Struttura di un SO

Sistemi Operativi

Antonino Staiano
Email: antonino.staiano@uniparthenope.it

Introduzione

- Funzionamento di un SO
- Struttura di un SO
- SO con struttura monolitica
- Progettazione a strati di un SO

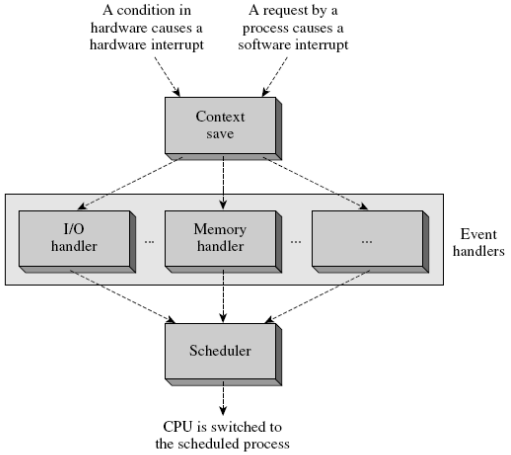
Introduzione (cont.)

- SO con macchina virtuale
- SO basati su kernel
- SO basati su micro-kernel
- Casi di studio

Funzionamento di un SO

- Quando un computer è avviato, è eseguita una procedura di *boot*
  - Analizza la sua configurazione, tipo di CPU, dimensione della memoria, dispositivi di I/O e dettagli di altro hardware
  - Carica parte del SO in memoria, inizializza le strutture dati e passa ad esso il controllo del sistema
- Durante il funzionamento del computer, possono verificarsi delle interruzioni causate da:
  - Un evento: completamento di un'operazione di I/O; conclusione di uno slot temporale
  - Una chiamata di sistema fatta da un processo (*interruzione software*)
- Routine di servizio di un'interruzione
  - Esegue il salvataggio del contesto
  - Attiva il gestore di eventi
- Lo scheduler seleziona un processo da servire

Funzionamento di un SO (cont.)



Panoramica del funzionamento del SO

Funzionamento di un SO (cont.)

Function	Description
Process management	Initiation and termination of processes, scheduling
Memory management	Allocation and deallocation of memory, swapping, virtual memory management
I/O management	I/O interrupt servicing, initiation of I/O operations, optimization of I/O device performance
File management	Creation, storage and access of files
Security and protection	Preventing interference with processes and resources
Network management	Sending and receiving of data over the network

Struttura di un SO

- Politiche e meccanismi
- Portabilità ed Estensibilità di un SO

Politiche e meccanismi

- Nel determinare come il SO esegue una funzione, il progettista deve pensare a due livelli distinti
  - Politica: principio in base al quale il SO esegue la funzione
    - Decide cosa dovrebbe esser fatto
  - Meccanismo: azione necessaria per implementare una politica
    - Determina come farlo e la attua
  - Esempio:
    - Lo scheduling round-robin è una politica
    - Meccanismo: mantiene una coda di processi pronti e fa il dispatch di un processo

Portabilità ed Estensibilità dei SO

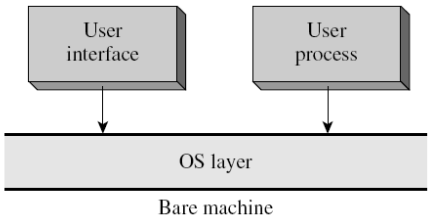
- Porting: adattare il software per usarlo in un nuovo sistema di computer
- Portabilità: semplicità con cui un programma può essere portato
  - Inversamente proporzionale allo sforzo del porting
- Porting di un SO: cambiare parti del suo codice che dipendono dall'architettura per funzionare con il nuovo HW
  - Esempi di dati e istruzioni dipendenti dall'architettura in un SO:
    - vettore delle interruzioni, informazioni di protezione della memoria, istruzioni di I/O, ecc.

Portabilità ed Estensibilità dei SO (cont.)

- Estendibilità: facilità con cui possono essere aggiunte nuove funzionalità ad un sistema software
  - L'estendibilità di un SO è necessaria per due scopi:
    - Incorporare nuovo HW in un sistema
      - Tipicamente nuovi dispositivi di I/O o adattatori di rete
    - Fornire nuove caratteristiche per soddisfare nuove pretese degli utenti
  - I primi SO non fornivano alcun tipo di estendibilità
  - I SO moderni facilitano l'aggiunta di un driver di dispositivo
    - Forniscono anche capacità *plug-and-play*

SO con Struttura Monolitica

- I primi SO avevano una struttura monolitica
  - Il SO costituiva un singolo strato software tra l'utente e la nuda macchina (hardware)



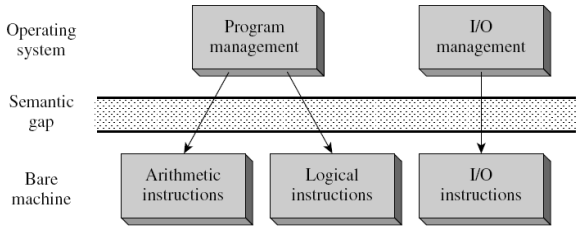
SO monolitico

SO con Struttura Monolitica (cont.)

- Problemi della struttura monolitica
  - Il solo SO aveva un'interfaccia con l'HW
    - Codice dipendente dalla macchina distribuito in tutto il SO
      - Cattiva portabilità
    - Rendeva il testing ed il debugging difficoltoso
      - Elevati costi di manutenzione e potenziamento
- Modi alternativi per strutturare un SO
  - Struttura a strati
  - Struttura basata su kernel
  - Struttura basata su microkernel

Progettazione a Strati dei SO

- **Gap semantico**: discrepanza tra la natura delle operazioni necessarie nell'applicazione e la natura delle operazioni fornite nella macchina

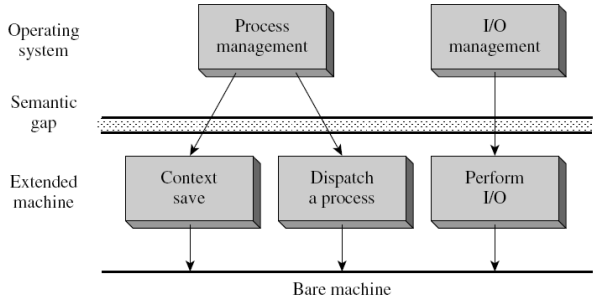


Gap semantico

- Il gap semantico è ridotto:
  - Usando una macchina con maggiori capacità
  - Simulando una *macchina estesa* in uno strato inferiore

Progettazione a Strati dei SO (cont.)

- Le routine di uno strato devono usare solo i servizi dello strato immediatamente inferiore
  - Solo tramite le sue interfacce



Progetto di un SO a strati

Esempio: struttura sistema multi-programmato

Strati nel sistema multi-programmato

Layer	Description
Layer 0	Processor allocation and multiprogramming
Layer 1	Memory and drum management
Layer 2	Operator–process communication
Layer 3	I/O management
Layer 4	User processes

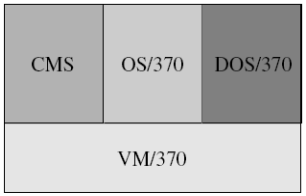
Esempio: struttura sistema multi-programmato (cont.)

- Problemi
  - Il funzionamento del sistema è rallentato dalla struttura stratificata
  - Difficoltà nel sviluppare un progetto a strati
    - Problema: ordinamento degli strati che richiedono ciascuno i servizi dell'altro
      - Spesso risolto suddividendo uno strato in due e mettendo altri strati tra le due metà
  - Stratificazione delle funzionalità del SO
    - Progettazione complessa
    - Perdita di efficienza nell'esecuzione
    - Limitata estendibilità

SO con Macchina Virtuale

- Classi di utenti differenti hanno bisogno di diversi tipi di servizi utente
- Un SO con Macchina Virtuale (SO VM) crea numerose macchine virtuali
  - Una macchina virtuale è una risorsa virtuale
  - Ogni VM è allocata ad un utente che può usare un qualsiasi SO
    - SO ospiti (guest) sono eseguiti su ciascuna VM
- Il SO VM è eseguito sulla macchina reale (macchina host)
  - Schedula i vari SO guest
- La distinzione tra le modalità privilegiata e utente della CPU causa alcune difficoltà nell'uso di un SO VM

Esempio: struttura del SO VM - VM/370



SO con Macchina Virtuale (cont.)

- Virtualizzazione: mapping delle interfacce e delle risorse di una VM nelle interfacce e le risorse di una macchina host
  - La virtualizzazione completa può indebolire la sicurezza
  - La para-virtualizzazione sostituisce una istruzione non virtualizzabile con istruzioni virtualizzate in modo semplice
    - Il codice di un SO guest è modificato per evitare l'uso di istruzioni non virtualizzabili
      - Porting del SO guest per funzionare sotto il SO VM
      - Usando una traduzione binaria dinamica del kernel di un SO guest

SO con Macchina Virtuale (cont.)

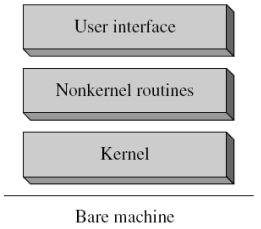
- Le VM sono impiegate per diversi scopi
  - Consolidamento del carico di lavoro
  - Fornire sicurezza e attendibilità alle applicazioni che usano lo stesss host e lo stesso SO
  - Testare un SO modificato su un server concorrentemente con esecuzioni in produzione di quel SO
  - Fornire capacità di gestione disastri
    - Una VM è trasferita da un server che deve arrestarsi ad un altro server disponibile in rete

SO con Macchina Virtuale (cont.)

- Le VM sono usate anche senza un SO VM
  - Virtual Machine Monitor (VMM)
    - Chiamato anche *hypervisor*
    - Ad esempio, VMware e XEN
- VM per linguaggi di programmazione
  - Pascal negli anni '70
    - Sostanziale perdità di prestazioni
  - Java
    - Java Virtual Machine (JVM) per sicurezza e affidabilità
    - La perdita di prestazioni può essere compensata implementando la JVM in HW

SO basati su kernel

- Motivazioni storiche, per una struttura basata su kernel, furono la portabilità e convenienza nella progettazione del SO e nella codifica delle routine non kernel
  - Meccanismi implementati nel kernel, le politiche all'esterno
- I SO basati su kernel hanno una limitata estendibilità



SO basati su kernel (cont.)

- Funzioni e servizi tipici offerti dal kernel

OS functionality	Examples of kernel functions and services
Process management	Save context of the interrupted program, dispatch a process, manipulate scheduling lists
Process communication	Send and receive interprocess messages
Memory management	Set memory protection information, swap-in/ swap-out, handle page fault (that is, "missing from memory" interrupt of Section 1.4)
I/O management	Initiate I/O, process I/O completion interrupt, recover from I/O errors
File management	Open a file, read/write data
Security and protection	Add authentication information for a new user, maintain information for file protection
Network management	Send/receive data through a message

Evoluzione struttura basata su kernel dei SO

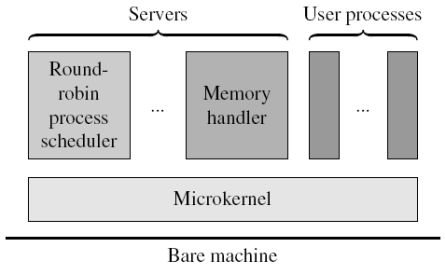
- Moduli kernel caricabili dinamicamente
  - Kernel progettato come un insieme di moduli
    - I moduli interagiscono attraverso interfacce
  - Il kernel di base è caricato durante il boot
    - Gli altri moduli sono caricati quando necessario
      - Risparmia l'uso di memoria
  - Usato per implementare i driver di dispositivo e nuove chiamate di sistema
- Driver di dispositivo di livello utente
  - Facilità di sviluppo, debugging, distribuzione e robustezza
  - Le prestazioni sono assicurate attraverso mezzi HW e SW

SO basati su micro-kernel

- I micro-kernel furono sviluppati nei primi anni ‘90 per ovviare ai problemi di portabilità, estendibilità e affidabilità dei kernel
- Un micor-kernel è un nucleo essenziale del codice del SO
  - Contiene solo un sottoinsieme dei meccanismi inclusi tipicamente nel kernel
  - Supporta solo un piccolo numero di chiamate di sistema, usate e testate massicciamente
  - Al di fuori del kernel c’è meno codice essenziale

SO basati su micro-kernel (cont.)

- Il micor-kernel non include lo scheduler e il gestore della memoria
  - Sono eseguiti come server



SO basati su micro-kernel (cont.)

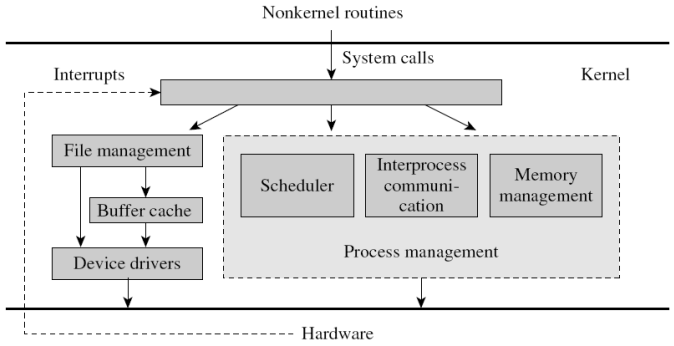
- C’è una variabilità considerevole nei servizi inclusi in un micro-kernel
- I SO con micro-kernel di prima generazione soffrivano fino al 50% di degrado nel throughput
  - I micro-kernel L4 rappresentano la seconda generazione
    - IPC più efficiente eliminando il controllo di validità/diritti come default
    - Solo 5% di degrado
  - L’exokernel fornisce solamente un multiplexing efficiente delle risorse HW
    - Gestione distribuita delle risorse
    - Estremamente veloce

Casi di Studio

- Architettura di Unix
- Il kernel di Linux
- Il kernel di Solaris
- Architettura di Windows

Architettura di Unix

- Lo Unix originale era monolitico
- I moduli kernel furono aggiunti successivamente



Kernel del SO Unix

Il Kernel di Linux

- Fornisce le funzionalità di Unix System V e BSD
- Aderisce allo standard POSIX
- Kernel monolitico
- Moduli caricabili individualmente
  - Pochi moduli kernel caricati al boot
- Miglioramenti nel kernel Linux 2.6
  - Il kernel è prelazionabile
    - Più responsivo ai programmi utente e alle applicazioni
  - Supporta architetture che non possiedono una MMU
  - Migliore scalabilità attraverso un modello migliorato dei thread

Il Kernel di Solaris

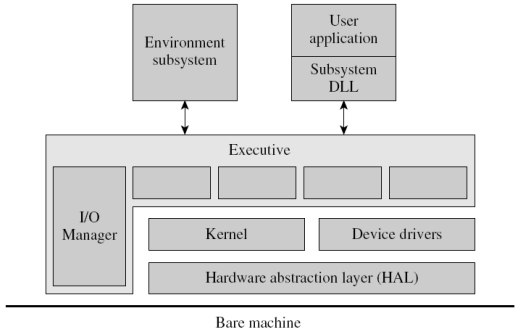
- Il SO Sun era basato su Unix BSD
- Solaris è basato si Unix SVR4
- Dagli anni ‘80 Sun si è orientata sul networking e l’elaborazione distribuita
  - Le caratteristiche sono diventate standard
    - RPC
    - NFS
  - Più tardi, Sun si è rivolta anche ai sistemi multi-processore
    - Rendendo il kernel multi-thread e rendendolo prelazionabile
    - Tecniche di sincronizzazione veloci nel kernel

Il Kernel di Solaris (cont.)

- Solaris 7 impiega la tecnologia di progettazione del kernel che carica dinamicamente i moduli kernel
  - Supporta sette tipi di moduli caricabili
    - Classi di scheduler
    - File system
    - Chiamate di sistema caricabili
    - Loader per differenti formati di file eseguibili
    - Moduli di stream
    - Controllori di bus e driver di dispositivo
    - Moduli vari
  - Facilmente estendibile

Architettura di Windows

- Interfaccia HAL con la macchina HW
- I sottosistemi d'ambiente supportano l'esecuzione dei programmi per MS DOS, Win 32 e OS/2



Riepilogo

- Portabilità: la facilità con cui il SO può essere implementato su computer con differenti architetture
- Estendibilità: facilità con cui le sue funzionalità possono essere modificate o migliorate per adattarlo a nuovi ambienti di elaborazione
- Una funzionalità del SO tipicamente contiene una politica e pochi meccanismi per implementarla
- I primi SO avevano una struttura monolitica

Riepilogo (cont.)

- La progettazione a strati usa il principio dell'astrazione per controllare la complessità nella progettazione del SO
- Il SO con macchina virtuale (SO VM) supporta simultaneamente il funzionamento di diversi SO su un computer
  - Crea una macchina virtuale per ogni utente
- Nela progettazione basata su kernel, il kernel è il nucleo del SO, che invoca routine non kernel per implementare operazioni su processi e risorse
- Un micro-kernel è il nucleo essenziale del codice del SO
  - I moduli per la politica sono implementati come processi server