TPS

(Tecnologie e Progettazione di Sistemi Informatici e di Telecomunicazioni)

Modulo 4

Il Sistema Operativo

Istituto Tecnico - Indirizzo Informatica & Telecomunicazioni
Articolazione Informatica - Classi Terze

Panoramica

Generalità sui sistemi operativi

All'accensione, il primo codice ad essere eseguito (BIOS) è contenuto nella ROM ed effettua:

1. POST (Power On Self Test) dei vari moduli componenti il computer
   - scheda madre
   - dimensione memoria RAM a disposizione
   - tastiera, mouse, ecc
   - hard disk, eventuali stampante, modem, ecc
2. Caricamento del kernel (nucleo del sistema operativo) in RAM

Generalità sui sistemi operativi

A questo punto, il sistema è pronto ad eseguire i programmi utente.

Definizione: Il sistema operativo è un insieme di programmi che gestiscono il funzionamento del computer agendo come intermediario fra l'utente ed il calcolatore.

Su una data macchina si possono anche installare più sistemi operativi partizionando opportunamente il disco rigido.

Funzioni del sistema operativo

Il S.O. svolge fondamentalmente due funzioni:

1. Gestisce le risorse HW (CPU, memoria, periferiche)
2. Fornisce il supporto per l'esecuzione dei comandi utente

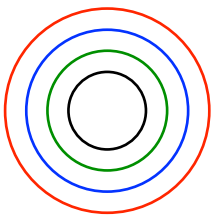
Il S.O. è memorizzato sull'hard disk e solo la sua parte principale (kernel o nucleo) risiede in memoria; gli altri moduli vengono caricati solo quando necessario.

Struttura del sistema operativo

Il S.O. si può immaginare come successivi strati (onion skins) avvolti concentricamente intorno all'HW.

Dall'esterno verso l'interno abbiamo quindi:

- Shell
- Funzioni di base
- Nucleo
- Hardware



L'utente quindi interagisce solo con la shell, che nasconde i dettagli degli strati sottostanti.

Il kernel

Interagisce con i programmi applicativi che quindi non possono accedere direttamente all'HW.

Offre un insieme di funzionalità dette *primitive di sistema.*

Questa organizzazione permette all'utente di non doversi preoccupare dei dettagli dell'HW, che sono mascherati dal S.O.

La shell

È un programma che permette all'utente di interagire in modo (più o meno) naturale ed intuitivo con la macchina.

Viene anche detta interfaccia utente.

Vi sono vari stili di interfacce utente:

- CLI (Command Line Interface)
  (es.: le varie shell di UNIX)
- GUI (Graphical User Interface)
  (es.: Windows, OS X, Linux)
- CUI (Conversational User Interface)
  (il futuro ...)

I S.O. più diffusi

Per gestire reti (server di rete):

- Linux
- Windows Server

Per PC:

- Windows
- OS X
- Linux

Per mini e mainframe:

- UNIX
- Linux
- Altri sistemi operativi (proprietari)

I S.O. più diffusi

In realtà, i più diffusi S.O. sono quelli per dispositivi mobili (smartphone/tablet)

Solo nel 3Q2013 ne sono state vendute 261M unità così ripartite:

- Android 81%
- iOS 12.9%
- Windows Phone 3.6%
- Blackberry 1.6%

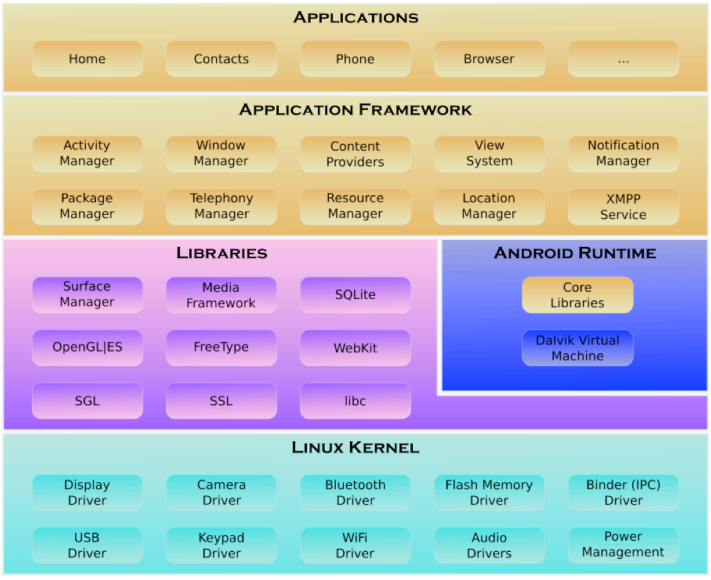
In particolare, Android viene utilizzato anche su:

PC portatili, netbook, lettori mp3/mp4, lettori di ebook, fotocamere, smart TVs, smart watches, smart glasses, navigatori, console di gioco, telecamere, frigoriferi (!), specchi (!!!)

Android - architettura

- HW: basato su architettura ARM (x86 su alcuni apparecchi)
- Kernel: derivato da Linux con pesanti modifiche per adattarlo ad un ambiente operativo molto diverso da quello di un normale computer
- Applicazioni: principalmente scritte in Java, eseguite all'interno di una Virtual Machine (Dalvik, non Java, per motivi di compattezza ed efficienza)

Android - architettura



Android - ciclo di vita

- Ogni versione, oltre al numero di release, ha un codename corrispondente ad un dolcetto, in ordine alfabetico (cupcake, donut, eclair, froyo, gingerbread, ice cream sandwich, honeycomb, jelly bean, kit kat, ecc)
- Google rilascia i sorgenti per i suoi apparecchi (es.: Google Nexus)
- Gli altri produttori, se vogliono allinearsi, devono portare la nuova release sui loro apparecchi (ormai venduti e vecchi ...)

Evoluzione dei sistemi operativi

Lettura autonoma pag. 179-190 del testo

La gestione del processore

- Programma

  Entità statica, è il codice memorizzato su memoria di massa
- Processo

  Entità dinamica, istanza in continua evoluzione di un programma, residente in RAM.
  È costituito da
  - Codice (non modificabile => può essere condiviso fra più processi che eseguono lo stesso programma)
  - Dati (un set di dati per ogni istanza di uno stesso programma)

Multitasking e Multiprocessing

- Multitasking

  Esecuzione di più processi da parte di un unico processore. Il parallelismo che si ottiene è solo apparente.
- Multiprocessing

  Multitasking distribuito su più processori. Il parallelismo è reale.

Gestione dei processi

- Process Descriptor (PD) o Process Control Block (PCB)

  È la struttura dati che contiene tutte le informazioni necessarie al SO per gestire l'esecuzione in parallelo dei vari processi
- Stato dei processi

  Una fra le cinque possibili situazioni in cui si può trovare un processo in esecuzione

Stato dei processi

1. New

   Appena creato (1 volta sola nella sua vita)
2. Running

   La CPU sta eseguendo il codice del processo (1 per processore)
3. Waiting

   In attesa che si verifichi un evento (che si liberi una risorsa)
4. Ready-to-Run

   Ha tutte le risorse, tranne la CPU
5. Terminated

   Finito di girare, in attesa di liberare le risorse prima di farlo scomparire (1 volta sola nella sua vita)

Process Control Block (PCB)

Il PCB contiene, fra l'altro, le seguenti informazioni fondamentali per gestire ciascun processo:

- PID - l'identificatore del processo
- Stato - lo stato attuale del processo
- PC - il Program Counter
- Registri - i registri della CPU
- Priorità - la priorità attuale del processo
- Puntatori alla memoria del processo
- Puntatori alle risorse allocate dal processo (p.es. i file aperti)

Process Scheduling

Per scheduling si intende la politica di scelta dell'assegnazione della CPU ai processi in Ready List.

Al cambio di processo running il *dispatcher* effettua un *context switch:*

- salva il contenuto dei registri nel PCB del processo che lascia la CPU
- recupera il contenuto dei registri dal PCB del processo che deve impadronirsi della CPU

User mode e Kernel mode

User mode e Kernel (o supervisor) mode sono due diversi livelli di privilegio del codice in esecuzione.

- User mode: quando si esegue il codice del programma utente; in generale, il processo è interrompibile
- Kernel mode: quando si esegue il codice del routines interne del S.O. oppure le chiamate al sistema; in generale, il processo non è interrompibile

Politiche di scheduling

Windows XP:

- CPU scheduling con prelazione
- 32 livelli di priorità in Round Robin
- Due classi di priorità:
  - Real time (16..32)
  - Variable (1..15)
    - La priorità si abbassa all'esaurirsi del quanto di tempo
    - in seguito ad uno sblocco, la priorità viene aumentata (boosted)
    - i processi in foreground sullo schermo hanno una priorità più alta

Politiche di scheduling

Linux:

- CPU scheduling con prelazione
- 140 livelli di priorità
- Tre classi di priorità:
  - Real time 0..99
    (FCFS non pre-empted oppure RR pre-empted)
  - System thread 0..99
    (FCFS non pre-empted)
  - User 100..139
    (RR Pre-empted) priorità calcolata dinamicamente in base al grado di interattività

Comunicazione fra processi

I metodi più diffusi di comunicazione fra processi sono basati su:

- Memoria condivisa (shared memory) ossia variabili o strutture dati condivise da più processi
  metodo usato tipicamente fra processi residenti su di una stessa macchina
- Scambio di messaggi (message passing) ossia invio di messaggi su canali di comunicazione
  Può essere usato anche fra processi residenti su macchine diverse

La gestione della memoria

Classificazione delle memorie per tempo tipico di accesso e capacità:

Tempo		**Capacità**
1 ns	Registri	< 1 kB
qualche ns	Cache	1-8 MB
10 ns	RAM	2-8 GB
10 ms	Dischi magnetici	1-10 TB
100 s	Nastri magnetici	1-100 TB

Il memory manager

Il memory manager si occupa principalmente di gestire l'utilizzo della RAM da parte dei processi.

- Sapere quali parti sono in uso e quali libere
- Allocare e deallocare le porzioni di memoria necessarie ai processi
- Gestire lo *swapping* tra RAM e disco quando la RAM non è sufficiente

Indirizzi logico e fisico

Gli indirizzi di memoria generati da compilatore e linker devono essere *rilocabili* (indirizzi logici).

Al momento dell'attivazione di un nuovo processo, il *loader* trasforma tali indirizzi negli indirizzi *fisici* effettivi.

ind. fisico = ind. logico + offset

Tale operazione prende il nome di *rilocazione* e può essere di due tipi:

- Rilocazione *statica*: l'offset è determinato una volta per tutte alla partenza del processo
- Rilocazione *dinamica*: l'offset è fornito dal contenuto di un registro (registro di rilocazione) e può quindi variare durante la vita del processo

Indirizzi logico e fisico

Ciascun processo deve quindi "vedere" uno spazio di indirizzamento contiguo.

Alcune osservazioni:

- I programmi attuali hanno dimensioni spesso notevoli
- Il continuo caricamento e scaricamento di programmi porta ad una frammentazione della memoria
- Spesso solo una parte delle istruzioni (e dello spazio dati) di un programma vengono realmente utilizzate

Per ovviare a questi inconvenienti si possono adoperare diverse tecniche.

Swapping, DLL, Overlay

Swapping

Per liberare dello spazio di memoria RAM si possono scaricare su disco dei processi temporaneamente inattivi.

Si tratta però di un'operazione molto onerosa e lenta, da attuarsi solo in casi estremi.

DLL (Dynamic Link Library)

Un programma molto ricco di funzionalità può essere suddiviso in un blocco principale di codice sempre presente in RAM ed in alcune DLL che restano su disco e vengono caricate in RAM solo se necessario.

Swapping, DLL, Overlay

Overlay

Se un programma contiene zone di codice che si escludono a vicenda, nel senso che non servono contemporaneamente, il programmatore può destinare loro la stessa zona di memoria, in cui quindi tali zone di codice sono sovrapposte (overlay); a run-time il loader caricherà solo la parte necessaria.

Allocazione della memoria

Partizionamento

La memoria viene suddivisa in *partizioni* che possono essere di dimensione fissa o variabile.

Si tratta di tecniche di gestione dell'allocazione della memoria (descritte sul libro a pagg. 213-216) piuttosto complesse e con limitazioni tali (limitazione del numero dei processi, frammentazione e deframmentazione della memoria, ricerca del miglior segmento cui associare un nuovo processo, lento e progressivo declino delle prestazioni, ecc).

Per questi motivi sono state superate dai più moderni sistemi a *Memoria Virtuale.*

Allocazione della memoria

Memoria Virtuale

Il SO mantiene in memoria solo le parti di codice e dati in uso e lascia su disco tutto il resto.

Questa strategia risulta vincente grazie al *principio di località*.

Lo spazio di indirizzamento di un programma e la dimensione del suo codice non sono quindi più vincolati dalla dimensione fisica della memoria.

Il programmatore può quindi scrivere il proprio codice disponendo di uno spazio di memoria (*virtuale*, non reale) limitato solo dall'architettura della CPU (p.es. 4GB per CPU a 32 bit).

Memoria Virtuale - Paginazione

Sia i programmi sia la memoria centrale vengono divisi in *pagine* di dimensione fissa.

La memoria risulta quindi suddivisa in un numero fisso di *pagine fisiche.*

Ciascuna *pagina fisica* può essere libera oppure occupata da una pagina di codice o dati di un determinato processo.

Poiché una stessa pagina di codice può essere caricata in indirizzi diversi in tempi diversi, il codice deve essere *rilocabile dinamicamente*.

Memoria Virtuale - Paginazione

La mappatura degli indirizzi da virtuale a fisico viene eseguita da un dispositivo HW apposito integrato nella CPU: la MMU (Memory Management Unit).

La struttura dati che contiene le informazioni necessarie per la mappatura si chiama *Page Table.*

La mappatura è totalmente a carico del SO, perciò risulta invisibile al SW applicativo.

Particolarmente importante il caso del *Page Fault* ed il suo trattamento.

La paginazione elimina il problema della frammentazione della memoria ed è usata nella maggior parte dei SO moderni.

Memoria Virtuale - Paginazione

A ciascuna pagina fisica sono associati opportuni flag, fra cui:

- *locked* (aka *pinned*, *wired*, *fixed*): la pagina deve rimanere sempre in memoria (p.es. il codice che gestisce i page fault!)
- *modified:* la pagina è stata modificata (solo pagine dati)
- *referenced:* la pagina è stata recentemente usata (resettato periodicamente)

Tali flag vengono usati dagli algoritmi di sostituzione delle pagine (NRU, LRU) quando è necessario eliminare una pagina per fare spazio ad un'altra

Il File System

Struttura e realizzazione del File System

La sicurezza del File System

La gestione dell'I/O

Video

1. Computer Operating Systems: Managing Hardware and Software Resources (basic)
2. OS Functions: Security, System Management, Communication and Hardware & Software Services (basic)
3. Enterprise, Workgroup & Personal Operating Systems
4. Files Systems: FAT, NTFS, and HFS+