Sistemi Operativi per Sistemi di Elaborazione Ubiqui

Sistemi Operativi per Ubiquitous Computing

- Palm OS
- Symbian OS
- Windows Mobile
- Embedded Linux
- QNX Neutrino
- BeOS
- TinyOS
- Java Card

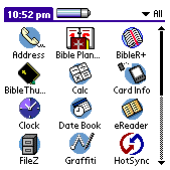
Sistemi Operativi per Ubiquitous Computing

- I sistemi operativi per sistemi di elaborazione ubiqui seguono i principi dei SO classici ma
  - devono gestire risorse con caratteristiche particolari,
  - hanno interfacce del tutto diverse
- In questo settore non esiste un SO prevalente. I più diffusi sono Symbian, Palm OS e Windows Mobile insieme a TinyOs e Embedded Linux.

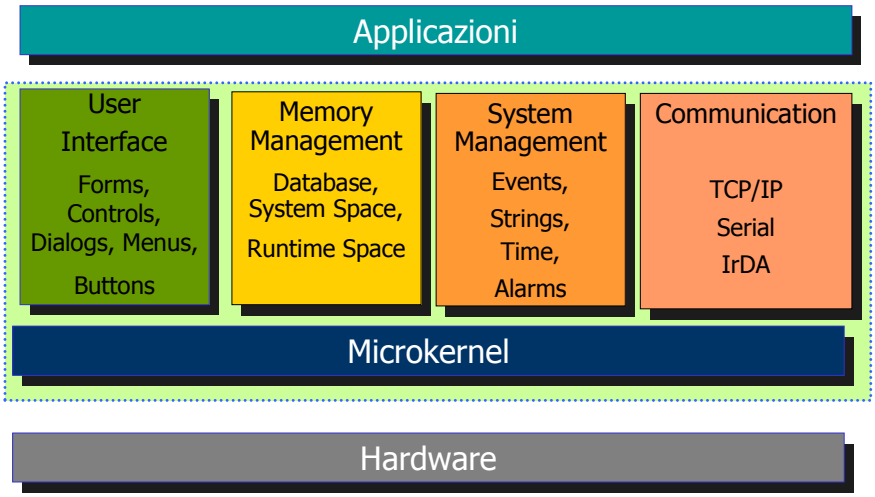
Sistemi Operativi : Palm OS

- Sistema operativo per cellulari e palmari (PDA).
- Sistema operativo a 32 bit

- Versioni correnti : (Palm OS Garnet 5.x, Palm OS Cobalt, version 6.1) con supporto Bluetooth e 65K colori per PDA/cellulari multimediali.



Sistemi Operativi : Palm OS



Sistemi Operativi : Palm OS

- **User Interface**: gestione dell'I/O grafico, menu, buttons, forms.
- **Memory Management**: DB, runtime space, system space, variabili globali.
- **System Management**: eventi, alarms, time, strings, …
- **Communication Layer**: I/O seriale, TCP/IP, Infrared Data Association (IrDA).

Sistemi Operativi : Palm OS

- **User management**: SO single user.
- **Dimensione** : v3.5 richiede circa 1.4 MBytes.
  La versione Cobalt:
  - Per cellulari richiede 32MB SDRAM + 32MB Flash
  - Per PDA 32MB SDRAM + 16MB ROM
- **Task Management**: multitasking e multithreading.
- **Power Management**: tre stati (sleep, doze, running)

Sistemi Operativi : Palm OS

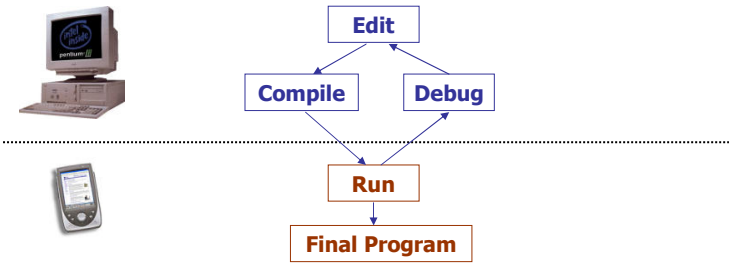
- **Memory Management**: le applicazioni non sono separate (una applicazione può causare il crash del sistema). Il file system tradizionale è sostituito da un insieme di database gestiti da un *Database Manager*
- La memoria è separata in
  - *Dynamic heap* : dimensione tra 64Kb e 256Kb e serve a contenere le variabili globali, lo stack, e la memoria allocata dinamicamente durante l'uso.
  - *Storage* : contiene dati permanenti (DB, files) che non vanno cancellati allo spegnimento.

Sistemi Operativi: Palm OS

- Palm OS è un sistema multi-task *event driven*.
- Gli eventi possono essere:
  - Una azione della user interface (es., touch screen op.)
  - Un system notification (es., un alarm del timer)
  - Un evento di una applicazione (es., richiesta di search)

Sistemi Operativi: Palm OS – Sviluppo SW

- Sviluppo del Software in C e C++ con CDK e SDK esterno.

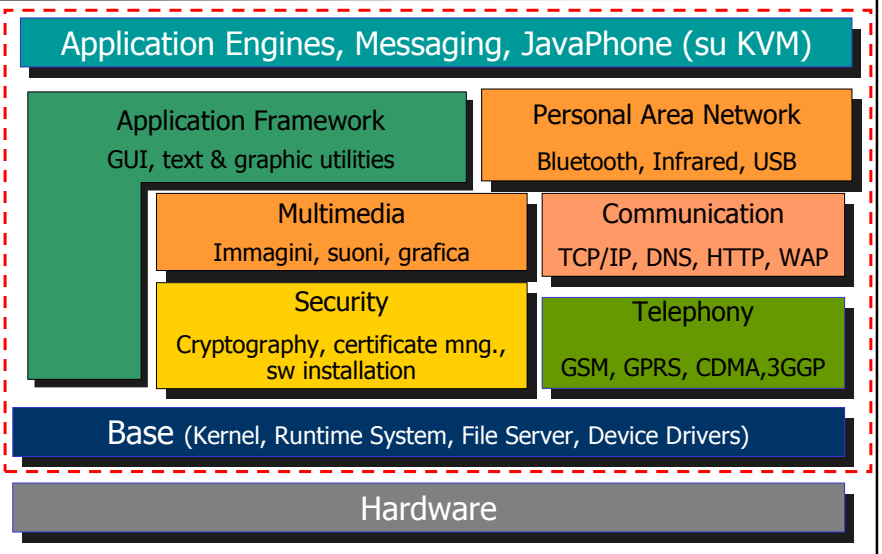


- Esiste un Palm Emulator per sviluppare e fare il test delle applicazioni prima di eseguirle su Palm OS
- Macchine virtuali: KVM, J9, WabaVM (poco efficienti)

Sistemi Operativi: Symbian OS

- Creato con il nome di EPOC da Psion come SO per telefonia mobile.
- Attualmente sviluppato da Symbian.
- Usato in cellulari NOKIA e Sony Ericsson e su diversi processori (anche in emulazione).
- Caratteristiche: multi-tasking real-time pre-emptive, 32 bit.

Sistemi Operativi : Symbian OS

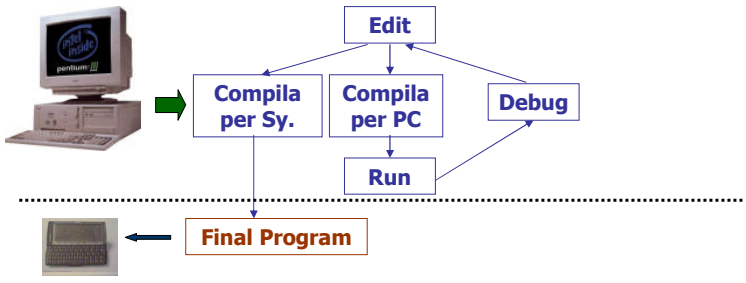


Sistemi Operativi : Symbian OS

- **User management**: SO single user.
- **Task Management**: microkernel real-time, multitasking con scheduling pre-emptive e con priorità. (Gestione complessa di applicazioni)
- **User Management**: con interfaccia standard: grafica, suoni e tastiera.
- **Memory Management**: MMU con spazi di indirizzi separati per applicazioni.

Sistemi Operativi: Symbian OS

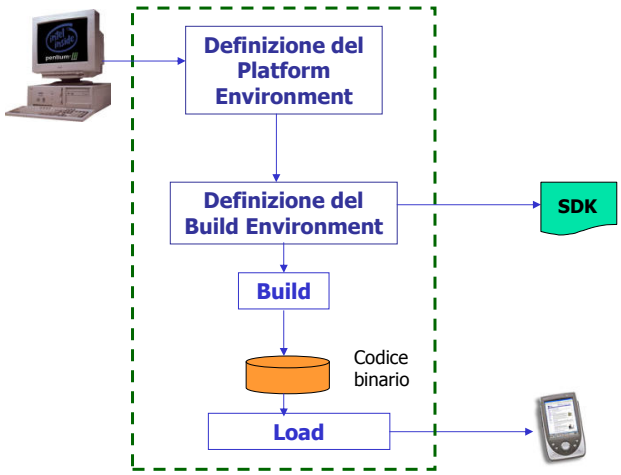
- Sviluppo del Software in C++, Java e OPL (Basic-like).
- Esiste un Simulatore per sviluppare e fare il test delle applicazioni prima di eseguirle su Symbian OS



Sistemi Operativi: Windows Mobile

- Windows Mobile è una versione di Windows sviluppata per sistemi mobili e pervasivi.
- Il sistema va configurato per la specifica piattaforma (PDA, cellulare, altro) su cui deve essere usato. Basato su memoria ROM.
- Ha l'interfaccia tipica di Windows adattata per i display dei sistemi mobili.

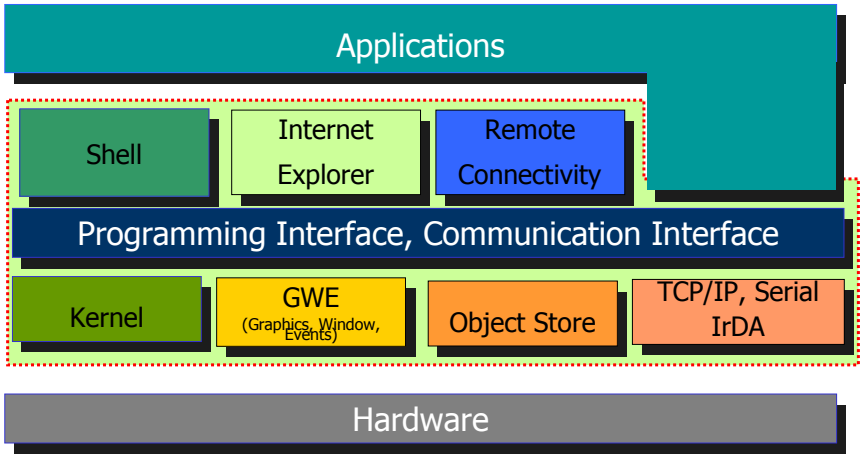
Sistemi Operativi: Windows Mobile – Configurazione



Sistemi Operativi: Windows Mobile

- **User management**: SO single user.
- **Task Management**: fino a 32 processi e un numero elevato di thread (limitato dalla memoria disponibile)
- **User Interface**: icone, dialog boxes, menu, suoni (approccio alla Windows)
- **Memory Management**: memoria protetta con 32 MB per processo, heap per file system, registry e object store (fino a 256 MB).

Sistemi Operativi : Windows Mobile



Sistemi Operativi: Windows Mobile

- **Dimensione** : La versione CE occupava da 400 Kb (kernel) a 3 MB (sistema completo) fino a 8 MB con Pocket Word e Internet Explorer.
- **Security**: Crittografia con una libreria per gestire i dati memorizzati in sicurezza. Memorizzazione sicura con smart card.
- **Ambienti Software** : Visual C++, Visual Basic, KVM, J9 e Waba.

Sistemi Operativi: Embedded Linux

- Versione di Linux per sistemi di elaborazione ubiqui.
- Architettura a microkernel. Funzioni e servizi compilabili nel kernel o generabili come moduli separati da caricare dinamicamente.
- Ampia gamma di protocolli e servizi per il networking.
- Configurabile e scalabile da un orologio ad un multiprocessore.

Sistemi Operativi: Embedded Linux

- **User management**: SO multi user.
- **Task Management**: Multitasking con scheduler pre-emptive e real-time (opzionale). Supporto per multiprocessori.
- **User Interface**: basata su X-Window
- **Memory Management**: gestione alla Linux con MMU e memoria virtuale.
- **Dimensione** : da 200 Kb (kernel) a circa 10 MB.

Sistemi Operativi: Embedded Linux

- La specifica è basata su:
  - Linux Standards Base 1.2.
  - IEEE POSIX 1003.1-2001 specification, contenente funzionalità Realtime, Threads and Networking.
  - Single UNIX Specification v3.

Sistemi Operativi: Embedded Linux

- L'ambiente di sviluppo per Linux è disponibile in Embedded Linux.
- Linguaggi: C, C++, Java
- Driver, utility, protocolli e programmi client e server disponibili per connessioni Internet.
- Similitudini con QNX Neutrino: sistema operativo POSIX compliant.

Sistemi Operativi: Embedded Linux on a watch

- Embedded Linux in un orologio
- Comunicazione wireless con altri dispositivi, PIM (calendario, address book, event list, …).
- Connessione ad Internet.
- Processore ARM7, 8MB memoria flash, 8MB memoria DRAM, infrarossi, RF wireless, 96x112 touch screen, roller wheel.

Sistemi Operativi : BeOS

- BeOS è un sistema operativo per multimedia boxes.
- Supporto per multi-processori e file system a 64 bit.
- Disponibile su processori Intel e PowerPC.
- Progettato per real-time multimedia e communication.
- Gestisce multithreading su sistemi multiprocessore.

Sistemi Operativi per Sensori

- Un sistema operativo per sensori deve possedere alcune caratteristiche base:
  - dimensioni ridotte,
  - basso consumo durante l'elaborazione,
  - consumo quasi nullo durante lo stato di *idle*,
  - gestione della concorrenza,
  - implementazione di protocolli di rete a seconda della periferica di rete utilizzata
  - fornire un astrazione per i dispositivi hardware (sensori ed attuatori), montati sul nodo sensore.
- Ad esempio un protocollo di trasporto come il TCP/IP non può essere impiegato in quanto offre un servizio orientato alla connessione , e siccome un nodo sensore possiede limitate capacità , anche di memoria , un protocollo di questo tipo è attualmente inapplicabile.

Sistemi Operativi per Sensori

Esistono 2 approcci allo sviluppo di sistemi operativi per sensori:

- Sviluppare un sistema i cui componenti vengono compilati insieme all'applicazione.
  - Questo consente che ci sia una singola applicazione in esecuzione in un dato momento con riduzione dei consumi energetici per *l'assenza di context-switch* e sistemi molto piccoli (nel sistema sono caricati solo i moduli che effettivamente verranno utilizzati).
  - Lo svantaggio principale riguarda la versatilità ed i vincoli di riconfigurabilità dell'applicazione.
- Sviluppare un sistema che includa i tradizionali strati di software di un sistema operativo tradizionale in versione ridotta.
  - In questo caso è difficile tenere sotto controllo i consumi e le risorse impiegate, ma si guadagna in termini di versatilità visto che ci possono essere, più applicazioni in "esecuzione".

Sistemi Operativi per Sensori: TinyOS

- TinyOS consiste in un insieme ridotto di moduli (servizi) software che forniscono funzionalità per il controllo di un nodo sensore.
- Quando un'applicazione viene compilata, i componenti di TinyOS vengono compilati insieme ad essa ed il risultato costituisce l'intero software del sensore, consentendo così un risparmio di energia e di memoria.
- Non è possibile installare più applicazioni indipendenti sullo stesso sensore.

Sistemi Operativi per Sensori: TinyOS

- In TinyOS non esiste un kernel che gestisce le risorse disponibili dividendoli tra più applicazioni, ma solo uno scheduler che si limita ad eseguire una sequenza di task secondo una politica FIFO preemptive basata su eventi.
- L'esecuzione del task corrente può essere interrotta solo se si verifica un evento (preemption). Un task non può interrompere un altro task .
- Un task è un contesto di esecuzione che viene eseguito in background fino al completamento e senza interferire con gli eventi del sistema.

Sistemi Operativi per Sensori : TinyOS

- Quando non ci sono task da eseguire lo scheduler mantiene il processore a riposo e attende la segnalazione di un nuovo evento per riprendere l'esecuzione.
- Come si intuisce lo scheduling ha due livelli di priorità, quello normale per i task e quello più alto per gli eventi che possono interrompere i task.
- TinyOS è scritto in nesC, una variante del linguaggio C, concepito per sistemi embedded e ottimizzato per la programmazione dei nodi sensori.

Sistemi Operativi per Sensori : TinyOS

- In TinyOS, il modello di comunicazione rappresenta una delle parti fondamentali.
- Lo scambio di dati tra i nodi utilizza gli active messages (messaggi attivi).
- Questo modello permette di inviare messaggi, a tutti o a un singolo nodo, tra i nodi vicini.
- Questa tecnologia è molto leggera, infatti non specifica dei meccanismi orientati alla connessione, ma ogni pacchetto e un'entità indipendente.
- Ogni *active message* contiene l'identificatore dell' *handler* (generalmente un intero che rappresenta il numero della porta) dell'application level che deve essere invocato sul nodo destinatario ed un payload dati nel quale sono specificati gli argomenti.

Sistemi Operativi per Ubiquitous Computing

- Palm OS è semplice, compatto, ma non implementa meccanismi di security.
- Symbian OS è più complesso, ma più generale e gestisce un efficiente multitasking.
- Windows Mobile supporta configurazioni flessibili e usa crittografia per la security.
- Embedded Linux offre l'interfaccia di programmazione avanzata di Linux su sistemi ubiqui.
- TinyOS è il sistema operativo multitasking per sensori più diffuso.