Problematiche particolari di ingresso / uscita

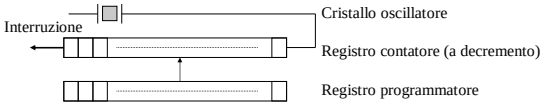
Indice

Orologi - 1

- Hanno 2 funzioni fondamentali
  - Misurano lo scorrere del tempo di sistema
  - Misurano la durata dell'utilizzo di risorse critiche da parte di processi
- Usano un *hardware* speciale, visto come dispositivo di ingresso
- Richiedono un *software* di gestione
  - Tipicamente posto sotto il controllo del S/O

Orologi - 2

- Visione hardware
  - Modello rudimentale
    - Dispositivo collegato alla linea di alimentazione
    - Una interruzione ad ogni ciclo di voltaggio (50-60 Hz)
  - Modello avanzato, programmabile



Orologi - 3

- Il segnale periodico emesso dal cristallo viene moltiplicato per generare la frequenza desiderata
  - Distribuito all'intero sistema per sincronizzazione
- Ogni emissione amplificata di segnale causa un decremento nel registro contatore il cui valore iniziale è dato dal registro programmatore
- Quando il registro contatore vale 0 si produce una interruzione verso la CPU
  - Modalità non ripetitiva (*one-shot*)
    - Inizializzazione → decremento → arresto → riprogrammazione
  - Modalità ciclica (*square-wave*)
    - Inizializzazione → decremento → ricaricamento automatico
    - Interruzione periodica detta *clock tick*

Orologi - 4

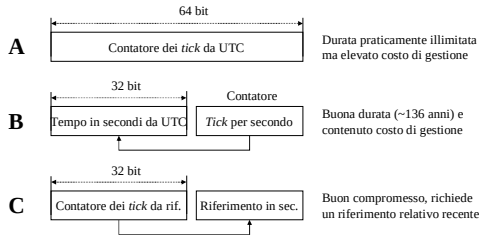
- Frequenza di interruzione controllata a software
  - Cristallo ad 1 GHz → 1 decremento ogni 1 *ns*
  - Registro contatore a 32 bit → interruzioni con periodo programmabile tra 1 *ns* e 4,3 *s*
- Un singolo dispositivo può contenere + orologi programmabili con svariate opzioni
  - Contatore ad incremento; interruzioni disabilitate
- Un orologio di riserva (a basso consumo) alimentato a batterie mantiene il tempo attuale a computer spento
- Il valore del tempo corrente si misura come trascorso dall'*Universal Coordinated Time* (**UTC**)
  - UNIX: 01/01/70 Windows: 01/01/80

Orologi - 5

- Un gestore *software* specifico (*clock driver*) associa azioni significative alle interruzioni generate dall'orologio *hardware*
  - Avanzamento del tempo corrente (*real time*)
    - Gestione dell'*overflow* di registro
  - Controllo del tempo di esecuzione dei processi
    - Utile per ordinamento a quanti e prevenzione di eccessi
  - Verifica della situazione del sistema
    - Statistiche di gestione, monitoraggi periodici
  - Gestione di meccanismi di allarme (*watchdog*)

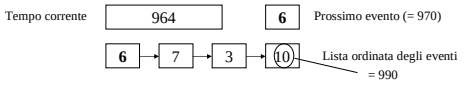
Orologi - 6

- Gestione dell'avanzamento del tempo corrente



Orologi - 7

- Controllo del tempo di esecuzione
  - Decremento del quanto
  - Un orologio (virtuale) privato per ogni processo
    - Avanza per ogni *tick* che il processo è in esecuzione (accuratezza a costo non trascurabile)
- Gestione di meccanismi di allarme
  - Lista ordinata inversamente alla distanza delle richieste



Schermo e tastiera

- Schermo e tastiera collettivamente denominati *terminali*
- 3 classi di terminali per 3 distinte categorie di utenti
  - Ad interfaccia seriale (RS-232)
  - Ad interfaccia grafico (tipo *personal computer*)
  - Ad interfaccia di rete

Schermo e tastiera
Terminale ad interfaccia seriale - 1

- Funzionamento per carattere (*character-oriented*)
  - Uscita sullo schermo per linee di testo (25×80) via **CRT**
- Collegato da linea seriale di tipo RS-232
  - 1 bit alla volta su 1 dei 9 o 25 pin del cavo
    - Bit di controllo delimitano i caratteri inviati → bassa velocità
    - Conversione da flusso di bit a caratteri effettuata da **UART** (*Universal Asynchronous Receiver Transmitter*)
    - Una interruzione segnala la disponibilità a ricevere / inviare (anche insieme) un nuovo carattere
  - `/dev/tty`*n* (1,2,…) per UNIX/Linux (`Teletype`®)
  - `COM`*n* (1,2) per Windows
- Modalità semplice (*dumb*) o intelligente
  - Intelligenza è interpretare "caratteri o sequenze di *ESCape*"

Schermo e tastiera
Terminale ad interfaccia seriale - 2

- 2 strategie di funzionamento per il gestore di tastiera (*input*)
  - Per carattere (*raw mode*), non canonico$_{\text{POSIX}}$
    - Utile per associare azioni a sequenze di tasti (***emacs***)
  - Per linea (*cooked mode*), canonico$_{\text{POSIX}}$
    - Nasconde il dettaglio della trasformazione
    - Richiede memorizzazione fino a linea completa
      - Mediante catena di piccoli buffer (~10 char) rilasciati e riusati a linea inviata
      - Strutture + capaci (~200 char) direttamente nella memoria del terminale intelligente
- Immissione slegata da accettazione (*type ahead*)

Schermo e tastiera
Terminale ad interfaccia seriale - 3

- Eco sullo schermo di ogni carattere ricevuto
  - Meno ovvio di quanto sembri
- Trattamento di riposizionamento (*carriage return*) ed avanzamento di linea (*linefeed*)
  - Per UNIX/Linux LF vale {CR + LF}, ma Windows richiede entrambi (`unix2dos`)
- Caratteri speciali standard di tipo CTRL-x (POSIX)
  - Conflitti interpretativi evitati mediante "carattere di *escape*": carattere seguente interpretato letteralmente
- Controllo dell'uscita video mediante "sequenze di ESCape"
  - Interpretate via `termcap` o standard ANSI

Schermo e tastiera
Terminale ad interfaccia grafico - 1

- Noto come GUI (*Graphical User Interface*)
  - Introdotto dal modello Lisa di Apple nel gennaio 1983 (`http://www.apple-history.com/lisa.html`)
  - Basato sul paradigma WIMP
    - Finestre (*windows*), icone (*icons*), menu e dispositivi di puntamento (*pointing*)
- Implementabile sia come programma utente (UNIX/Linux) che come parte del S/O (Windows)

Schermo e tastiera
Terminale ad interfaccia grafico - 2

- Video rappresentato in memoria per bit
- Tastiera collegata mediante porta seriale / parallela o USB
  - Pentium: tastiera dotata di microprocessore collegato con CPU tramite porta seriale speciale
    - Interruzione generata ad ogni (de)pressione di tasto, associata ad uno specifico codice con corrispondenza ASCII
    - Il gestore di tastiera determina il significato inteso
      - L'emissione di "A" richiede 4 codici
- Dispositivo di puntamento invia messaggi di tipo (`Δx, Δy, pulsante`) periodicamente e/o per evento

Spostamento relativo al messaggio precedente

Vedi §4.6-7 di AE-1 per funzionamento video

Schermo e tastiera
Terminale ad interfaccia grafico - 3

- Finestra = area rettangolare localizzata dalle coordinate dei due vertici diagonali
- Programma di controllo Windows® esegue azioni veicolate da messaggi accodati
  - Corrispondenza azione - messaggio
- Ogni finestra è istanza di una classe specifica
- L'aspetto grafico è gestito da una libreria chiamata GDI (*Graphics Device Interface*) operante in modalità "vettore" (*vector graphics*)
  - Un *metafile* (`.wmf`) descrive un diagramma complesso come sequenza di chiamate GDI
  - L'altra modalità è detta "a linee orizzontali" (*raster* o *bitmap graphics*), trattata dalla procedure `BitBlt`

Esempio 1
Struttura di programma principale Windows per gestione finestre

Parametri in ingresso:
- Puntatore (*handle*) oggetto finestra;
- Stringa comando di creazione;
- Caratteristiche della finestra iniziale;
- Altro per compatibilità

Causa di complessità!

Azioni:

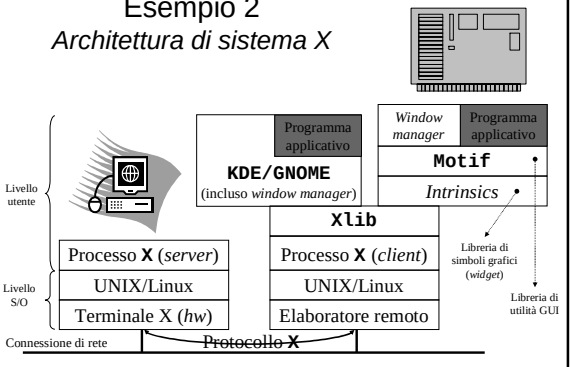
Inizializzazione:
1. Associa attributi ad oggetto finestra (p.es. : nome, icona);
2. Installa puntatore a procedura **`WndProc`** che ne gestisce i messaggi;
3. Registra oggetto nell'anagrafe di Windows;
4. Assegna memoria per l'oggetto;
5. Rendi l'oggetto graficamente sul video;

Ciclo operativo:
1. Preleva messaggio da coda messaggi;
2. Interpreta messaggio;
3. Invia messaggio a **`WndProc`**;

Schermo e tastiera
Terminale ad interfaccia di rete - 1

- Terminale connesso ad elaboratore remote tramite collegamento di rete
  - Terminale potente → limitato carico di rete
    - Scelta UNIX/Linux **X Window System** (`http://www.x.org`)
    - Modello detto *fat client* (`/usr/X11R6/bin/X`)
  - Rete molto capace → terminale minimo (*dumb*)
    - Rivisitazione migliorata (ora detta *thin client*) di modello architetturale vecchio
    - Terminale utente solo esecutore senza stato
      - ***SLIM***: *Stateless Low-level Interface Machine*

Esempio 2
Architettura di sistema X



Schermo e tastiera
Terminale ad interfaccia di rete - 2

- **X** ha struttura modulare, flessibile e portabile
  - Architettura progettata per evolvere
  - La struttura Windows è invece *conseguenza* dell'esigenza commerciale di evolvere il prodotto
- Basata su messaggi e risorse
  - L'applicazione scambia messaggi con il terminale
    - Programma → terminale : comandi grafici, interrogazioni
    - Terminale → programma : eventi, risposte ad interrogazioni
  - L'applicazione crea risorse (oggetti con attributi ma senza classe) che rappresentano entità grafiche associate a specifiche finestre
- Permette anche di emulare il semplice interfaccia seriale
  - Il programma cliente chiamato **`xterm`** si comporta come un terminale intelligente ad interfaccia seriale
    - Ciò consente il riutilizzo di programmi di grande diffusione ed utilità come **`emacs`**

Esempio 3
Struttura di programma applicativo su X

Dichiarazioni:
- Identificatore di *server*;
- Identificatore di finestra;
- Portafoglio di attributi grafici di finestra (*graphic context*);
- Contenitore di evento;

Azioni:
Inizializzazione:
1. Connetti a *server*; // acquista il diritto di usare il *display* sul terminale
2. Assegna memoria a finestra;
3. Registra finestra all'anagrafe del gestore finestre (*window manager*);
4. Assegna valori al portafoglio attributi;
5. Invia messaggio richiesta di esporre finestra sul video;

Ciclo operativo:
1. Ricevi evento ponendolo in contenitore; // dal terminale
2. Tratta evento; // anche inviando comandi al *server*

Fine:
1. Distruggi portafoglio;
2. Rilascia memoria;
3. Chiudi connessione con *server*;

Schermo e tastiera
Terminale ad interfaccia di rete - 3

- L'architettura **SLIM** vuole terminali minimi
  - Terminale = semplice schermo gestito come *bitmap* 1280×1024, tastiera e mouse, nessuna installazione di programmi
    - *Thin client*
  - Tutto l'onere è posto su un elaboratore centrale e su una rete di connessione molto potenti
    - 5 messaggi di base sono sufficienti per controllare in remoto la grafica sul terminale
    - Le prestazioni percepite dall'utente dipendono criticamente dalla capacità della rete
      - Rete a 100 Mbps
        - Eco di carattere sullo schermo 60 volte più rapido che con modello tradizionale
        - Aggiornamento di schermo sotto applicazioni grafiche avanzate (p.es. Netscape) 20 volte più rapido del necessario
      - Reti ad 1-10 Mbps danno prestazioni ancora buone, al di sotto scadenti