



UNIVERSITÀ DI PARMA

Input & Output

*The future belongs to those who believe in the
beauty of their streams.*

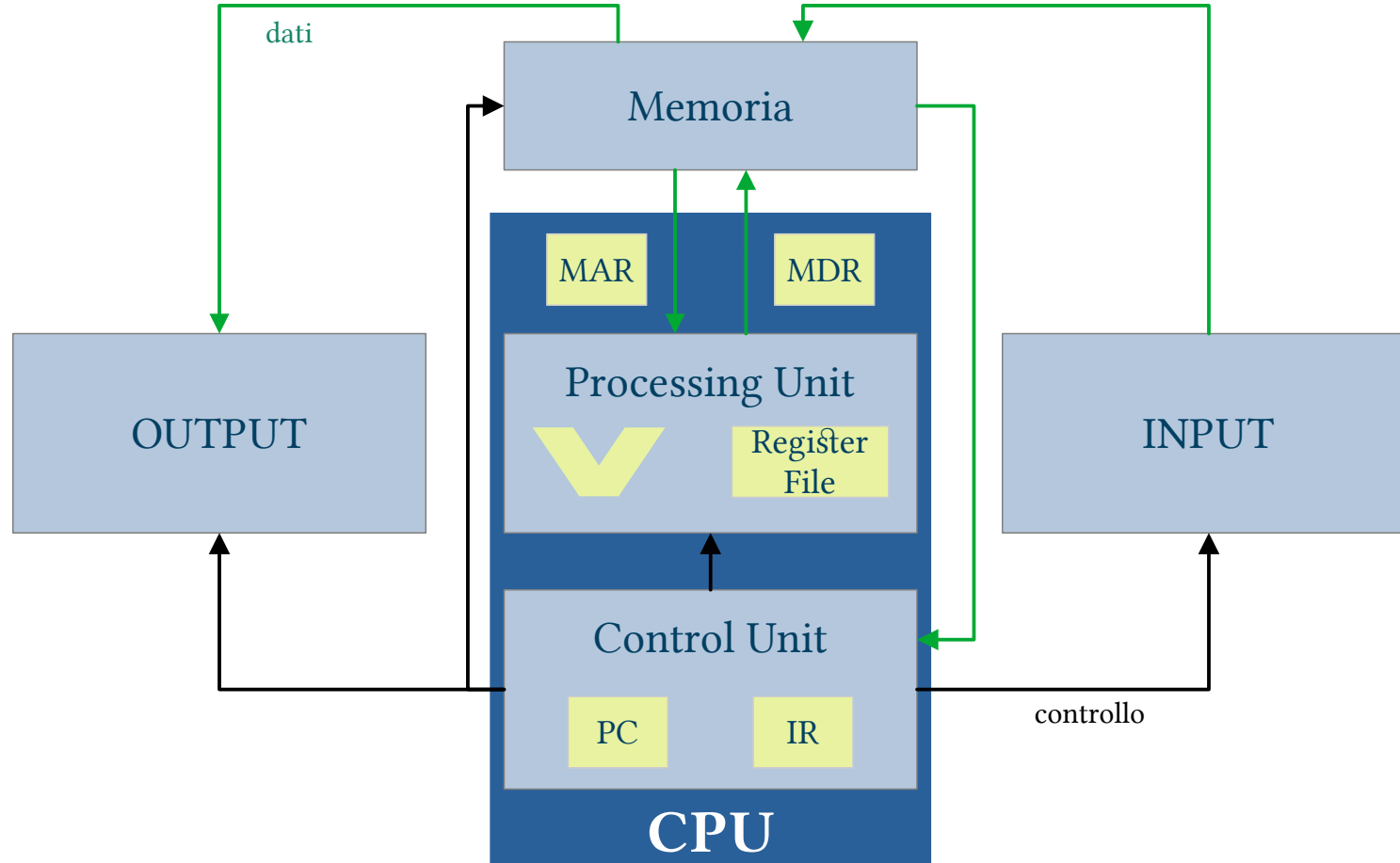
Eleanor Roosevelt (mod.)

- Non esaustivo → Manuale
- gli stream
- concetto di file
- principali operazioni
 - definizione
 - associare uno stream
 - lettura/scrittura e altre operazioni
 - File testuali e binari
 - chiusura
- I/O della console

SUMMARY



Von Neuman model

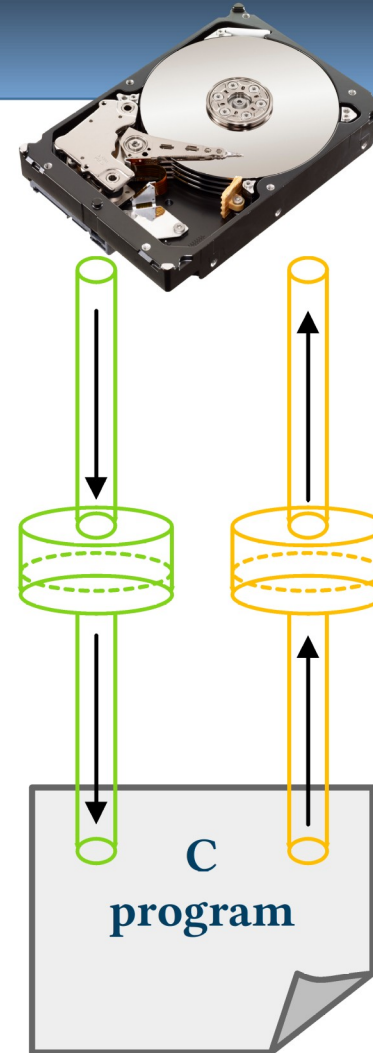


What is a stream?

- In C sono i “canali” di input o output
 - Già visto: tastiera e video
 - Ma ne possiamo avere altri
 - File, porte seriali, USB, ecc.
- Predefiniti
 - `stdin` → `scanf()`, ...
 - `stdout` → `printf()`, ...
 - `stderr`
- Vantaggio: un solo set di primitive per la gestione

What is a stream?

- Gli stream sono una struttura ad alto livello che ci separa dall'HW
- Tipicamente I/O asincrono
 - Bufferizzato
- Input, output o entrambi



- Definire uno stream
- Associare uno stream (ovvero “aprire”)
- Lettura o scrittura dati
 - Altre operazioni
- Chiusura

- Definire uno stream
- Associare uno stream (ovvero “aprire”)
- Lettura o scrittura dati
 - Altre operazioni
- Chiusura

- Si usa la struttura FILE
- Deve sempre essere un puntatore

```
FILE *miostream;  
FILE *altrostream, *altroancora;
```


- Definire uno stream
- Associare uno stream (ovvero “aprire”)
- Lettura o scrittura dati
 - Altre operazioni
- Chiusura

- Dopo averlo definito possiamo “associare” uno stream
 - File
 - Periferica
 - ...
- Noi vedremo solo file

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

Cosa è un file?

- In italiano “archivio”
- Principale contenitore informazioni su supporti digitali
 - Hard disk
 - DVD/CD/BD
 - Memory stick
 - ...
- Permanente



Come si usa un file?

- Nastro magnetico
 - Testina che legge/scrive
 - Man mano che legge/scrive si sposta
 - Ma posso anche spostarmi in un punto specifico del nastro
- File
 - File pointer → equivalente testina
 - Ad ogni operazione “avanza” nel file
 - Possibile saltare in posizioni specifiche



- Testuali o ASCII
 - Solo caratteri stampabili
 - Leggibili da essere umano
 - Esempi: codice sorgente C, file di configurazione (.ini),...
 - Massima portabilità \leftrightarrow spazio occupato
- File binari
 - Anche caratteri non stampabili
 - Più compatti \leftrightarrow portabilità

- Permette apertura file
- Due argomenti, stringhe:
 - Nome file (percorso relativo o assoluto), solo caratteri ASCII
 - Modalità apertura
- Restituisce
 - NULL se fallisce
 - Puntatore a struttura FILE se successo

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

r	lettura	Il file deve esistere
w	scrittura	Il file viene creato se non esistente o azzerato se esistente
a	scrittura a fine file	Il file viene creato se non esistente
r+	lettura e scrittura da inizio	Il file deve esistere
w+	azzeramento/creazione e lettura/scrittura	Il file viene creato se non esistente o azzerato se esistente
a+	lettura e scrittura a fine file	Il file viene creato se non esistente
b	In aggiunta ai precedenti se file binario	Esempi: “rb”, “w+b”, “wb+”...

- L'operazione di apertura può fallire:
 - File non esistente se richiesta lettura
 - Permessi
 - Errori di percorso...
- Fondamentale controllare quanto restituito da fopen()
- Utile l'uso di `void perror(const char *s)`

- Definire uno stream
- Associare uno stream (ovvero “aprire”)
- Lettura o scrittura dati
 - Altre operazioni
- Chiusura

- Buona norma chiudere file quando non più necessario
 - Massimo numero file aperti contemporaneamente
 - Svuoto buffer
- Disassocia stream
- Puntatore riutilizzabile

```
int fclose(FILE *stream);
```

- Definire uno stream
- Associare uno stream (ovvero “aprire”)
- Lettura o scrittura dati
 - Altre operazioni
- Chiusura

- (f)getc() legge un carattere da un file
- (f)putc() scrive un carattere in un file
- fgets() legge stringa (linea) da file
- fputs() scrive stringa in un file

- fscanf() legge un insieme di dati da un file
- fprintf() scrive un insieme di dati in un file

- I/O formattato vs non formattato
 - `fgetc()/fputc()` e `fgets()/fputs()` vs `fprintf()` e `fscanf()`
 - Preferire lettura formattata quando contenuto va interpretato
 - Errore comune: leggo stringa e poi la cerco di interpretare in memoria
- Le operazioni di I/O possono essere collo di bottiglia
 - Lettura/scrittura di singoli byte inefficiente?

- Scrivo/leggo un intero blocco di memoria
 - Indirizzo blocco
 - Dimensione singoli elementi
 - Numero elementi
 - Stream

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

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

- Come capisco se arrivo a fine file (solo lettura)?
- Due approcci
 - Funzione specifica `int feof(FILE *stream);`
 - Scatta solo dopo aver tentato di leggere oltre la fine del file!
 - Valori restituiti da funzioni lettura
 - `fscanf()` → numero specificatori di formato che riesce a leggere
 - `fread()` → numero byte letti
 - `fgets()` → NULL se non riesce a leggere
 - `fgetc()` → EOF se non riesce a leggere

- Definire uno stream
- Associare uno stream (ovvero “aprire”)
- Lettura o scrittura dati
 - Altre operazioni
- Chiusura

- Riposizionamento all'interno di un file
 - `int fseek(FILE *stream, long offset, int whence);`
 - `void rewind(FILE *stream);`
 - `long ftell(FILE *stream);`
- Altre...

- Come accennato anche la console (tastiera + video) è associata a stream
 - `stdin` → stream di input (tastiera)
 - `stdout` → stream di output (video)
 - `stderr` → stream di output (messaggi di errore)
- Possibile uso delle funzioni di I/O
 - Ma esistono alias di molte
 - `fprintf(stdin, ...` → `printf(...`

- Caratteri

- `fgetc()` → `int getchar(void);`
- `fputc()` → `int putchar(int c);`

- Stringhe

- `fputs()` → `int puts(const char *s);`
- `fgets()` → ~~`char *gets(char *s);`~~ //DEPRECATED

- La libreria del C *bufferizza* l'accesso ai flussi di I/O
 - Ciò che scrivo da tastiera finisce in un'area di memoria che sarà man mano “consumata” dalle operazioni di lettura
 - Tipicamente il buffer viene riempito alla pressione del tasto di invio
 - Va posta attenzione alla gestione dell'Input da console



UNIVERSITÀ DI PARMA

Input & Output



KEEP
CALM
IT'S
QUESTION
TIME

Don't cross the streams.

Egon, 1984