SC 14/4/2026

13/04/26 17:44

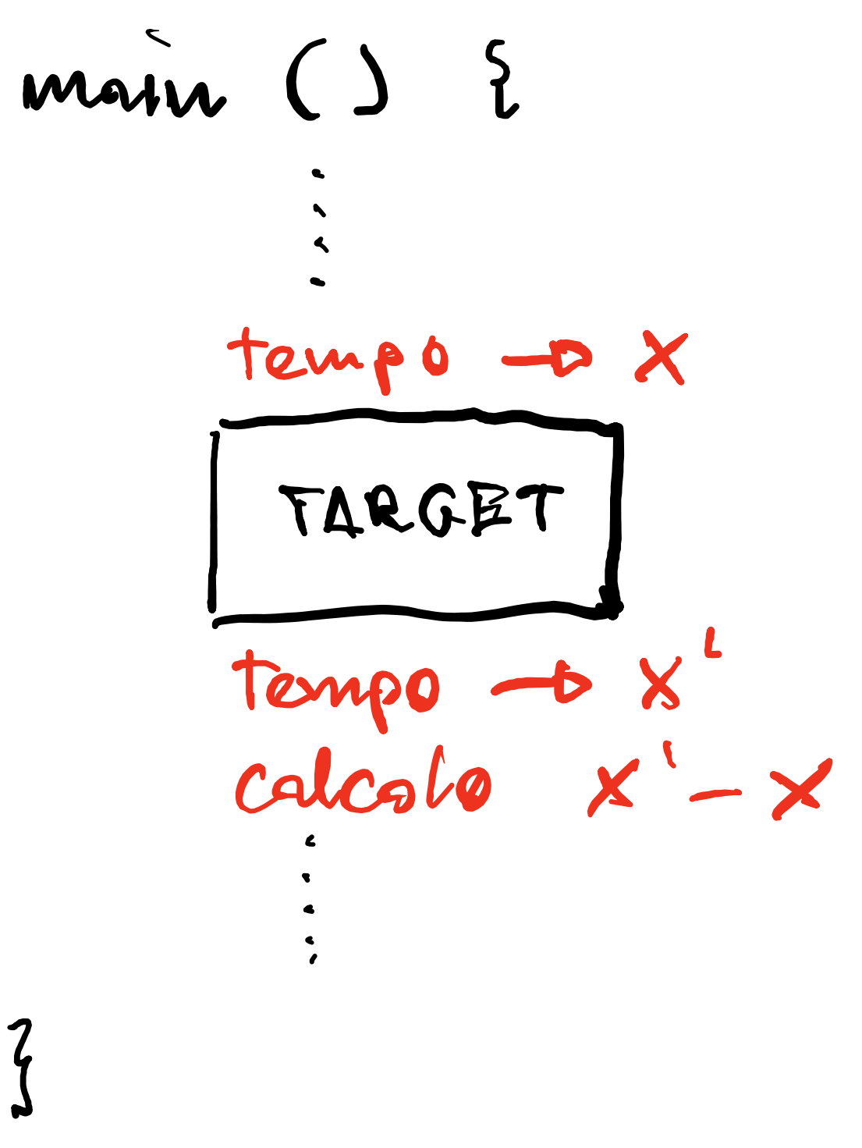
PROFILAZIONE DELLE PRESTAZIONI

Calcolare la latenza di esecuzione di ogni parte di un programma

MANUALE : tramite instrumentazione manuale del codice

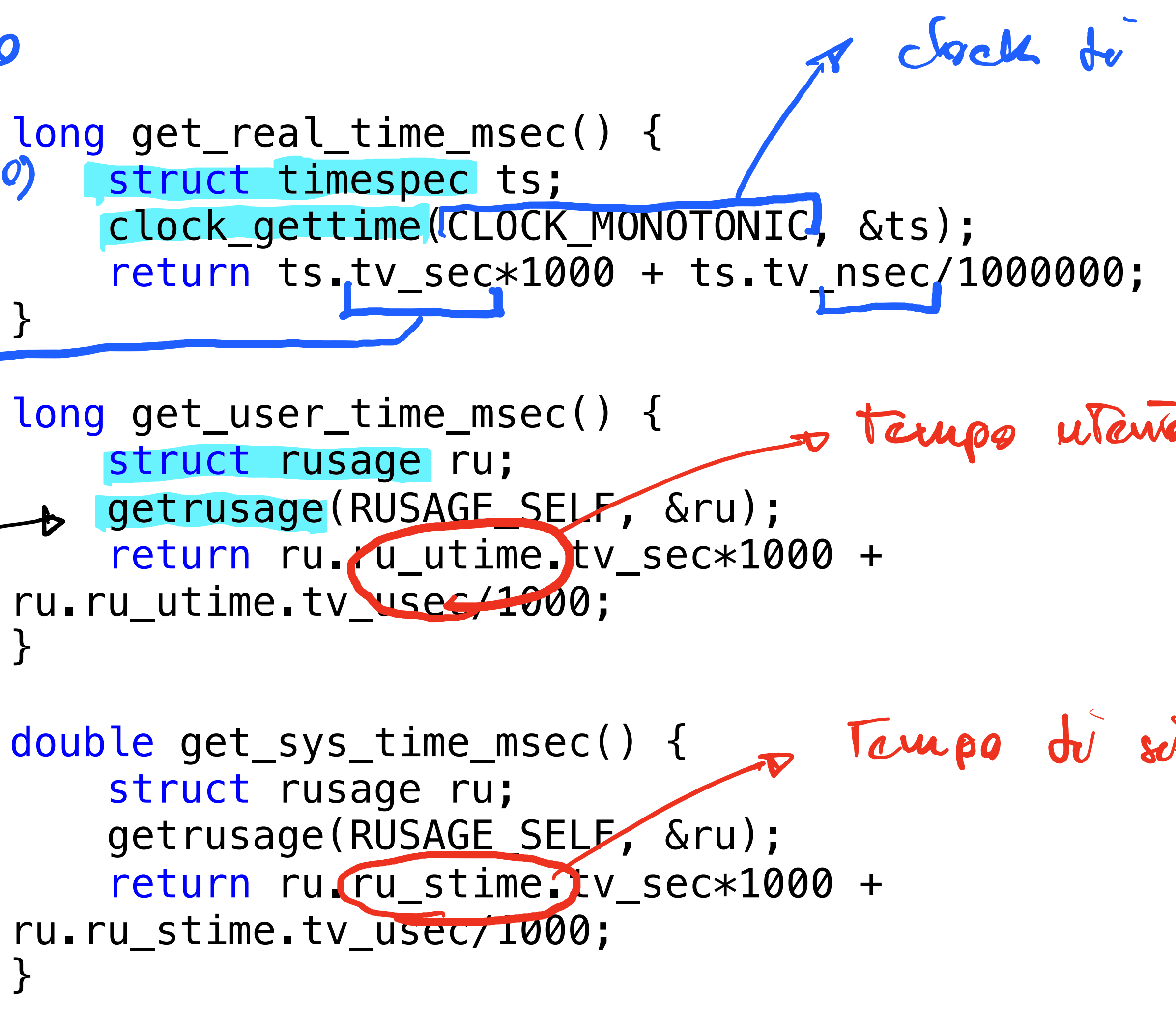
AUTOMATICA : instrumentazione fatta da un Tool

PROFILAZIONE MANUALE



Numero di msec passati da TEMPO STANDARD UNIX (1/1/1970 00:00:00)

metodo di lettura del clock di sistema



```
long get_real_time_msec() {
    struct timespec ts;
    clock_gettime(CLOCK_MONOTONIC, &ts);
    return ts.tv_sec*1000 + ts.tv_nsec/1000000;
}
```

tempo utente

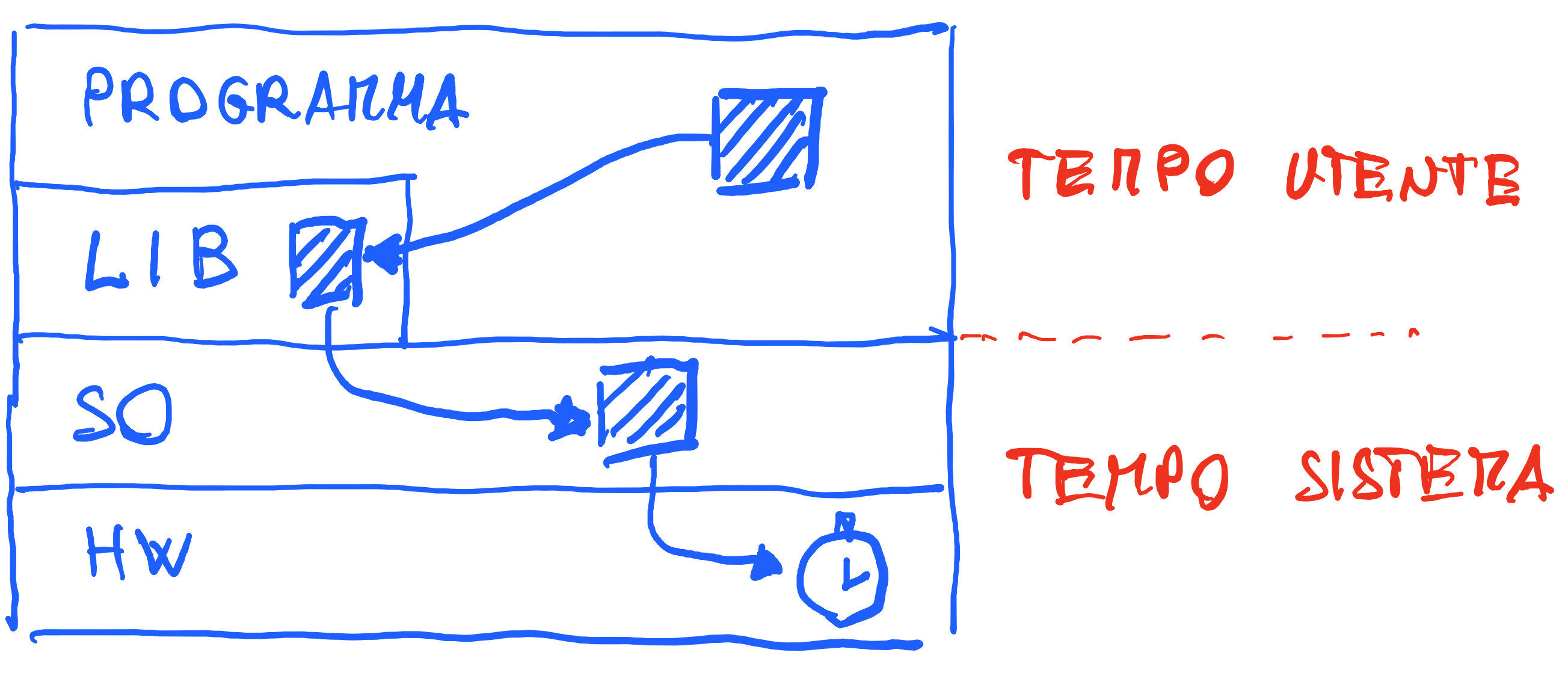
distingue tra tempo utente e tempo di sist.

```
long get_user_time_msec() {
    struct rusage ru;
    getrusage(RUSAGE_SELF, &ru);
    return ru.ru_utime.tv_sec*1000 +
ru.ru_utime.tv_usec/1000;
}
```

Tempo di sistema

```
double get_sys_time_msec() {
    struct rusage ru;
    getrusage(RUSAGE_SELF, &ru);
    return ru.ru_stime.tv_sec*1000 +
ru.ru_stime.tv_usec/1000;
}
```

Quale "tempo" stiamo misurando ?



LIMITI NELLA MISURAZIONE

① **LATENZA DELLO STRUMENTO** : tempo necessario per eseguire la misurazione

② **RISOLUZIONE DELLO STRUMENTO** : minimo intervallo temporale misurabile

IMPORTANTE : La misurazione di T con uno strumento A è significativa solo se la risoluzione di A è maggiore di T

IMPORTANTE : La latenza dello strumento usato per la misura deve influire sui risultati in modo trascurabile

PROFILAZIONE AUTOMATICA

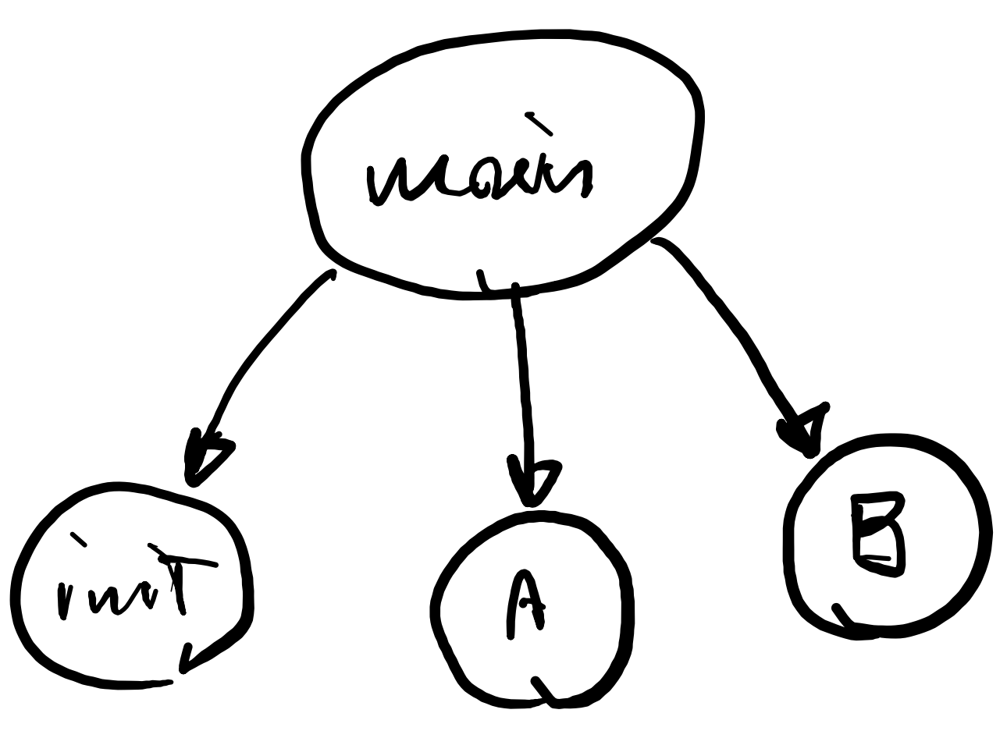
Utilizzo del Tool "gprof"

① Compilare il codice target con "-pg"

② Eseguo il programma compilato

③ Visualizza i risultati con "gprof <nome target>"

CALL GRAPH



ASPETTI CHIAVE DELLE OTTIMIZZAZIONI

OBIETTIVO : ridurre uso delle risorse massimizzando le prestazioni (latenza) senza impatto su correttezza

Possiamo intervenire su due livelli

ALGORITMO : scegliere l'algoritmo con costo computazionale minore che risolve il problema

⇓

progettista o programmatore

+ Impatto MOLTO significativo sulle prestazioni
− Non sempre è possibile
− Richiede una implementazione ad-hoc

IMPLEMENTAZIONE : ottimizzare il codice che implementa l'algoritmo scelto

⇓

programmatore o compilatore

+ Soluzione (parzialmente) automatica
+ Soluzione "economica" da adottare
− Possono essere "platform - dependent"

TECNICHE DI OTTIMIZZAZIONE DEL CODICE

Due approcci :

① RIDUZIONE DEL "WORK" : ridurre il numero di istruzioni da eseguire

② RIDUZIONE DEL "COSTO" : scelgo tra più alternative quella che impegna la CPU per meno tempo

Il compilatore applica questi approcci in molte numerose tecniche di ottimizzazione attraverso queste opzioni :

-O0 : default con cui non applica nessuna ottimizzazione

-O1 : ottimizza

-O2 : " +

-O3 : " ++

-Og : ottimizza il codice cercando di preservare i simboli di debug

-Os : ottimizza riducendo lo spazio