Il modello di esecuzione

Programmazione di Sistema

A.A. 2017-18

Argomenti

- Modello di esecuzione di un programma
- Implementazione del modello
- Preparazione ed esecuzione di un processo

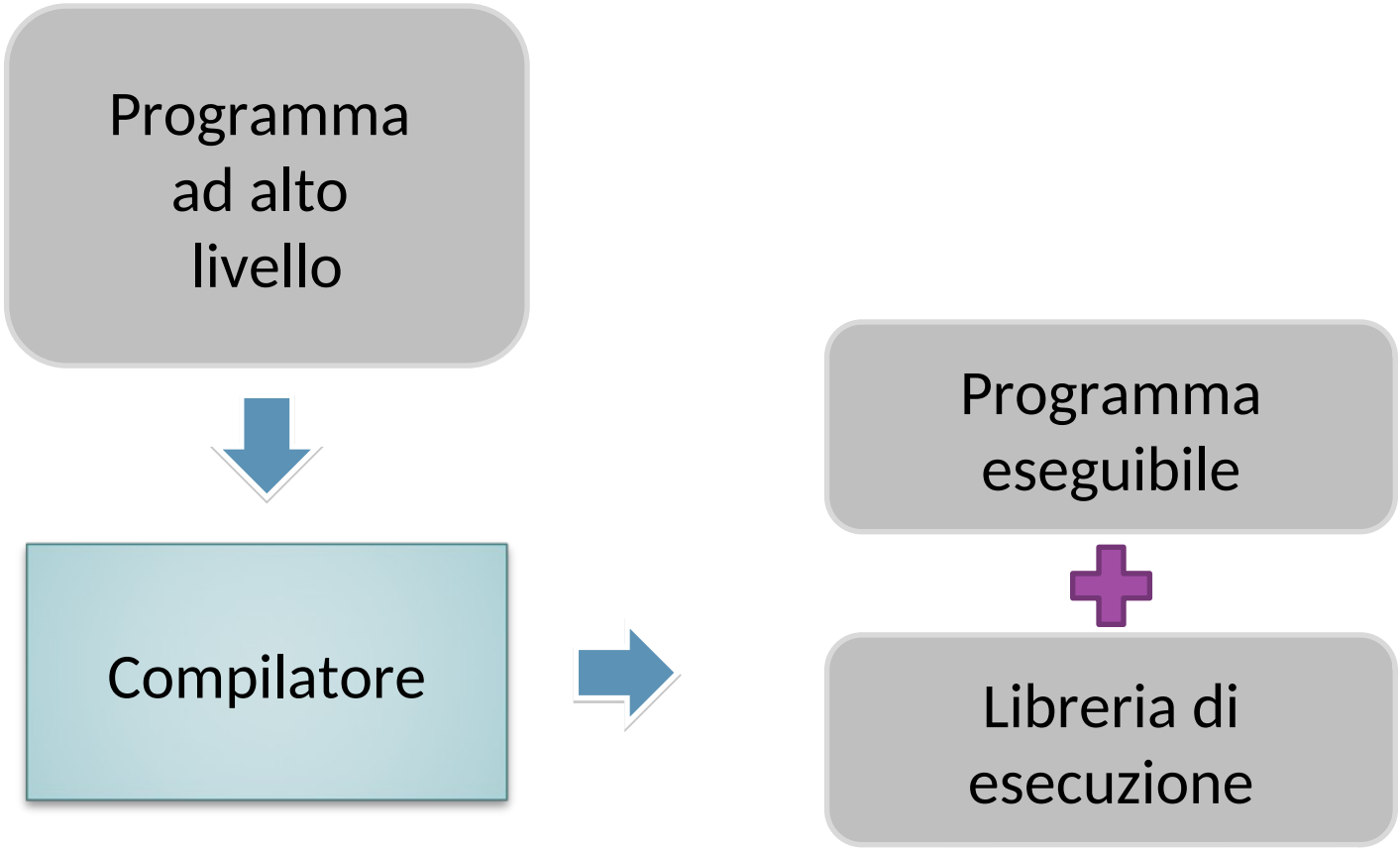
Modello di esecuzione di un programma

- Ogni linguaggio di programmazione propone uno specifico modello di esecuzione
  - Insieme di comportamenti attuati dall'elaboratore a fronte dei costrutti di alto livello del linguaggio
- Tale modello per lo più non corrisponde a quello di un dispositivo reale
  - Occorre introdurre uno strato di adattamento che implementi il modello nei termini offerti dal dispositivo soggiacente

Livelli di astrazione

- Il programma originale viene trasformato in un nuovo programma dotato di un modello di esecuzione più semplice
  - Che può essere eseguito da una macchina fisica (CPU) o subire un'ulteriore trasformazione (esecuzione virtuale)
- Il compilatore ha il compito di eseguire tale traduzione
  - In parte, riscrivendo il linguaggio in istruzioni più semplici (codice macchina)
  - In parte, avvalendosi di una libreria di esecuzione

Livelli di astrazione



Librerie di esecuzione

- Offrono agli applicativi meccanismi di base per il loro funzionamento
  - Supportano le astrazioni del linguaggio di programmazione
  - Forniscono un'interfaccia uniforme tra i diversi S.O. per le funzioni ad essi demandate
- Costituite da due tipi di funzioni
  - Alcune, invisibili al programmatore, sono inserite in fase di compilazione per supportare l'esecuzione (es.: controllo dello stack)
  - Altre offrono funzionalità standard, gestendo opportune strutture dati ausiliarie e/o richiedendo al S.O. quelle non altrimenti realizzabili (es.: malloc, fopen, ...)

Modello di esecuzione nei linguaggi C e C++

- I programmi sono pensati come se fossero gli unici utilizzatori di un elaboratore, completamente dedicato loro (isolamento)
  - Le eventuali interazioni si riducono al più all'uso di risorse persistenti come il file system
- Un programma C/C++ assume di poter accedere a qualsiasi indirizzo di memoria
  - All'interno del quale può leggere o scrivere dati o dal quale può eseguire codice macchina

Esecuzione sequenziale, senza limitazioni

- Un programma è formato da un insieme di istruzioni eseguite, una per volta, nell'ordine indicato dal programmatore
  - Non ci sono limiti sulle istruzioni da eseguire, sul tempo richiesto e sulla memoria necessaria

Flusso di esecuzione

- Il programma è costituito da un flusso di esecuzione il cui punto di partenza è predefinito
  - La funzione main() nel linguaggio C
  - I costruttori delle variabili globali in C++
- Una struttura a pila, lo stack, permette di gestire chiamate annidate tra funzioni
  - Supporta la ricorsione e l'uso di variabili locali
  - In C++, supporta anche la gestione strutturata delle eccezioni

Concorrenza

- A partire dallo standard ISO/IEC 9899:2O11, in un programma possono essere attivati ulteriori flussi di esecuzione
  - Eseguiti parallelamente al primo
  - Ciascuno dotato di un proprio stack

Implementazione del modello di esecuzione

- La libreria di esecuzione non può – da sola –implementare tutte le astrazioni del modello di esecuzione
  - In particolare quelle legate all'isolamento ed alle sue conseguenze
- Se più programmi sono in esecuzione, il (mal-) funzionamento di uno non deve avere conseguenze sugli altri
  - Occorre che il S.O. crei un meccanismo di isolamento semi-permeabile

Processi

- Per gestire l'esecuzione isolata dei programmi, i S.O. introducono il concetto di **processo**
  - Costituisce il contesto di esecuzione di un programma
  - Viene modellato da un'opportuna struttura dati interna al S.O.
- Il sistema operativo sovraintende alla creazione dei processi
  - e alterna, nel tempo, l'esecuzione dei flussi di elaborazione ad essi associati

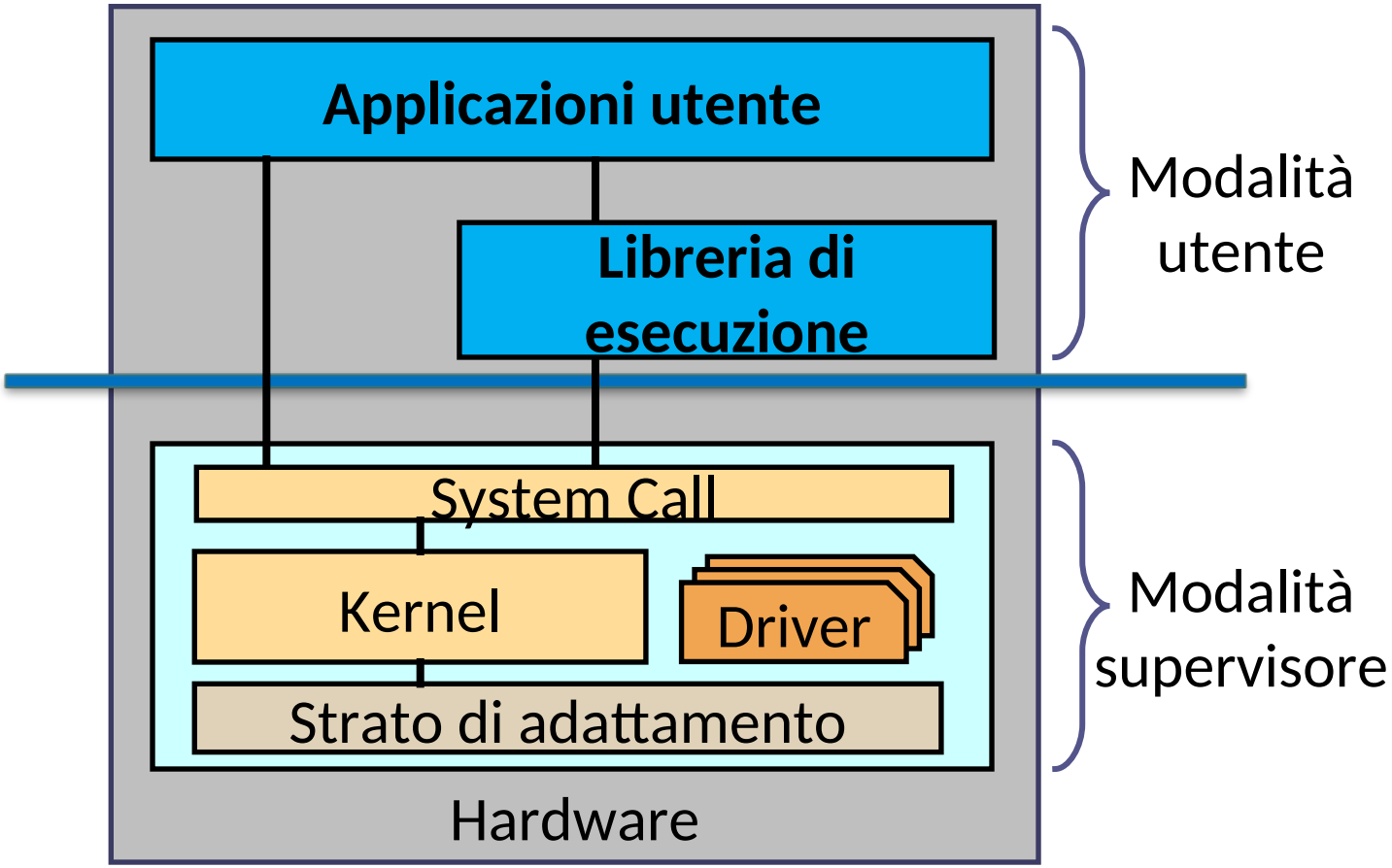
Processi

- Quando un processo viene selezionato per l'esecuzione, il S.O. :
  - configura il processore con le relative risorse (organizzazione della memoria, privilegi, periferiche, ...)
  - ripristina lo stato ad esse associato

Sistema Operativo

- Richiede un supporto hardware per distinguere due modalità di esecuzione
  - modalità **utente** – esecuzione di un sottoinsieme delle istruzioni offerte dalla CPU
  - modalità **supervisore** – accesso illimitato a tutte le funzionalità del sistema

Sistema operativo



Sistema operativo

- In modalità supervisore
  - Crea e gestisce le strutture dati che modellano i processi, il loro spazio di indirizzamento e le altre risorse che posseggono
  - Interagisce con le periferiche, il file system e la rete
- In modalità utente
  - Esegue il codice associato ai diversi processi
  - Sottostando ad un insieme di vincoli
    - Accesso parziale alla memoria ed alle istruzioni della CPU
    - Impossibilità di accedere direttamente alle periferiche

Innalzamento di privilegio

- Per permettere l'esecuzione di programmi dotati di una qualche utilità
  - i S.O. offrono meccanismi per accedere, in modo controllato, ad un insieme predefinito di funzionalità (System Call)
- Per garantire l'assenza di violazioni del modello di esecuzione
  - l'innalzamento di privilegio comporta la sostituzione dello stato della CPU e delle strutture di supporto all'esecuzione tramite istruzioni sicure

Innalzamento di privilegio

- Per i processori IA-32 ci sono due meccanismi
  - Interrupt software (trap) – più lento, funzionante anche su processori obsoleti
  - Le coppia di istruzioni SYSENTER/SYSEXIT – più efficienti, richiedono processori più recenti
- Il costo per attraversare la barriera tra modo utente e supervisore è comunque elevato
  - Nelle architetture IA-32 Windows/Linux l'operazione può richiedere 500 cicli
  - Una chiamata semplice, ne richiede 5

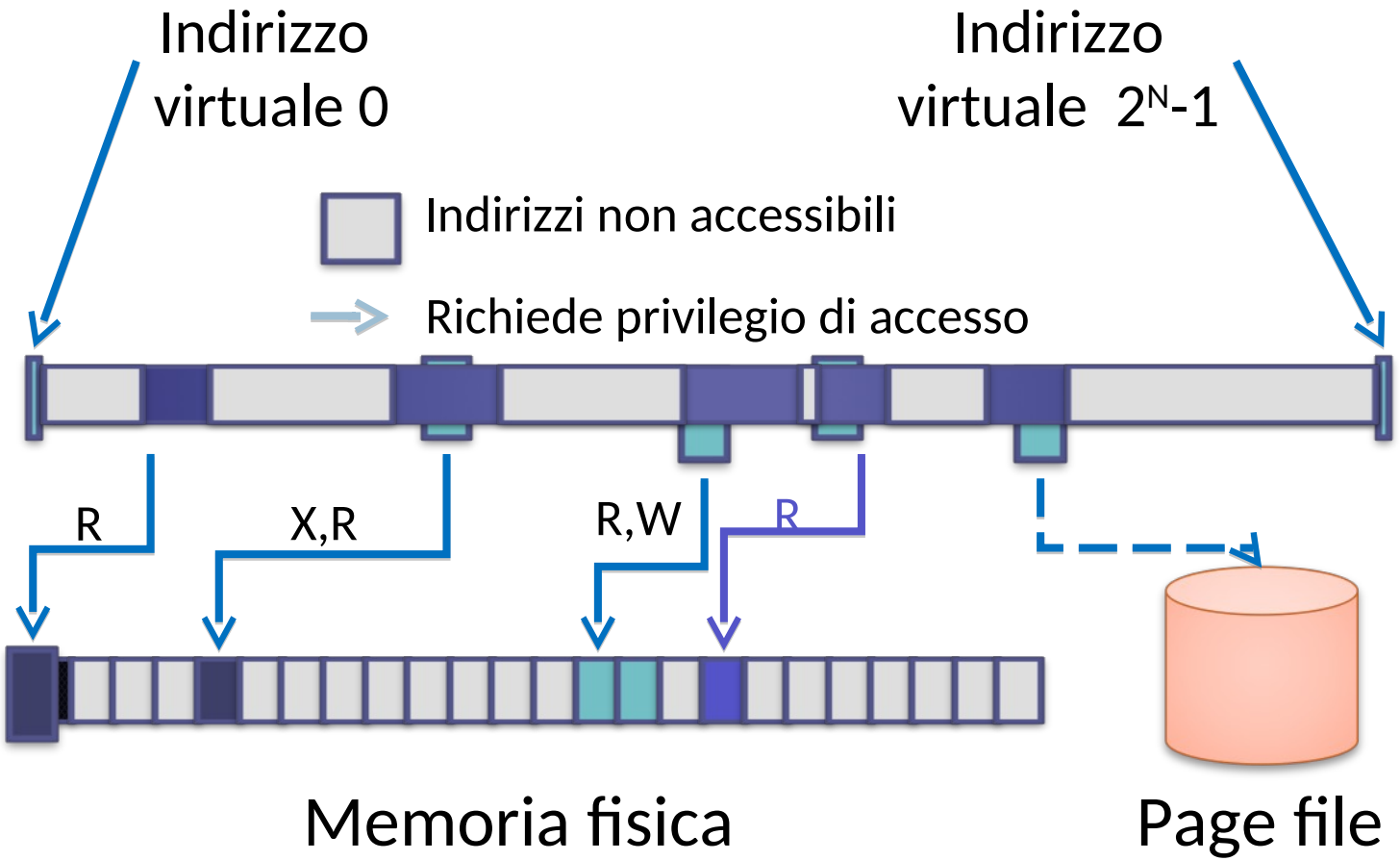
Esecuzione di un programma

- L'esecuzione di un programma comporta la creazione di un nuovo processo
  - Acquisendo le risorse necessarie
  - E inizializzandone lo stato in modo opportuno
- La creazione di un processo è articolata in diverse sotto-fasi
  - Creazione spazio di indirizzamento
  - Caricamento dell'eseguibile in memoria
  - Caricamento delle librerie
  - Avvio dell'esecuzione

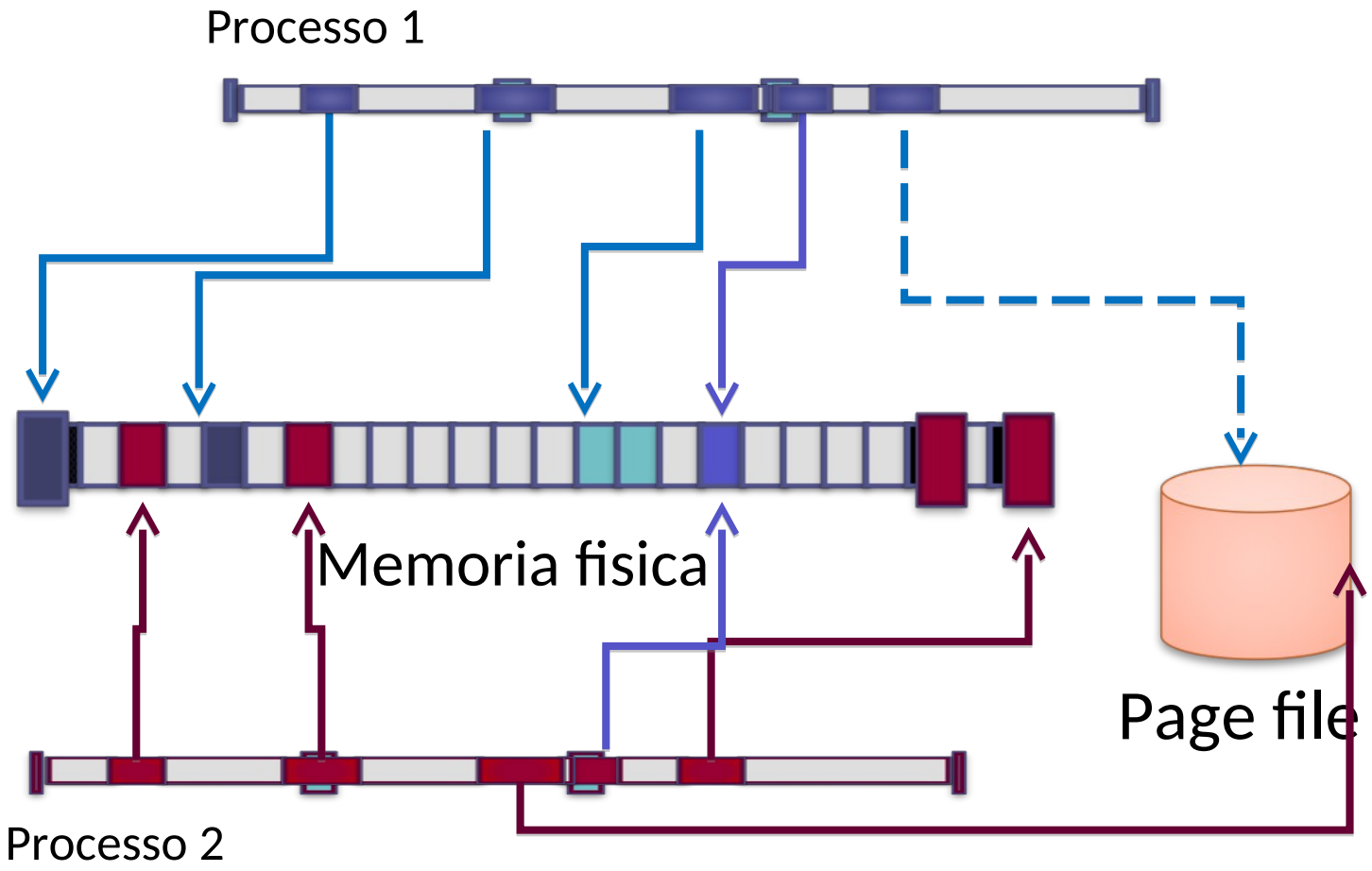
Spazio di indirizzamento

- L'esecuzione di un programma avviene nel suo spazio di indirizzamento
  - Insieme di locazioni di memoria accessibili tramite indirizzo virtuale
  - Sottoinsieme delle celle indirizzabili, gestito dal sistema operativo
- Attraverso funzionalità offerte dal blocco MMU del processore, gli indirizzi virtuali vengono tradotti in indirizzi fisici
  - Cambiando questa corrispondenza, possono essere creati innumerevoli spazi tra loro separati

Spazio di indirizzamento



Spazio di indirizzamento



Paginazione

- I processori supportano il meccanismo di page_fault
  - Tentativi di accesso ad indirizzi non mappati generano un'interruzione
  - Appoggiandosi ad una memoria persistente, si possono indirizzare contenuti molto più grossi della memoria fisica disponibile

File mapping

- Il S.O. può associare blocchi di indirizzi virtuali al contenuto di un file
  - In caso di accesso ad uno di essi, il S.O. trasferisce il corrispondente contenuto del file in una pagina fisica libera e la associa agli indirizzi relativi

Caricamento del codice eseguibile

- Il loader è il componente del S.O. responsabile di inizializzare lo spazio di indirizzamento con il contenuto del programma da eseguire
  - E di tutte le sue dipendenze
- Il file eseguibile è dotato di una struttura interna
  - Formato ELF in Linux
  - Formato PE2 in Windows
- Suddiviso in sezioni
  - Contenenti informazioni omogenee (codice, dati, ...)

Caricamento del codice eseguibile

- Nello spazio di indirizzamento si allocano una sezione per il codice ed una per i dati
  - Mappate sulle corrispondenti sezioni del file eseguibile
  - Ulteriori sezioni vengono create per stack e heap

Caricamento delle dipendenze

- Una sezione del file eseguibile è dedicata ad elencare gli ulteriori eseguibili necessari
  - Librerie dinamiche
  - Esse vengono ricorsivamente mappate nello spazio di indirizzamento
- Tutti i riferimenti a variabili/ funzioni delle dipendenze vengono aggiornati con gli indirizzi effettivi
  - Rilocazione degli indirizzi dei simboli contenuti nella tabella di importazione

Avvio dell'esecuzione

- Quando il processo di caricamento termina, è possibile avviare l'esecuzione
  - Un'apposita funzione della libreria di esecuzione è delegata a tale scopo
- Il comportamento della funzione di avvio dipende dal linguaggio di programmazione e dal S.O.
  - In generale ha il compito di garantire l'esistenza
    e la consistenza di tutte le necessarie strutture dati di supporto

La funzione di avvio

- Configurazione della piattaforma di esecuzione
  - Inizializzazione dello stack, dei registri e delle strutture dati per la gestione delle eccezioni
- Invocazione dei costruttori degli oggetti globali
- Invocazione della funzione principale
  - main(int argc, char** argv)
- Invocazione dei distruttori degli oggetti globali
  - Nell'ordine opposto a quello dei costruttori
- Terminazione del processo
  - Rilasciando l'intero spazio di indirizzamento e tutte le risorse in esse allocate, attraverso l'invocazione di un'opportuna system call (exit)

La funzione di avvio in GCC/ Linux

- Il linker indica come punto di ingresso la funzione _start()
  - Ha il compito di preparare una serie di parametri
    e di invocare la funzione __libc_start_main()

```
int __libc_start_main(
  int (*main) (int, char**,char **),
  int argc,
  char** ubp_av,
  void (*init) (void),
  void (*fini) (void),
  void (*rtld_fini) (void),
  void* stack_end
);
```


La funzione di avvio in VisualStudio/Windows

- Il linker seleziona tale funzione in base al tipo di progetto
  - L'opzione /SUBSYSTEM:WINDOWS indica un'applicazione grafica e invoca WinMainCRTStartup()
  - E' l'opposto di /SUBSYSTEM:CONSOLE
- Un ulteriore distinzione viene fatta in base al set di caratteri usato
  - ANSI o UNICODE
- Il punto di ingresso utente riflette i quattro casi
  - main(...),wmain(...),WinMain(...),wWinMain(...)

La funzione di avvio in VisualStudio/Windows

- Il comportamento è simile
  - Inizializzazione della libreria di supporto
  - Invocazione dei costruttori
  - Preparazione degli argomenti
  - Funzione principale
  - Invocazione dei distruttori
  - Terminazione del processo

Spunti di riflessione

- Si crei, in linux, un'applicazione che stampi «Hello sysprog!» e poi termini
  - Senza usare la libreria di esecuzione (opzione –nostdlib)
  - Si usi come punto di ingresso la funzione _start()