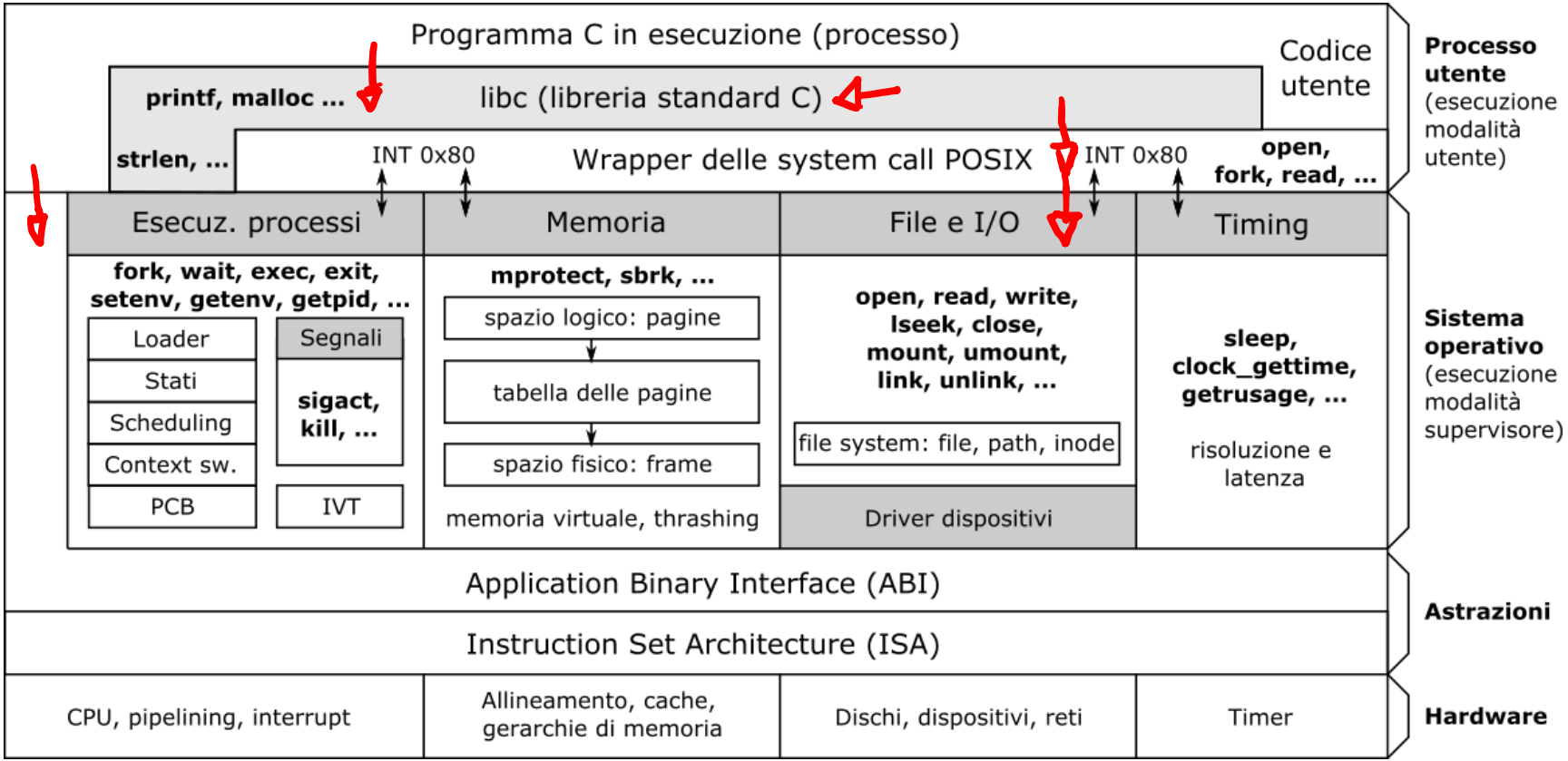
SAPIENZA
UNIVERSITÀ DI ROMA

Sistemi di calcolo

Capitolo 4
Librerie standard per il programmatore C

Corso di Laurea in Ingegneria Informatica e Automatica

Hardware-sofware stack di un sistema Linux



Accesso alla documentazione: man

La documentazione è divisa in capitoli:

1) Comandi eseguibili dalla shell (command line)
2) chiamate a sistema
3) chiamata a funzioni di libreria C

DEMO: man printf, man write, man strlen

Gestione degli errori

Ogni processo ha un valore intero che viene usato come codice di terminazione restituito al processo che lha invocato.

Il codice di terminazione viene definito dalle chiamate exit, _exit, abort, o dal return del main. Il codice vale 0 se e solo se l'esecuzione è andata a buon fine.

Gestore terminazione di un programma: atexit

variabile che contiene indirizzo funzione

puntatore a funzione

```
#include <stdlib.h>
int atexit(void (*function)(void));
```

Parametri:

- `function`: puntatore a una funzione da eseguire in caso di terminazione con la funzione `exit` oppure con `return` dal `main`

Risultato:

- 0 in caso di successo, -1 in caso di errore

es. memoria esaurita (molto raro)

Gestore terminazione di un programma: atexit

Esempio:

```
#include <stdio.h>   // printf
#include <stdlib.h>  // atexit

void handler() {
    printf("uscita dal programma\n");
}

int main() {
    atexit(handler);
    return 0; // stampa: "uscita dal programma"
}
```

handler (gestore)

Installa handler

provoca esecuzione handler

Errori non recuperabili

non recuperabili: provocano terminazione programma (es. divisione per zero o esecuzione della macro assert)

```
#include <assert.h>
assert(test);
```

Parametri:

- `test`: valore di verità. Se diverso da zero, provoca la terminazione del programma con un codice diverso da zero e invia sul canale stderr un messaggio di errore della forma "Assertion failed: (<test>), function main, file <nomefile.c>, line <numero-linea>"

Risultato:

- nessuno

Es. `p = malloc(10000000);`
`assert (p != NULL)`

Errori recuperabili

Provocano la terminazione della funzione corrente con un codice di terminazione e l'impostazione della variabile globale errno (#include <errno.h>) al codice di errore

Convenzioni:
- funzione che restituisce puntatore restituisce NULL in caso di errore.
- funzioni dello standard POSIX restituiscono 0 in caso di successo e tipicamente -1 se errore (e impostano errno)

Pattern gestione errori recuperabili

```
int mia_funzione() {
    int res, *buffer = NULL;
    ...
    buffer = malloc(...);
    if (buffer == NULL) goto cleanup; // errore nella chiamata
    ...
    res = chiamata(...);
    if (res != 0) goto cleanup;       // errore nella chiamata
    ...
    free(buffer);
    return 0;

cleanup:
    ...
    if (buffer != NULL) free(buffer); // deallocazione buffer
    return -1;
}
```

pulizia risorse prima di uscire

Libreria standard C: gestione delle stringhe

```
#include <string.h>

size_t strlen(const char *s);

char *strcpy(char *dest, const char *src);

char *strcat(char *dest, const char *src);

int strcmp(const char *s1, const char *s2);

char *strtok(char *str, const char *delim);

#include <stdlib.h>
int atoi(const char *nptr);
```

lunghezza stringa

copia stringa

concatena stringhe

confronta stringhe:
- <0 : s1 < s2
- 0 : s1 = s2
- >0 : s1 > s2

"tokenizza" stringhe (vedi esempio)

converte a numero int prefisso numerico stringa

Libreria standard C: gestione delle stringhe

DEMO 4.1-strings

Libreria standard C: gestione delle stringhe

```
#include <stdio.h>

int sprintf(char *str, const char *format, ...);

int sscanf(char *str, const char *format, ...);
```

crea stringa secondo il formato specificato da format

estrae dati da una stringa str seguendo il formato format

Libreria standard C: gestione delle stringhe

DEMO 4.1-strings

Libreria standard C: manipolazione memoria

Copia n byte dall'indirizzo src all'indirizzo dest

```
#include <string.h>

void *memcpy(void *dest, const void *src, size_t n);

int memcmp(const void *s1, const void *s2, size_t n);

void *memset(const void *dest, int c, size_t n);
```

setta a c n byte che iniziano a dest

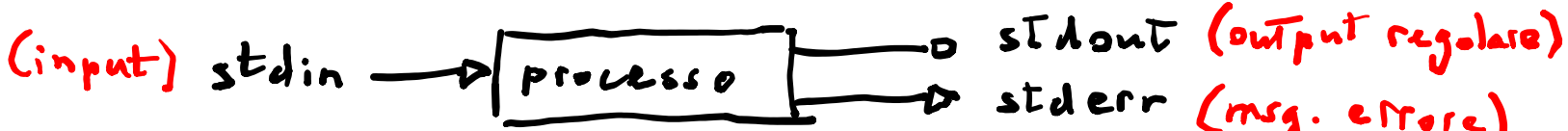
confronta lessicograficamente i byte dei due buffer di n byte s1 ed s2.

Libreria standard C: manipolazione memoria

DEMO 4.2-mem

Libreria standard C: stdin, stoud, stderr

La libreria standard C definisce tre canali di comunicazione con il terminale:



Vedremo in seguito come leggere dati da stdin e scrivere dati su stdout e stderr. Vedremo inoltre come redirigere questi canali in modo che si riferiscano a file e non al terminale.

Libreria standard C: gestione dei file di testo

```
#include <stdio.h>

FILE *fopen(const char *pathname, const char *mode);

int fclose(FILE *stream);

int fprintf(FILE *stream, const char *format, ...);

int fscanf(FILE *stream, const char *format, ...);

char* fgets(char *str, int size, FILE *stream);
```

apertura file come canale
modalità apertura (r/w)
chiusura file
scrittura stringhe su canale
lettura stringhe da canale
lettura linee testo da canale

Libreria standard C: gestione dei file binari

```
#include <stdio.h>

size_t fwrite(const void *ptr, size_t size,
              size_t nitems, FILE *stream);

size_t fread(void *ptr, size_t size,
             size_t nitems, FILE *stream);

int fseek(FILE *stream, long offset, int whence);

long ftell(FILE *stream)
```

scrittura byte su canale

lettura byte da canale

imposta posizione corrente nel canale

recupera posizione corrente

Libreria standard C: file

DEMO 4.3-file

Libreria standard C: ordinamento e ricerca

```
#include <stdlib.h>

void qsort(void *base, size_t nel, size_t width,
     int (*compar)(const void *, const void *));

void *bsearch(const void *key, const void *base,
     size_t nel, size_t width,
     int (*compar) (const void *, const void *));
```

ordinamento di array di elementi generici

ricerca binaria in array ordinati

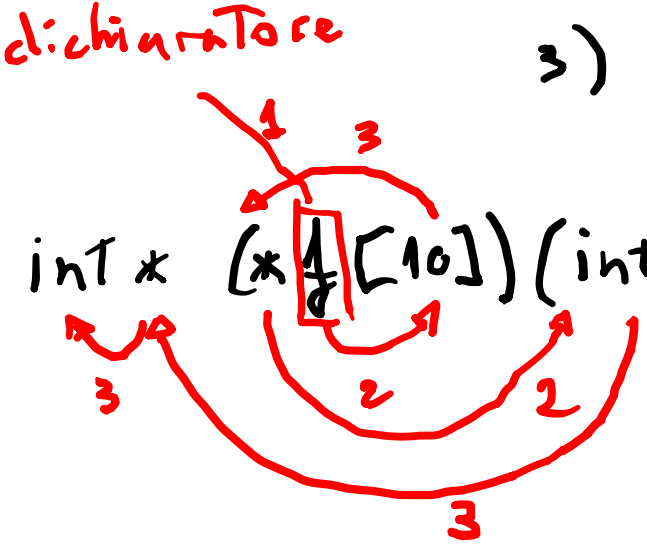
variabile che contiene l'indirizzo di una funzione

Comprensione espressioni di tipo in C

Regola generale: 1) parto dal dichiaratore (identificatore da dichiarare)

2) se posso, vado a destra

3) se devo (es.: chiusura parentesi) vado a sinistra



f è un array di 10 puntatori a funzione che prende un int e restituisce un puntatore a int.

Comprensione espressioni di tipo in C

Es. comparatore per qsort e bsearch

```
int (*compar)(const void*, const void*)
```

compar è un puntatore a una funzione che prende due void* e restituisce un int.

NB: Le parentesi intorno a (*compar) sono indispensabili

Libreria standard C: ordinamento e ricerca

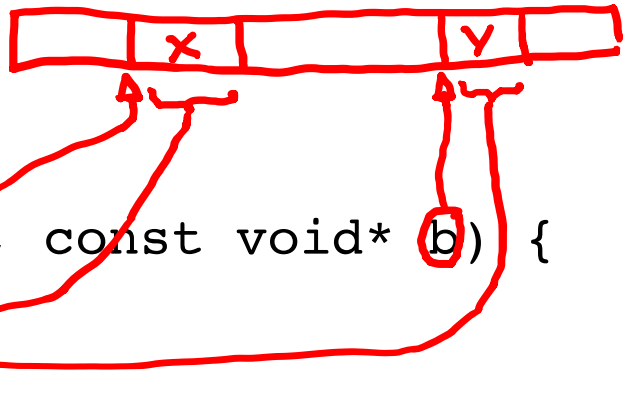
DEMO 4.4-sort-search

Ordinamento di interi

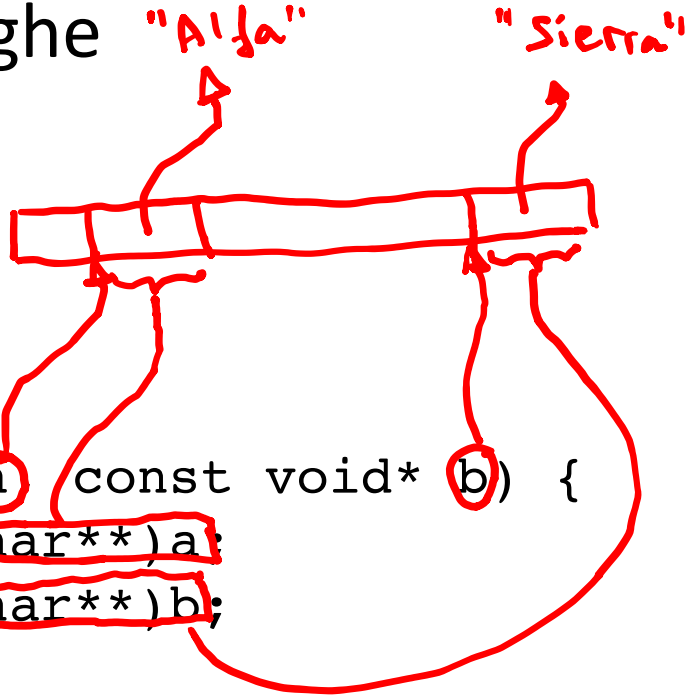
```
#include "e1.h"
#include <stdlib.h>
#include <string.h>

int compar(const void* a, const void* b) {
    int x = *(int*)a;
    int y = *(int*)b;
    return x-y;
}

void sort_ints(int ints[], size_t size){
    qsort(ints, size, sizeof(int), compar);
}
```



Ordinamento di stringhe




```
#include "e1.h"
#include <stdlib.h>
#include <string.h>

int compar(const void* a, const void* b) {
    const char* x = *(char**)a;
    const char* y = *(char**)b;
    return strcmp(x, y);
}

void sort_strings(char *strings[], size_t size){
    qsort(strings, size, sizeof(char*), compar);
}
```

Libreria standard C: funzioni e costanti matematiche

richiede anche che la libreria matematica sia linkata: gcc -lm ...

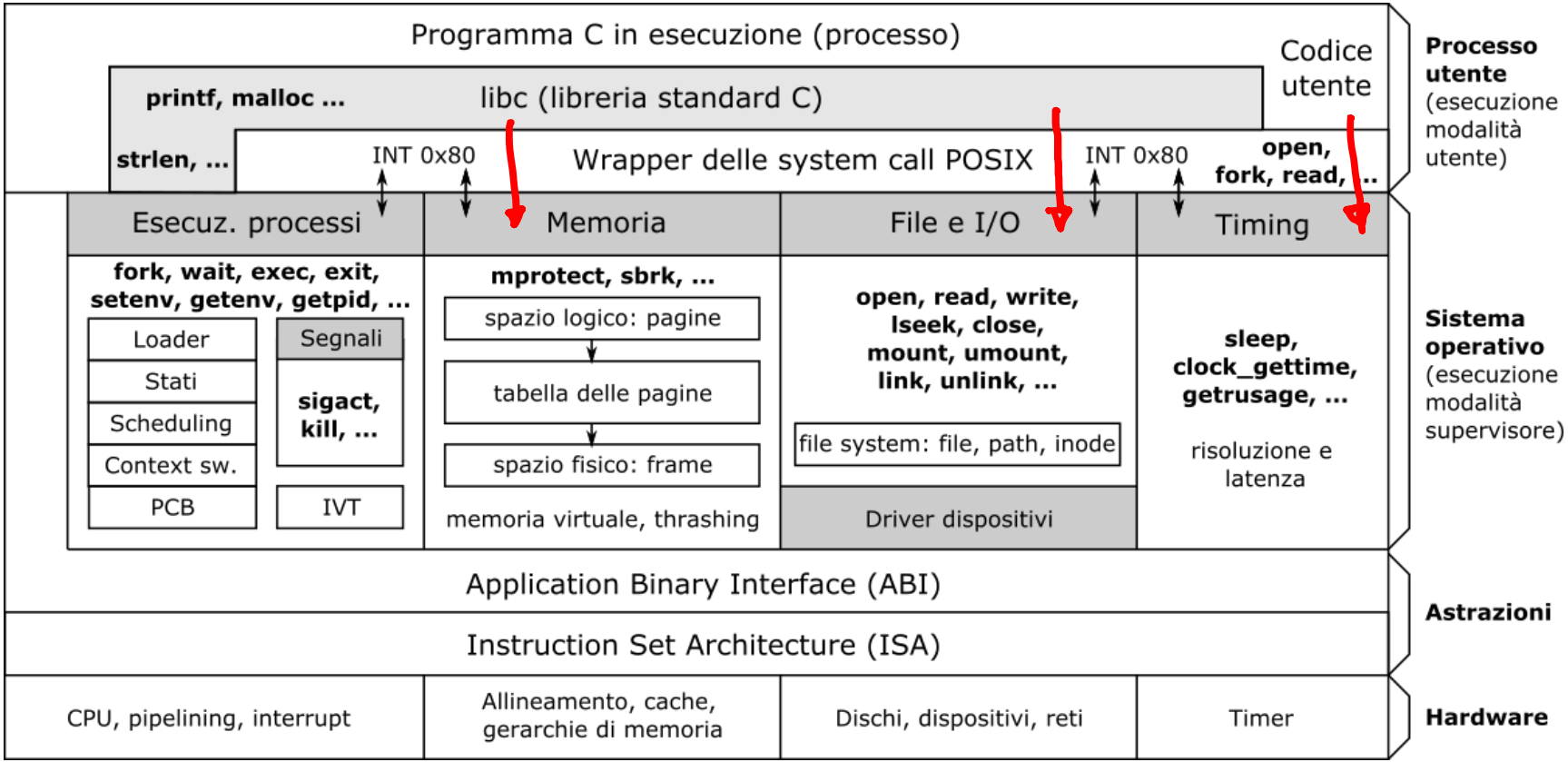
```
#include <math.h>

double sqrt(double);
double log(double);
double sin(double);
double cos(double);
double tan(double);
...
```

radice quadrata

https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/math.h.html

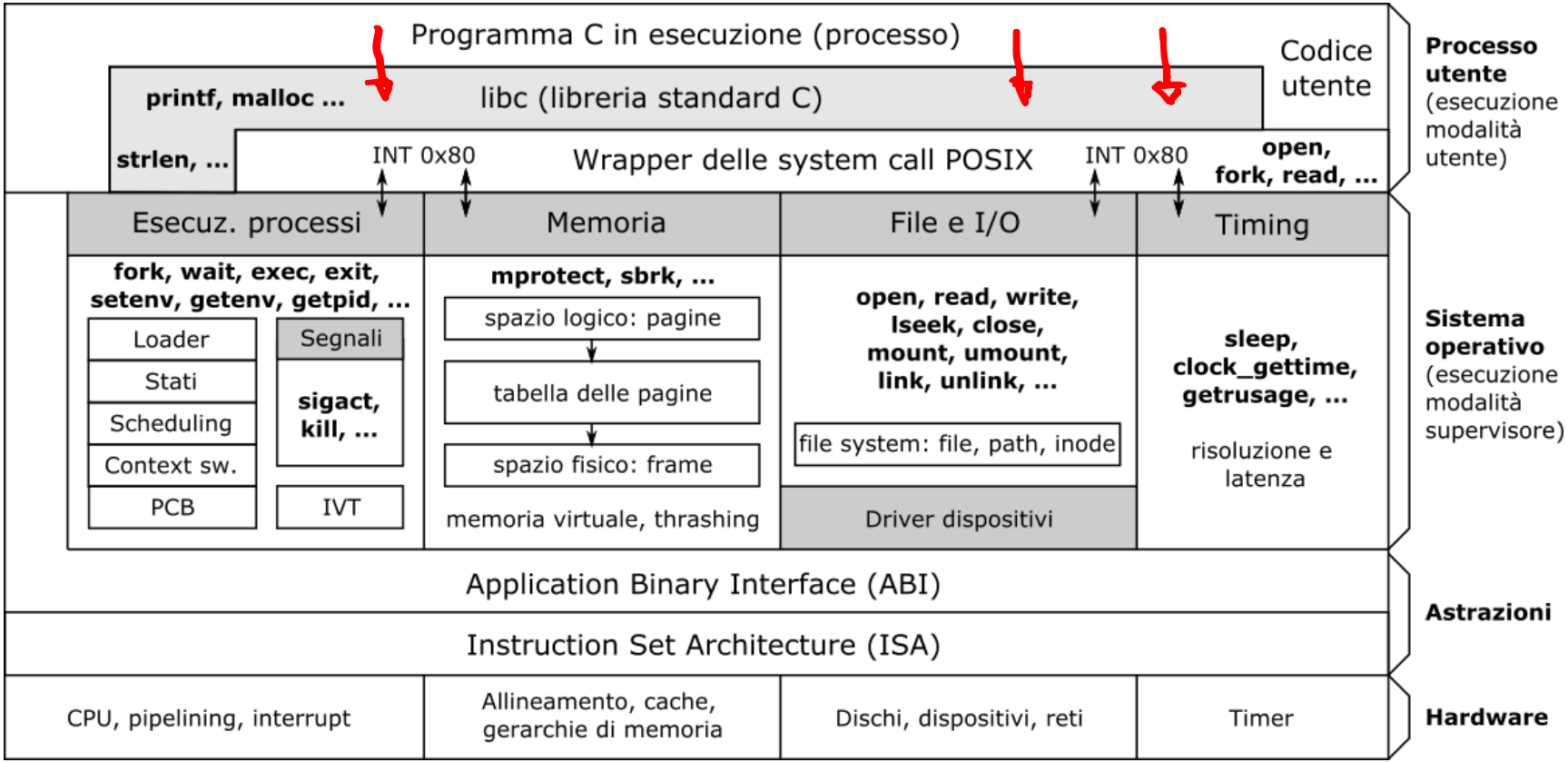
Tracciamento system call: strace



Tracciamento system call

DEMO 4.5-strace

Tracciamento chiamate a librerie: ltrace



Tracciamento chiamate a librerie

DEMO 4.5-ltrace