlen(), strcpy(), strcat(), strcmp()

#include <sys/types.h>
#include <sys/stat.h> per stat()
#include <unistd.h>
```

Addressing

```
struct sockaddr_in{
    sa_family_t sin_family;
    u_int16_t sin_port;
    struct in_addr sin_addr;
};
struct in_addr{
    u_int32_t s_addr;
};

struct sockaddr_in addr_a;
memset(&addr_a, 0, sizeof(addr_a)); /* blank with 0s */
addr_a.sin_family = AF_INET; /* IPv4 address */
addr_a.sin_port = htons(1234); /* SRV_PORT, network ordered */
inet_pton(AF_INET, "192.168.1.1", &addr_a.sin_addr);
```

Annotations:

- `sin_family` → AF_INET
- `sin_port` → port, 16 bit
- `s_addr` → IPv4 address, 32 bit

Server side

```
#define SA struct sockaddr;
struct sockaddr_in my_addr, cl_addr;
int ret, len, sk, cl_sk;

sk = socket(AF_INET, SOCK_STREAM, 0);
memset(&my_addr, 0, sizeof(my_addr));
my_addr.sin_family = AF_INET;
my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
my_addr.sin_port = htons(1234);

ret = bind(sk, (SA *) &my_addr, sizeof(my_addr));
ret = listen(sk, 10);

len = sizeof(cl_addr);
cl_sk = accept(sk, (SA *) &cl_addr, &len);
```

Client side

```
#define SA struct sockaddr;
struct sockaddr_in srv_addr;
int ret, sk;

sk = socket(AF_INET, SOCK_STREAM, 0);
memset(&srv_addr, 0, sizeof(srv_addr));
srv_addr.sin_family = AF_INET;
srv_addr.sin_port = htons(1234);
ret = inet_pton(AF_INET, "192.168.1.1", &srv_addr.sin_addr);

ret = connect(sk, (SA *) &srv_addr, sizeof(srv_addr));
```

Communication

- To send data:


```
int ret, sk_a;
char msg[1024];
...
strcpy(msg, "something to send");
ret = send(sk_a, (void *) msg, strlen(msg), 0);
if (ret == -1 || ret < strlen(msg)) { /* error */
    ...
}
```
- To receive data (MSG_WAITALL flag):


```
int ret, len, sk_a;
char msg[1024];
...
ret = recv(sk_a, (void *) msg, len, MSG_WAITALL);
/* len is the size of the incoming message (<= sizeof(msg)) */
if ( (ret == -1) || (ret < len) ) { /* error */
    ...
}
```

Communication

□ To receive data (0 flag):

```
int ret, len, sk_a;
char msg[1024];
...
ret = recv(sk_a, (void *) msg, len, 0);
/* len is the size of the incoming message (<=
sizeof(msg)) */
if(ret == -1){/* error */
    ...
}
```

Includes

□ Headers to be included

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
```

Progetto: esempio di run

□ Server

```
>./ftp_server
FTP server, v.0.1
Waiting for connections...
Client 131.114.5.78 connected!
File list sent to the client
Received file pippo.txt, size 56B
Sent file ciao.c, size 1035B
Client 131.114.5.78 disconnected!
```

□ Client

```
>./ftp_client
FTP client, v.0.1
Connected to server 146.48.25.3!
ftp>ls
Server file list is:
ftp>get
Which file?>pippo.txt
Received file pippo.txt, size is 56 B

ftp>lls
Local file list is:
ftp>put
Which file?>ciao.c
Sent file ciao.c, size is 1035B
ftp>bye
>
```