

Sistemi Operativi 1

AA 2018/2019

Componenti di un sistema operativo

Componenti di un S.O.

- Gestione dei processi
- Gestione della memoria primaria
- Gestione della memoria secondaria
- Gestione dell'I/O
- Gestione dei file
- Protezione
- Rete
- Interprete dei comandi

Gestione dei Processi

- Processo = programma in esecuzione
 - Necessità di risorse
 - Eseguito in modo sequenziale un'istruzione alla volta
 - Processi del S.O. vs. processi utente
- Il S.O. è responsabile della
 - Creazione e distruzione di processi
 - Sospensione e riesumazione di processi
 - Fornitura di meccanismi per la sincronizzazione e la comunicazione tra processi



Gestione della Memoria Primaria

- Memoria primaria conserva dati condivisi dalla CPU e dai dispositivi di I/O
 - Un programma deve essere caricato in memoria per poter essere eseguito
- Il S.O. è responsabile della
 - Gestione dello spazio di memoria (quali parti e da chi sono usate)
 - Decisione su quale processo caricare in memoria quando esiste spazio disponibile
 - Allocazione e rilascio dello spazio di memoria



Gestione della Memoria Secondaria

- Memoria primaria è volatile e “piccola”
 - Indispensabile memoria secondaria per mantenere grandi quantità di dati in modo permanente
- Tipicamente uno o più dischi (magnetici)
- Il S.O. è responsabile della
 - Gestione dello spazio libero su disco
 - Allocazione dello spazio su disco
 - Scheduling degli accessi su disco



Gestione dell'I/O

- Il S.O. nasconde all'utente le specifiche caratteristiche dei dispositivi di I/O
- Il sistema di I/O consiste di
 - Un sistema per accumulare gli accessi ai dispositivi (buffering)
 - Una generica interfaccia verso i device driver
 - Device driver specifici per alcuni dispositivi

Gestione dei File

- Le informazioni sono memorizzate su supporti fisici diversi (dischi, DVD, memory-stick, ...) controllati da driver con caratteristiche diverse
- **File** = astrazione logica per rendere conveniente l'uso della memoria non volatile
 - Raccolta di informazioni correlate (dati o programmi)
- Il S.O. è responsabile della
 - Creazione e cancellazione di file e directory
 - Supporto di primitive per la gestione di file e directory (copia, sposta, modifica, ...)
 - Corrispondenza tra file e spazio fisico su disco
 - Salvataggio delle informazioni a scopo di backup



Protezione

- Meccanismo per controllare l'accesso alle risorse da parte di utenti e processi
- Il S.O. è responsabile della:
 - Definizione di accessi autorizzati e non
 - Definizione dei controlli da imporre
 - Fornitura di strumenti per verificare le politica di accesso

Rete (Sistemi Distribuiti)

- Sistema distribuito = collezione di elementi di calcolo che non condividono né la memoria né un clock
 - Risorse di calcolo connesse tramite una rete
- Il S.O. è responsabile della gestione “in rete” delle varie componenti
 - Processi distribuiti
 - Memoria distribuita
 - File system distribuito
 - ...



Interprete dei comandi (Shell)

- La maggior parte dei comandi vengono forniti dall'utente al S.O. tramite “istruzioni di controllo” che permettono di:
 - Creare e gestire processi
 - Gestire l'I/O
 - Gestire il disco, la memoria, il file system
 - Gestire le protezioni
 - Gestire la rete
- Il programma che legge ed interpreta questi comandi è l'interprete dei comandi
 - Funzione: leggere ed eseguire la successiva istruzione di controllo (comando)



Interprete

- Principalmente ha il compito di prendere i comandi specificati dall'utente e di eseguirli
 - Codice comandi nell'interprete
 - Codice comandi in programmi predefiniti, da riferire semplicemente con il nome del programma (es. rm)
 - Se sono predefiniti per aggiungere nuovi comandi non è necessario modificare la shell

Interfaccia utente

- L'interprete puo' fornire diversi tipo di interfaccia utente
 - Command-Line (CLI), Graphics User Interface (GUI), Batch
- CLI permette di digitare direttamente comandi testo (es. UNIX)
 - Spesso implementata nel kernel, alcune volte come programma di sistema
 - Spesso è implementata in diverse varianti – **shells** (c, bourne, etc.)

Interfacce Utente di SO - GUI

- La metafora del **desktop** come interfaccia utente
 - Generalmente legata a mouse, tastiera e schermo
 - **I**cone rappresentano i file, programmi, azioni, etc
 - I vari tasti del mouse puntati sull' oggetto visualizzato nell' interfaccia possono eseguire diverse azioni (fornire informazioni, opzioni, eseguire funzioni, aprire directory (noti come **f**older))

System Call

- L'utente usa la shell, ma i processi?
 - Le system call forniscono l'interfaccia tra i processi e i servizi offerti dal S.O.
- Tipicamente scritte in linguaggio di alto livello (C o C+
+), qualcuna anche in assembler

Esempio di system calls

file di origine

file di destinazione

Esempio di ciclo esecutivo di una chiamata di sistema

- Acquisisce il nome del file in ingresso
 - Scriva messaggio di richiesta sullo schermo
 - Accetta i dati in ingresso
- Acquisisce il nome del file in uscita
 - Scriva messaggio di richiesta sullo schermo
 - Accetta i dati in ingresso
- Apri il file in ingresso
 - Se il file non esiste, termina con errore
- Crea il file in uscita
 - Se il file esiste, termina con errore
- Ripete
 - Legge dal file in ingresso
 - Scriva sul file in uscita
- Finché c'è ancora da leggere
- Chiude il file in uscita
- Scriva messaggio sullo schermo per informare del completamento
- Termina senza errori



DTrace

```
# ./all.d 'pgrep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
0 -> XEventsQueued U
0 -> _XEventsQueued U
0 -> _X11TransBytesReadable U
0 <- _X11TransBytesReadable U
0 -> _X11TransSocketBytesReadable U
0 <- _X11TransSocketBytesreadable U
0 -> ioctl U
0 -> ioctl K
0 -> getf K
0 -> set_active_fd K
0 <- set_active_fd K
0 <- getf K
0 -> get_udatamodel K
0 <- get_udatamodel K
...
0 -> releasef K
0 -> clear_active_fd K
0 <- clear_active_fd K
0 -> cv_broadcast K
0 <- cv_broadcast K
0 <- releasef K
0 <- ioctl K
0 <- ioctl U
0 <- _XEventsQueued U
0 <- XEventsQueued U
```


Implementazione delle System Call

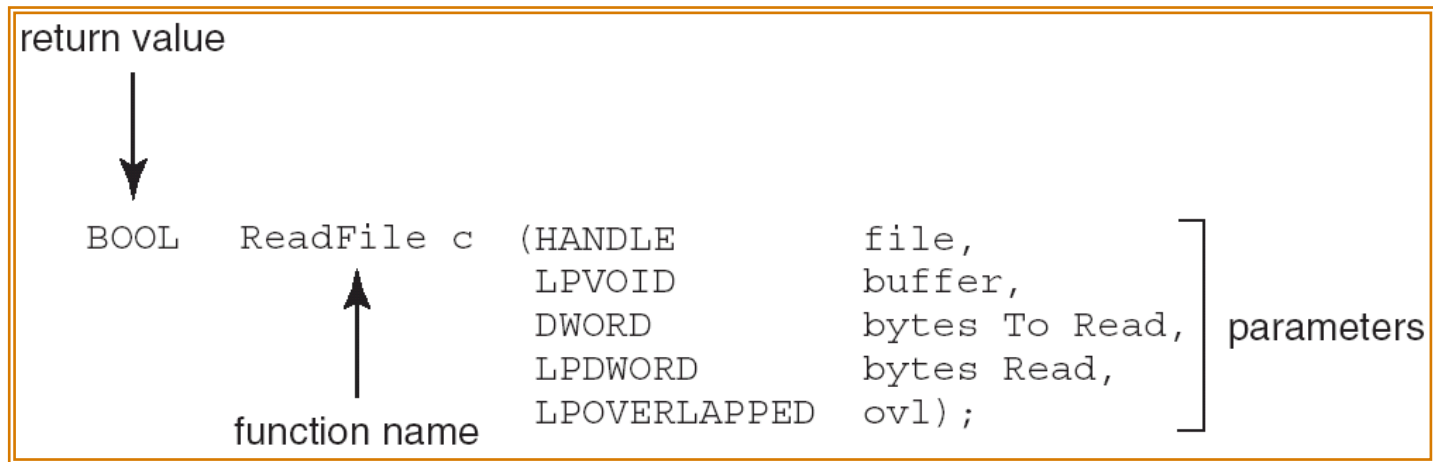
- Tipicamente, un numero associato ad ogni system call
 - L' interfaccia alle chiamate di sistema mantiene una tabella indicizzata secondo questi numeri
- L' interfaccia invoca la system call usata, nel kernel del SO e poi ritorna lo stato della system call e gli eventuali valori di ritorno
- Il chiamante non ha necessità di conoscere nulla di come la system call è implementata

System Call

- Per mascherare i dettagli implementativi il S.O. fornisce un livello intermedio.
- Per lo più chiamate da programmi attraverso l'interfaccia per la programmazione di applicazioni (*Application Program Interface (API)*) di alto livello piuttosto che usate direttamente

Tipico esempio di API

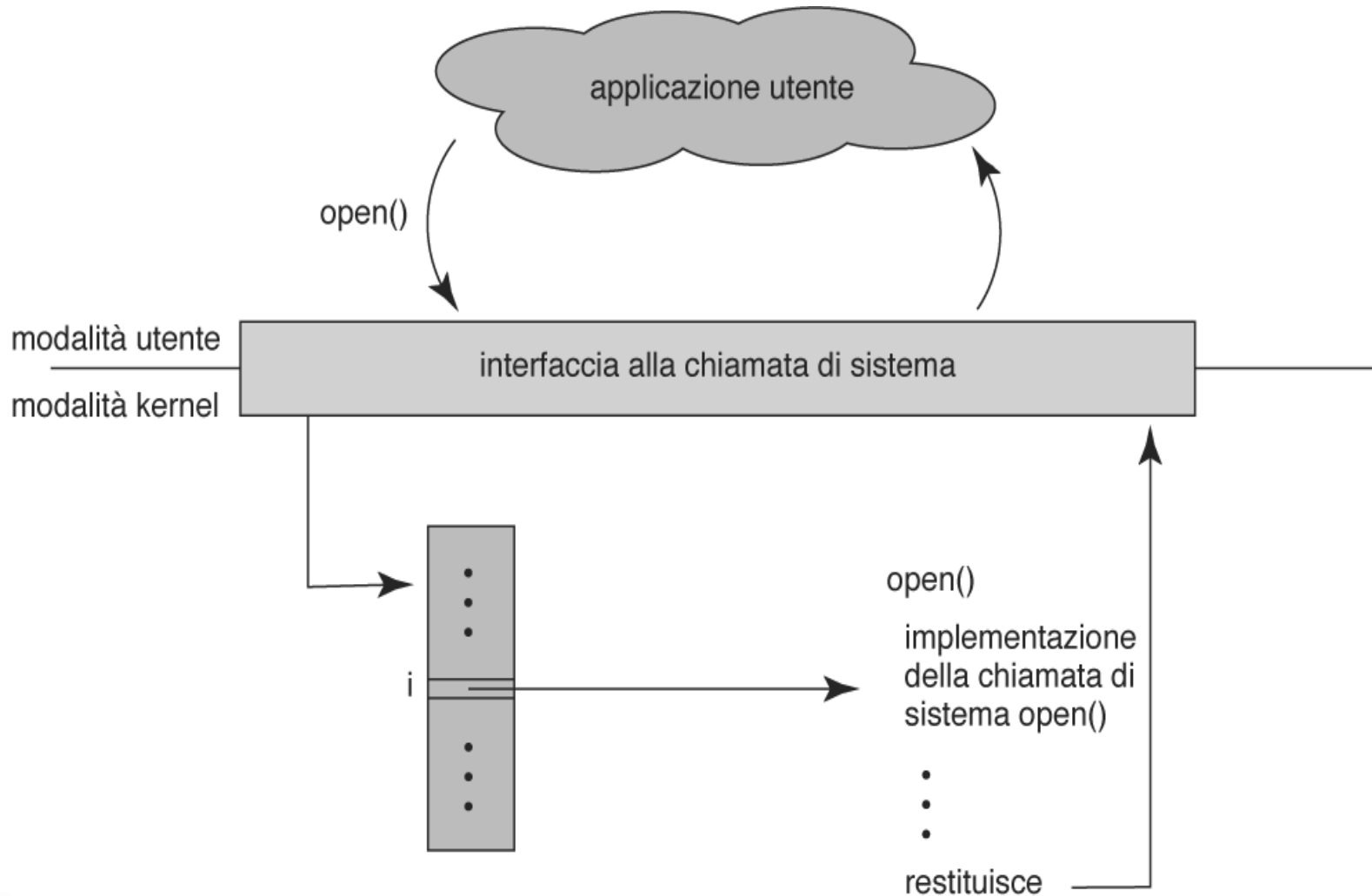
- Consideriamo la funzione ReadFile()
- Win32 API— la funzione per leggere da un file



Tipico esempio di API

- Descrizione dei parametri passati alla `ReadFile()`
 - `HANDLE` file— il file da leggere
 - `LPVOID` buffer—un buffer di interposizione tra letture/scritture
 - `DWORD` bytesToRead—il numero di bytes da leggere dal buffer
 - `LPDWORD` bytesRead—il numero di bytes letti nell' ultima read
 - `LPOVERLAPPED` ovl—indica se è stato usato spooling

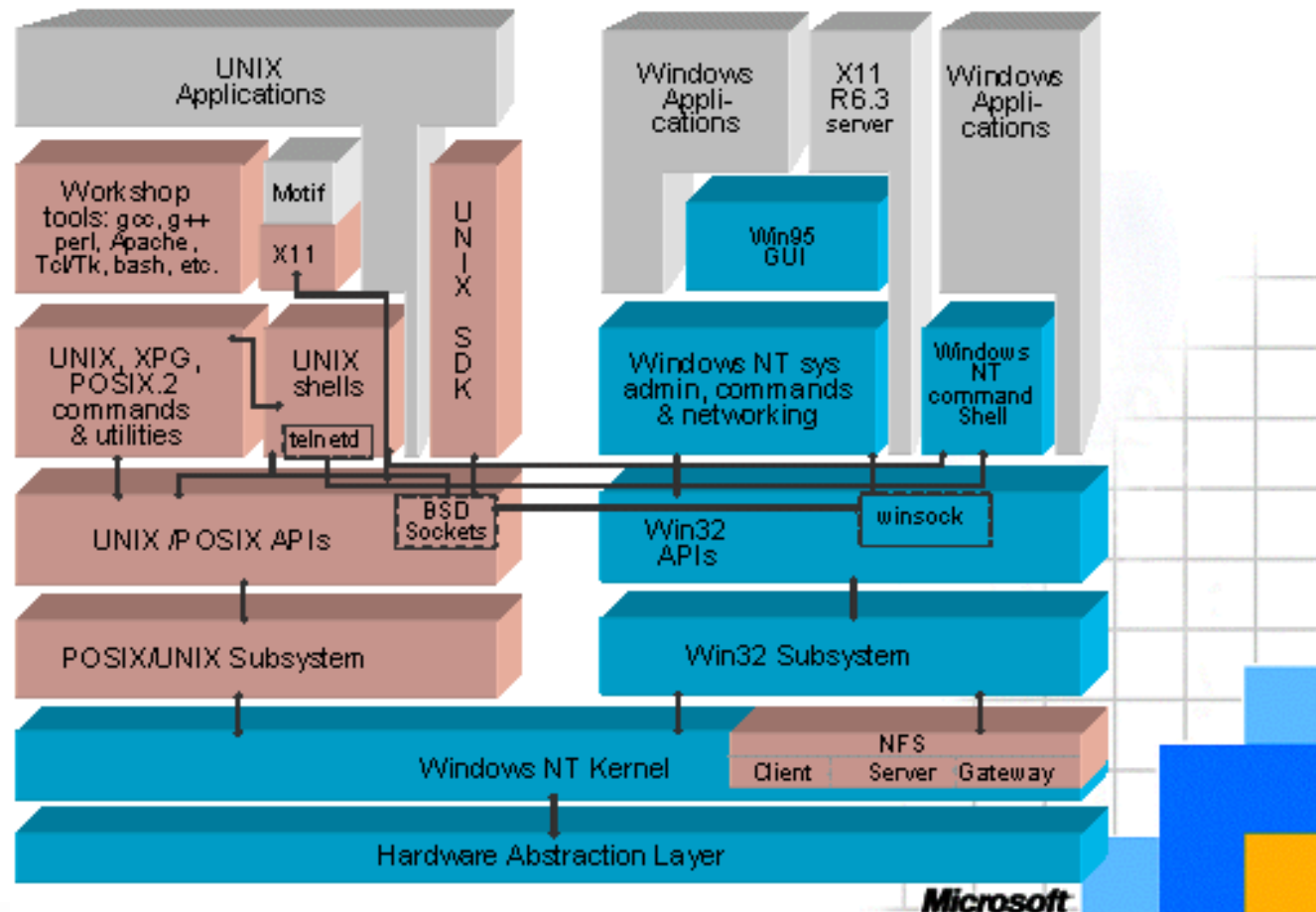
open() chiamata da un programma utente



API

- Le 2 APIs piú comuni sono Win32 API per Windows, POSIX API per sistemi POSIX (Portable Operating-System Interface for Unix)-based (che includono di fatto tutte le versioni di UNIX, Linux, e Mac OS X), e le Java API per la Java virtual machine (JVM)
- Dovrebbero garantire portabilitá' delle applicazioni (almeno sulle stesso tipo di API)




Win32 APIs



Example of standard API: read()

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count)
```

		
return value	function name	parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer



System Call

- Opzioni per la comunicazione tra il S.O. e un processo:
 - Passare i parametri (della system call) tramite registri
 - Passare i parametri tramite lo stack del programma
 - Memorizzare i parametri in una tabella in memoria
 - L'indirizzo della tabella è passato in un registro o nello stack

Passaggio di parametri nello stack

- Chiamata alla funzione di libreria $A(x)$
- Parametro x nello stack
 - Invocazione della vera system call $_A$ corrispondente ad A
 - $_A$ mette il numero di system call in un punto noto al S.O.
 - $_A$ esegue una TRAP (interruzione non mascherabile)
 - Effetto: passaggio da Modo User a Modo Kernel
 - Inizia l'esecuzione ad un indirizzo fisso (gestore interrupt)
 - Il S.O., in base al numero di system call, smista la chiamata al corretto gestore che viene eseguito
 - Una volta terminato, il controllo viene restituito al programma di partenza (funzione di libreria $A()$)



Passaggio di parametri nello stack

```
void main() {
```

```
...
```

```
  A(x);
```

```
...
```

```
}
```

Programma utente
(user mode)

```
A(int x) {
```

```
...
```

```
  push x
```

```
  _A()
```

```
...
```

```
}
```

```
_A() {
```

```
  scrivi 13
```

```
  TRAP
```

```
...
```

```
}
```

```
Leggi 13
```

```
Salta al gestore 13
```

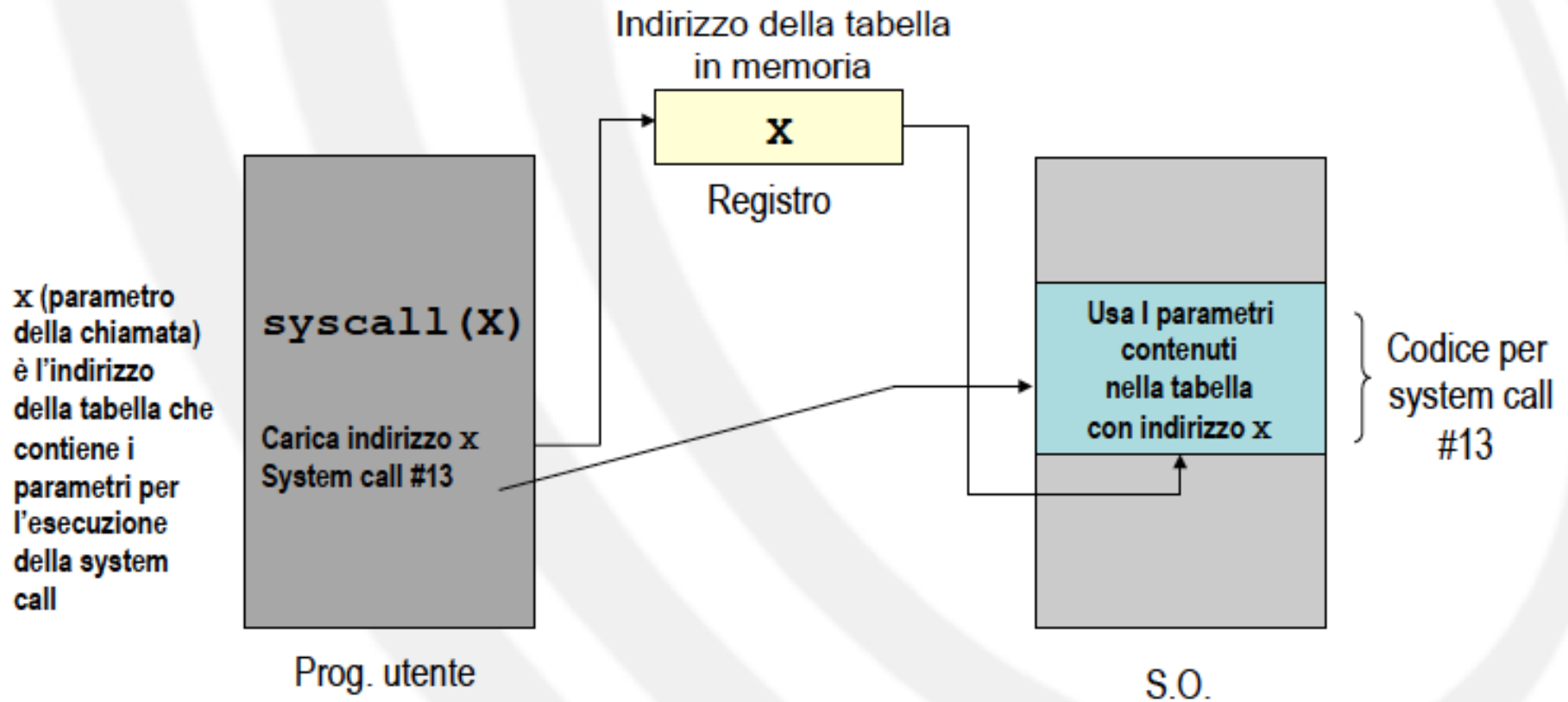
```
handler_13 () {
```

```
...
```

```
}
```

Sistema operativo
(kernel mode)

Passaggio di parametri tramite tabella



Programmi di sistema

- La vista utente delle operazioni di un sistema avviene tipicamente in termini di programmi di sistema (e non di system call)
 - Gestione/manipolazione dei file (crea, copia, cancella, ...)
 - Informazioni sullo stato del sistema (data, memoria libera, ...)
 - Strumenti di supporto alla programmazione (compilatori, assembleri, ...)
 - Formattazione documenti
 - Mail
 - Programmi di gestione della rete (login remoto, ...)
 - Interprete dei comandi
 - Utility varie



Riassumendo

- Servizi di un S.O.:
 - Esecuzione di programmi
 - Operazioni di I/O
 - Manipolazione del file system
 - Comunicazione
 - Memoria condivisa
 - Scambio di messaggi
 - Rilevamento di errori (logici e fisici)
 - Allocazione delle risorse
 - Contabilizzazione delle risorse
 - Protezione e sicurezza

