# 6 – Caso di Studio Linux

**Sommario**





0

# Obbiettivi

- Architettura del nucleo di Linux
- Implementazione delle componenti del S.O. Linux: processi, memoria e gestione dei file
- Livelli software che compongono il nucleo di Linux
- Come Linux organizza e gestisce i dispositivi del sistema
- Come Linux gestisce operazioni di I/O
- Cenni ai meccanismi di comunicazione e sincronizzazione tra processi in Linux
- Cenni alle funzioni di sicurezza di Linux



1

# Introduzione

- Linux kernel
  - Nucleo del S.O. **open-source** più diffuso, distribuito gratuitamente e completo
  - codice sorgente di Linux è disponibile a tutti per studio, installazione e possibile modifica
- Diffuso anche per server di fascia alta, sistemi *desktop* e sistemi dedicati *(embedded)*
- Supporta anche molte caratteristiche avanzate
  - SMP (Symmetric Multiprocessing)
  - accesso alla memoria non uniforme (NUMA)
  - accesso ai file di diversi sistemi hardware



2

# Storia di Unix e Linux

- CTSS poi Multics (MULTIplexed information and Computing Service) sviluppato all'M.I.T. poi da Bell Labs e GE, timesharing e multiprogrammazione
- Unics (UNIplexed information and Computing Service) versione di Ken Thompson su minicomputer PDP poi UNIX
- Compilatore C portabile
- Porting
- BSD Berkeley sofware distribution
  networking, editor (*vi*), shell (*csh*), compilatori
- Due versioni maggiormente usate 4.3BSD e System V
- POSIX *portable operating system* standard per procedure di libreria



3

## Storia di Unix e Linux

- Creato nel 1991 come evoluzione di Unix, da uno studente dell'Università di Helsinki, Finlandia, Linus Torvalds dal quale deriva il nome Linux combinazione di Linus e UNIX
- Utilizza come punto di partenza il codice sorgente libero del S.O. Minix (sviluppato da A. Tanenbaum, docente dell'Università di Amsterdam, 1987) e tende a migliorarlo
- Si basa sul contributo di opinioni e supporto da parte della comunità
- Gli sviluppatori hanno poi continuato a sostenere il concetto di un nuovo S.O. disponibile liberamente



4

## Storia

- Successivamente Linux è stato esteso con possibilità di interoperatività
- Obbiettivo: conformità allo standard delle interfacce delle applicazioni e servizi del S.O., POSIX (*Portable Operating System Interface*) per Unix
- Dal 1994 include caratteristiche più avanzate
  - Multiprogrammazione
  - Memoria virtuale
  - Supporto TCP/IP
  - Caricamento a richiesta
- Obbiettivi comuni a Unix
  - Multiprogrammazione e multiutente
  - Facilitare la condivisione e cooperazione
- Per altre funzioni (es. gestione account, GUI, editor):
  - distribuzioni Linux



5

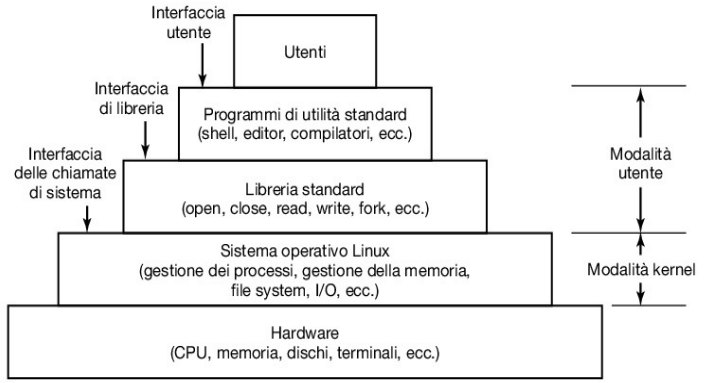
## Storia

- Distribuzioni Linux
  - Consente agli utenti non hanno familiarità con i dettagli Linux di installarlo e usarlo
  - Include il software come
    - Il nucleo di Linux
    - applicazioni di sistema
      - Es.: gestione degli account utente, la gestione della rete e strumenti di sicurezza
    - applicazioni utente
      - Es: GUI, browser web, editor di testo, applicazioni e-mail, database e giochi
    - Strumenti per semplificare il processo di installazione
  - Molte distribuzioni modificano il nucleo per aggiungere altri driver o caratteristiche specifiche



6

## Storia Unix – Linux - Livelli




A. Tanenbaum – Modern Operating Systems



7

## Storia Unix – programmi utilities

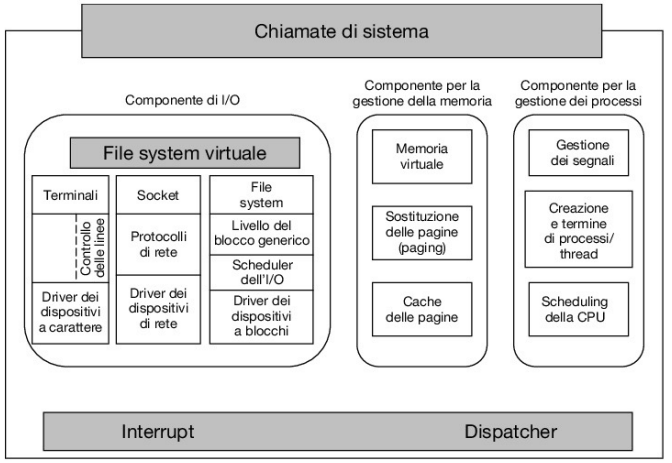
| Programma | Uso comune |
|---|---|
| cat | Concatena molteplici file sullo standard output |
| chmode | Modifica la modalità di protezione del file |
| cp | Copia uno o più file |
| cut | Taglia colonne di testo da un file |
| grep | Esamina un file per trovare un pattern |
| head | Estrae le prime righe di un file |
| ls | Elenca il contenuto della directory |
| make | Compila i file per costruire un binario |
| mkdir | Crea una directory |
| od | Trasforma un file in ottale |
| paste | Incolla colonne di testo in un file |
| pr | Formatta un file per la stampa |
| ps | Elenca i processi in esecuzione |
| rm | Rimuove uno o più file |
| rmdir | Rimuove una directory |
| sort | Ordina un file di righe in ordine alfabetico |
| tail | Estrae le ultime righe di un file |
| tr | Trasforma tra insiemi di caratteri |

Alcuni dei programmi utility UNIX richiesti da POSIX



8

## Linux – struttura del nucleo




Struttura indicativa del nucleo di sistema LINUX



9

## Storia

- Numeri di versione
  - Incrementato a discrezione da Torvalds per ogni versione del nucleo che contiene un set di funzionalità significativamente diverso da quello della precedente
  - numero di versione minore (cifra dopo il primo punto)
    - Fino alla versione 2.6 il numero pari indica una versione stabile
    - Numero dispari indica una versione minore, es. 2.6.1, indica una versione in fase di sviluppo
  - Le cifre che seguono il secondo punto decimale sono incrementate per ogni aggiornamento minore del nucleo
  - *Esempi*
    - *Linux 2.0 (1996), 2.2 (1999 con SMP), 2.4 (2001, supporto di diversi hw, migliori prestazioni e scalabilità), 2.6 (dal 2003), 3 (2011), 4 (2015)…*



10

## Overview di Linux

- I sistemi Linux includono interfacce utente e altre applicazioni oltre al nucleo
- Eredita da UNIX il modello a **livelli**
- Accesso tramite interfaccia utente
- Per le chiamate di sistema
- Il S.O. contiene *thread* del nucleo per eseguire i servizi
  - Implementati come demoni, che possono essere dormienti fino a quando non sono risvegliati da un componente del nucleo
- **Sistema multiutente**
  - Diritti di accesso
  - Sincronizzazione
  - Limita l'accesso alle operazioni importanti per gli utenti con privilegi da superutente (detto *root* o *superuser*)



11

## Sviluppo e comunità

- Torvalds controlla tutte le modifiche del nucleo
- Si appoggia su un gruppo una ventina di sviluppatori fidati per la gestione dei miglioramenti del nucleo
- Quando un nucleo di sviluppo si avvicina al completamento
  - Congelamento delle caratteristiche (***feature-freeze***): non si aggiungono nuove funzionalità al nucleo, correzioni per migliorare le prestazioni
  - Congelamento del codice (***code-freeze***): accettate solo correzioni di bug importanti al codice
- Molte aziende sostengono e migliorano lo sviluppo di Linux
- Linux è distribuito sotto la GNU Public License (GPL)
  - General Public Licence
  - GNU progetto della Free Software Foundation (1984) mira a fornire software libero e S.O. simili a Unix
- Linux è gratuito, ma software protetto da copyright





## Distribuzioni

- Oltre 300 distribuzioni disponibili
- Solitamente organizzati in *packages*, ciascuno con un solo servizio o applicazione
- Alcune fra le più diffuse distribuzioni
  (commerciali o senza scopo di lucro)
  - Debian
  - Ubuntu
  - Arch Linux
  - CentOS
  - Fedora
  - Linux Mint
  - Mandrake (c)
  - Red Hat (c)
  - SuSE (c)
  - Slackware
  - …





## Interfaccia

- Si può accedere tramite terminale emulato, tramite la riga di comando di una shell come *bash*, *csh* e *esh*
- La maggior parte delle interfacce grafiche (GUI) di Linux sono a livelli
  - X Window System (MIT 1984, Interfaccia grafica di basso livello)
    - Livello più basso
    - Fornisce ai livelli GUI più alti meccanismi per creare e manipolare componenti grafiche
  - Window manager
    - Costruito sui meccanismi nell'interfaccia X Window per controllare il posizionamento, l'aspetto, le dimensioni e altri attributi della finestra
  - Ambiente desktop (ad esempio, KDE, GNOME) – (opzionale)
    - Fornisce agli utenti interfacce per applicazioni e servizi

| | |
|---|---|
| 3 | *Desktop Environment* |
| 2 | *Window Manager* |
| 1 | *X Window System* |





## Standard

- Linux è conforme agli standard diffusi come POSIX
- Single UNIX Specification (SUS)
  - Suite di standard che definiscono le interfacce utente per la programmazione delle applicazioni e degli utenti per S.O. UNIX, le *shell* e le *utilities*
  - Versione 3 del SUS combina diversi standard (inclusi POSIX, standard ISO e le versioni precedenti di SUS)
- Linux Standard Base (LSB)
  - Progetto che mira a standardizzare Linux in modo che le applicazioni scritte per una distribuzione conforme a LSB compilino e si comportino come su qualsiasi altra distribuzione conforme

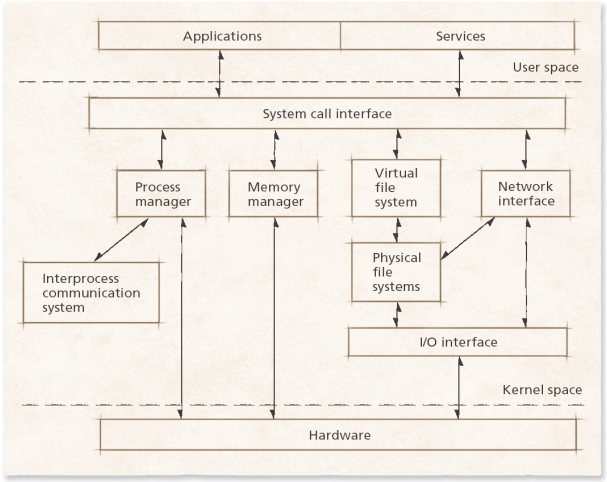
## Architettura del Nucleo

- Nucleo **monolitico ma**
  - Contiene **componenti** modulari
- S.O. UNIX-like o UNIX-based
  (es. servizi simili a Unix System V, BSD Unix)
- Sei principali sottosistemi
  1. Gestione dei **processi**
  2. Interprocess **communication**
  3. Gestione della **memoria**
  4. Gestione del **File system**
     VFS (virtual File System): fornisce una unica interfaccia a più file systems
  5. Gestione dei dispositivi di **I/O**
  6. Gestione della **rete**





## Architettura del Nucleo - Linux










## Piattaforme Hardware

- Supporta molte piattaforme fra le quali
  - *Es: x86 (incluso Intel IA-32), HP/Compaq Alpha AXP, Sun SPARC, Sun UltraSPARC, Motorola 68000, PowerPC, PowerPC64, ARM, Hitachi SuperH, IBM S/390 e zSeries, MIPS, HP PA-RISC, Intel IA-64,AMD x86-64,H8/300,V850 e CRIS*
- Codice specifico per l'architettura
  - Esegue operazioni realizzate in modo diverso dalle piattaforme con le istruzioni di basso livello
- **Porting**
  - Processo di modifica del nucleo per supportare una nuova piattaforma
  - Il codice specifico per il *porting* è separato dal nucleo e si trova in /arch
- Albero del codice sorgente (*Source tree*)
  - Organizza il nucleo in componenti separati in directory
  - Nelle directory in /arch vi sono i codici per ogni architettura
- User-Mode Linux (UML)
  - strumento importante per lo sviluppo del nucleo (verifica, messa a punto e correzione). Eseguito in modalità utente su dispositivi virtuali





## Moduli caricabili del nucleo

- Alternativa alla modifica del nucleo monolitico per estenderlo:
  uso di moduli caricabili per integrare le funzionalità del nucleo
- Modulo kernel: contiene il codice oggetto che, una volta caricato, è collegato dinamicamente al nucleo in esecuzione
- Eseguiti in modalità nucleo
  - sicurezza
- Consente il **caricamento a richiesta** del codice
  - Riduce l'occupazione di memoria del nucleo
  - Es.: file system, alcune periferiche hw
- I moduli scritti per versioni del nucleo diversi da quello corrente potrebbero non funzionare correttamente
- *Kmod*: sottosistema del nucleo che gestisce i moduli senza l'intervento dell'utente (abilitabile)
  - Determina le dipendenze dei moduli e li carica su richiesta

## Gestione dei Processi

- Gestore di processi: scheduler dei processi
  - Responsabile innanzitutto di **assegnare i processori** ai processi
  - Spedisce anche i segnali
  - Carica i moduli del nucleo
  - Riceve gli interrupt



20

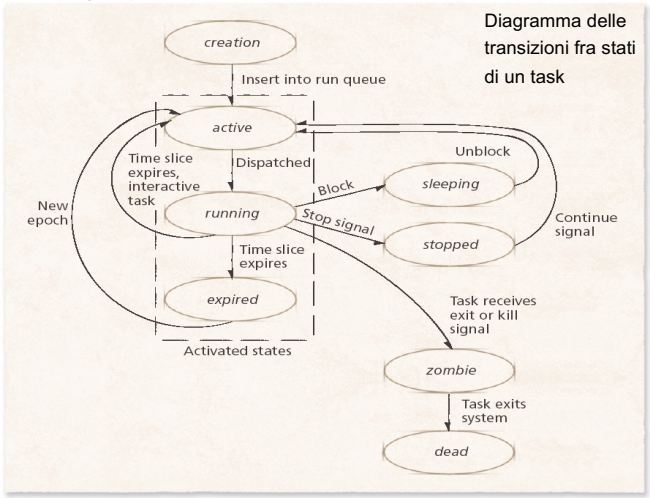
## Organizzazione dei processi e dei Thread

- Processi e *thread* sono chiamati **Tasks**
  - Processi e thread sono rappresentati internamente da una struttura (descrittore di processo) **task_struct**
- Il gestore di processi mantiene i riferimenti a tutti i task tramite
  - una **lista circolare** doppia di task
  - una **tabella hash**
- Creazione del task: assegnamento di un **PID** (*Process IDentifier*) usato per determinare con una funzione hash la posizione nella tabella dei processi
- Stati dei Task (variabile state)
  - *Running*
  - *Sleeping* (bloccato)
  - *Zombie* (terminato, ma non rimosso)
  - *Dead* (terminato e rimuovibile)
  - *Stopped* (sospeso)
  - *Attivo e terminato (expired)* (non memorizzato dalla variabile state)



21

## Organizzazione dei processi e dei Thread







22

## Organizzazione dei processi e dei Thread

- descrittore di processo - **task_struct**
- Informazioni su
  - Scheduling parametri
  - Memoria immagine, puntatori ai segmenti testo, dati e stack
  - Segnali (maschere)
  - Registri da usare per le *trap*
  - Stato della chiamata di sistema, parametri, risultati
  - Tabella dei descrittori di file
  - Accounting
  - Stack del nucleo stack fisso che il kernel può usare
  - Altro stato del processo, segnale atteso, tempo del prossimo allarme, PID,…



23

## Organizzazione dei processi e dei Thread

- *Init*
  - Processo iniziale che usa il nucleo per creare tutti gli altri task
    - La chiamata di sistema **clone** crea nuovi task
    - La chiamata di sistema **fork** crea task che inizialmente condividono lo spazio di indirizzi con i genitori utilizzando copy-on-write (analogamente ai processi), la scrittura provoca la copia
    - Quando un processo fa una chiamata di sistema **clone** può specificare quali strutture dati condividere con il padre
      - Se lo spazio indirizzo è condiviso, clone crea un thread tradizionale
      - Se clone viene chiamato da un processo del nucleo, crea un thread del nucleo che condivide lo spazio di indirizzamento del nucleo
    - Anche se meno portabile di Pthread, i thread Linux possono facilitare la programmazione e migliorare l'efficienza delle applicazioni sfruttando la condivisione di risorse fra task
      - Library di Native POSIX Thread (NPTL) progetto di libreria di thread conformi a POSIX





## Creazione di processi in Linux - fork

```
pid = fork( );                  /* se il fork ha successo, pid > 0 nel padre */
if (pid < 0) {
      handle_error( );          /* fork ha fallito (per esempio, la memoria o una tabella è piena) */
} else if (pid > 0) {
                                /* il codice del padre va qui */
} else {
                                /* il codice del figlio va qui */
}
```

*fork* restituisce 0 al figlio e PID del figlio al padre





## Organizzazione dei processi e dei Thread

| Flag | Significato quando impostato | Significato quando non impostato |
|---|---|---|
| CLONE_VM | Crea un nuovo thread | Crea un nuovo processo |
| CLONE_FS | Condivide umask, directory principale e di lavoro | Non condividerli |
| CLONE_FILES | Condivide i descrittori dei file | Copia i descrittori dei file |
| CLONE_SIGHAND | Condivide la tabella dei gestori dei segnali | Copia la tabella |
| CLONE_PID | Il nuovo thread riceve il vecchio PID | Il nuovo thread riceve il suo PID |
| CLONE_PARENT | Il nuovo thread ha lo stesso padre del chiamante | Il padre del nuovo thread è il chiamante |

Bit nella sharing_flags bitmap - flags per la chiamata clone in Linux
Condivisione selettiva

A. Tanenbaum – Modern Operating Systems





## Scheduling dei Processi

- Obbiettivi dello Scheduler
  - Eseguire tutte le attività entro un ragionevole lasso di tempo
  - Rispettare le priorità dei task
  - Mantenere di un elevato utilizzo delle risorse
  - Alto throughput
  - Ridurre l'overhead di operazioni di scheduling
  - Scalabilità
- Tre classi di thread: Real-time FIFO, Real-time Round Robin, time-sharing
  - Real-time FIFO massima priorità senza prelazione da parte di altri tipi
  - Real-time RR con quanto di tempo
  - time-sharing priorità in [100,139]

  Nota: i task 'real-time' non hanno scadenze associate, hanno priorità in [0,99]
- Obbiettivo: tutte le operazioni di scheduling devono essere eseguite in tempo costante scheduler O(1)
  - Migliora la **scalabilità** se il tempo di esecuzione è indipendente dal numero di task nel sistema

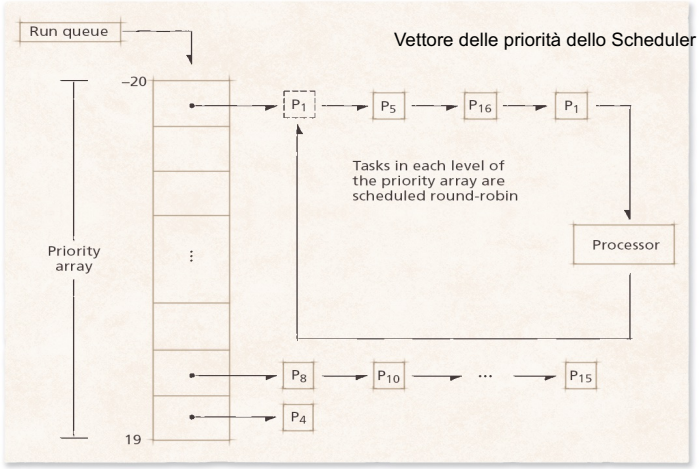
## Scheduling dei Processi

- Scheduler a prelazione
  - Ogni task è eseguito fino
    - allo scadere del suo quanto o intervallo di tempo
    - oppure diventa eseguibile un processo di priorità maggiore
    - oppure il processo si blocca
  - I task sono nella coda RUN (simili alle code multilivello con feedback)
  - Il vettore di priorità mantiene un puntatore a ogni livello della coda run
    - Un task con priorità *i* è posto nella coda della *i*-sima posizione del vettore di priorità della coda run
  - Lo Scheduler avvia il task in testa alla lista nel livello più alto del vettore di priorità
    - La coda per un livello priorità con più task è gestita dallo scheduler con round-robin
    - Quando un task entra in stato bloccato o *sleeping* (i.e., *waiting*) , o comunque non può essere eseguito, viene rimosso dalla sua coda run





## Scheduling dei Processi










## Scheduling dei Processi

- Per evitare l'attesa infinita, ogni task nella coda run è eseguito almeno una volta all'interno di un periodo detto epoca (*epoch*)
  - EPOCH è definita da un limite di massima attesa infinita, derivato empiricamente
    - Es.: 10 *n* sec. quando ci sono *n* task nella coda run
  - Lo Scheduler organizza i task in **due liste** con stati *active* e *expired*
    - Si possono eseguire solo i task nella lista *active*
    - Quando viene raggiunto il limite di attesa infinita, ogni task- quando scade il suo quanto- è spostato dallo stato *active* e inserito nella lista *expired*
    - Sospende temporaneamente i task ad alta priorità (eccetto quelli in tempo reale)
      - Garantisce che i task a bassa priorità alla fine siano eseguiti
    - Quando tutti sono nello stato *expired* (fine di un'epoca) sposta tutti i task da inattivi ad attivi e inizia una nuova epoca

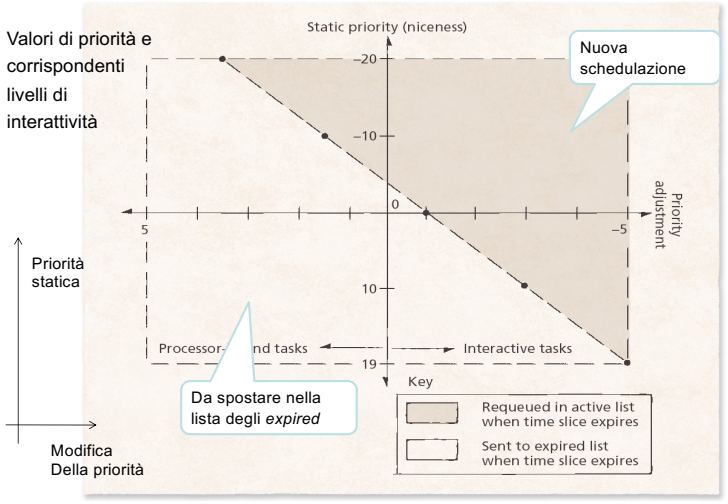




## Scheduling dei Processi

- Priorità
  - Ad ogni task creato è assegnata un valore interpretabile come priorità statica (detto anche valore di affinità - ***nice***) 0 di default
    - Modificabile con la chiamata `nice(value)`
    - 40 livelli in [-20, 19]
    - Valori bassi: priorità alta
  - Obbiettivo: alta interattività
  - I task sono schedulati in base alla loro effettiva priorità
    - I task I/O-bound ricevono la priorità alta
    - I task processor-bound sono penalizzati con priorità più bassa
  - Una task ad alta priorità può essere riprogrammato, ovvero variata dinamicamente la priorità, alla scadenza dell'intervallo di tempo
  - (entro una data soglia, e.g. differenza di 5 dalla prima priorità assegnata)

## Scheduling dei Processi







32

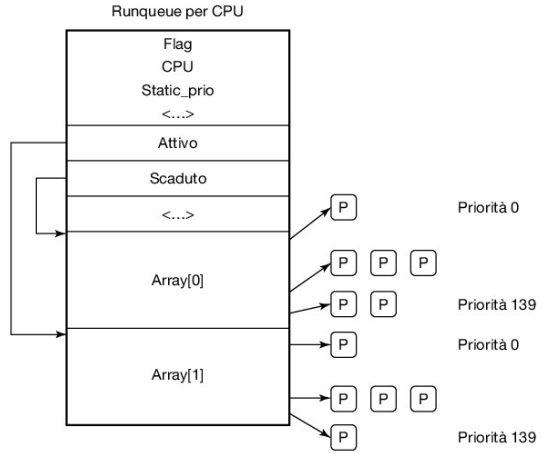
## Scheduling dei Processi

- Scheduler O(1)
  - Esecuzione a tempo costante
  - Runqueue divisa in due code (array): attivo (active) e scaduto (expired)
  - Ogni campo punta a un vettore di 140 liste di priorità (da 0 a 139)
  - La testa della lista punta ad una lista doppia di processi di quella priorità
  - Lo scheduler seleziona un task fra quelli a priorità massima
  - Se scade il quanto il task viene spostato nella lista dei processi scaduti (eventualmente ad una diversa priorità)
  - Quando un processo è bloccato al ritorno viene rimesso nell'array di task attivi originale, tenendo conto del tempo
  - Quando terminano i task attivi i puntatori delle due liste sono scambiati così gli scaduti diventano attivi e viceversa
  - Tempi di quanto diverso a diversi livelli di priorità



33

## Scheduling dei Processi




A. Tanenbaum – Modern Operating Systems


34

## Scheduling dei Processi

- Scheduling per sistemi multiprocessore
  - Una coda di task indipendente per ogni processore
  - Località della *cache*
  - Scheduler esegue il **bilanciamento dinamico del carico**
  - Cerca di ridurre lo sbilanciamento del carico, non bilanciare perfettamente le code run
  - Cerca di migrare solo i task *cache-cold*
    - la *cache* non contiene molti dei suoi dati
- Scheduling real-time
  - Scheduler soft real-time
  - I task RT possono usare una politica di scheduling round-robin, FIFO o di default
  - I task RT sono sempre rischedulati alla fine del quanto di tempo
  - I task RT possono essere creati solo da utenti con privilegi di root



35

## Gestione della memoria

- Gestore della memoria che supporta indirizzi a 32- e a 64-bit
- Supporta anche NUMA (Not Uniform Memory Access)

- Memoria logicamente divisa in tre segmenti: testo, dati e stack
- Il segmento dati contiene dati inizializzati e dati non inizializzati (chiamati *BSS- block started by symbol*)

- Solitamente nella memoria fisica *page frame* di dimensione **fissa**
  - Es.: 4KB, 8KB, *4 MB*
  - Informazioni relative alla pagina in una struttura page
    - # processi che condividono la pagina, bit *dirty*, indicatori di stato

- Memoria Virtuale
- Paginazione





## Organizzazione della memoria virtuale

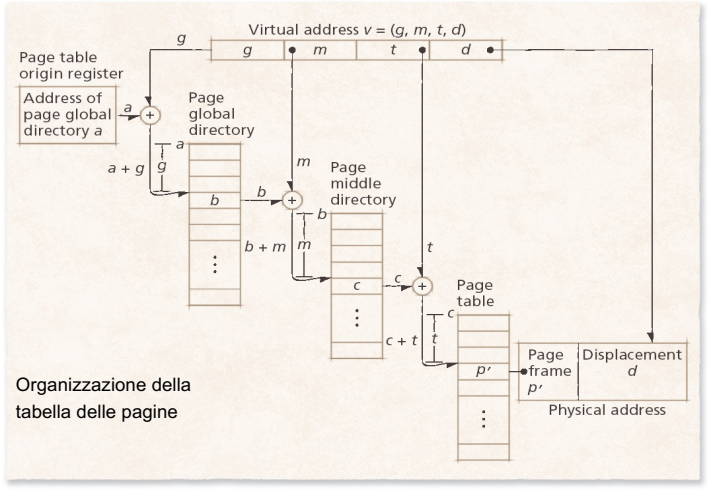
- Linux usa esclusivamente **paginazione**
  - Spesso implementato utilizzando dimensione della pagina fissa
    - Nei sistemi a 32-bit, il nucleo può indirizzare 4 GB di dati
    - Nei sistemi a 64 bit, il nucleo supporta fino a 2 Petabyte di dati (Milioni di GB)
  - Tre (o quattro) livelli di **tabelle di pagina**
    - **directory globale** di Pagina
    - (**directory alta** di Pagina)
    - **directory intermedia** di Pagina
    - **tabelle delle pagine**
  - Su sistemi che supportano solo due livelli di tabelle delle pagine, la directory media di pagina ha solo una riga

- Per un processo lo spazio di indirizzamento virtuale è organizzato in aree di memoria virtuale per raggruppare le informazioni con stesse autorizzazioni (simile a segmenti)





## Organizzazione della memoria



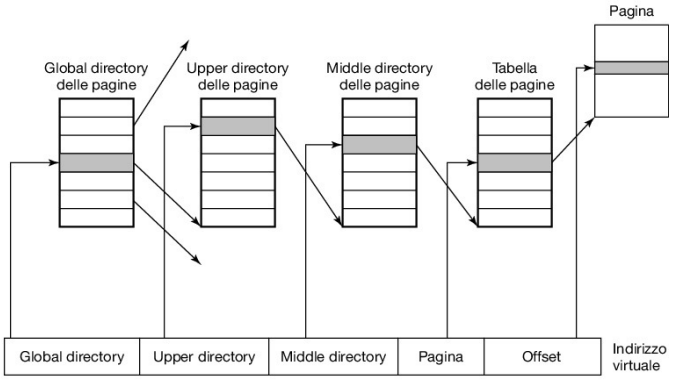

Organizzazione della tabella delle pagine







## Organizzazione della memoria - paginazione




Uso delle tabelle a quattro livelli

*(dalla versione 2.6.11)*

A. Tanenbaum – Modern Operating Systems

## Organizzazione della memoria virtuale

- Linux sull'architettura IA-32
  - Il nucleo cerca di ridurre l'overhead dovuto al cambiamento di contesto, per lo svuotamento (*flushing*) della memoria associativa TLB (*Translation Lookaside Buffer*) contenente le righe delle tabelle delle pagine usate più di recente (*Page Table Entries*)
  - Ogni spazio di indirizzamento di 4GB è diviso in una regione con
    - I primi 3GB per dati e istruzioni del processo e
    - 1GB per lo spazio di indirizzamento per dati e istruzioni del nucleo (non visibile in modalità utente)
  - L'invocazione del nucleo da parte di un processo non provoca lo svuotamento della TLB: migliori prestazioni
  - La maggior parte dello spazio di indirizzamento del nucleo è mappata direttamente in memoria principale in modo che possa accedere alle informazioni appartenenti a qualsiasi processo





## Organizzazione della memoria fisica

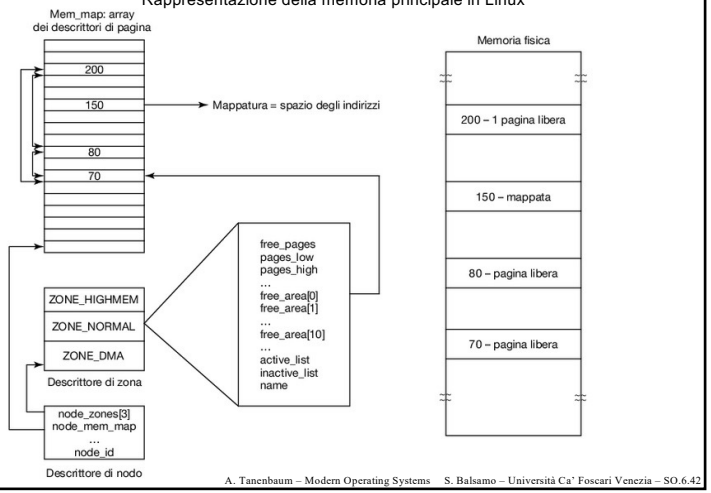
- Tre **zone** di memoria fisica
  - Memoria DMA: i primi 16MB di memoria principale
    - Il nucleo tenta di rendere la memoria disponibile in questa regione per l'hardware legacy
  - Memoria normale: tra i 16 MB e 896MB sull'architettura IA-32
    - Memorizza i dati utente e la maggior parte dei dati del nucleo
  - Memoria alta : > 896MB sull'architettura IA-32
    - Contiene la memoria che il nucleo non mappa in modo permanente al suo spazio di indirizzamento, e memoria per processi utente
- (*Bounce buffer*) buffer di rimbalzo
  - Per dispositivi che non possono indirizzare la memoria alta: alloca una piccola parte di memoria temporanea nella zona DMA per I/O
  - I dati vengono "rimbalzati" alla memoria alta (copiati) dopo che l'operazione di I/O è completata





## Organizzazione della memoria

Rappresentazione della memoria principale in Linux




A. Tanenbaum – Modern Operating Systems

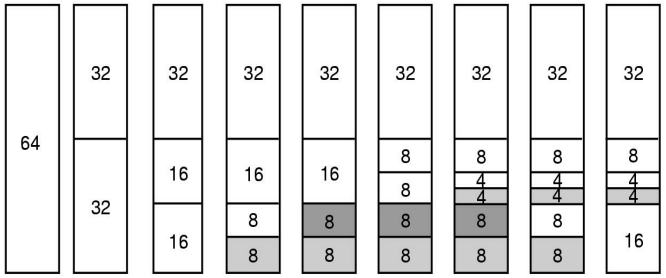




## Allocazione e deallocazione della memoria fisica

- Allocatore di **Zona**
  - Alloca ai processi page frame di memoria alta, se disponibile
    - Altrimenti, alloca dalla memoria normale, se disponibile
    - Alloca dalla memoria bassa, se non c'è altra memoria disponibile
  - Usa il vettore free_area di ogni zona
    - liste libere e maschera di bit per blocchi di memoria contigui
    - Blocchi di dimensione $2^n$ $n=0,1,2,\ldots$
  - Algoritmo *binary buddy* per trovare i blocchi di page frame contigui di dimensioni adatte al processo nel vettore free_area
    - Cerca un blocco di dimensioni corrette, se non esiste inizia da un blocco più grande e progressivamente lo dimezza e itera - nella deallocazione riunisce i liberi vicini
- Allocatore di Slab (lastre)
  - Alloca la memoria per strutture più piccole di una pagina
  - Slab cache: formata da un insieme di oggetti slab - struttura per contenere strutture dati multiple (dello stesso tipo) più piccole di una pagina
- Memory pool
  - Regione della memoria che il nucleo garantisce come disponibile per thread del nucleo o driver di periferica, indipendentemente dal carico di memoria, per evitare *page fault* critici

## Allocazione e deallocazione della memoria fisica




Allocatore di pagine - Algoritmo *buddy*

*Esempio*:

- Richieste di allocazione: 8 pagine, 8 pagine, 4 pagine
- Rilascio di 8 pagine, poi di 8 pagine -> buddy riunisce in 16 pagine

Frammentazione interna

A. Tanenbaum – Modern Operating Systems





## Sostituzione di Pagine

- Caratteristiche generali
  - Possono essere sostituite solo le pagine degli utenti
- Paginazione a richiesta
  - Come le pagine vengono lette in memoria, il nucleo le inserisce nella **cache** delle pagine
    - Le pagine sporche sono scaricate su disco con **cache write-back**
    - Pagine cache associate ad un dispositivo di memoria secondaria dove scaricarle (*swap out*) e se di un file ad un *i-node* (posizione in memoria secondaria)
    - file **swap** di sistema
      - Regione di memoria secondaria per scaricare e memorizzare le pagine (non collegate a file) dei dati e procedure di programmi





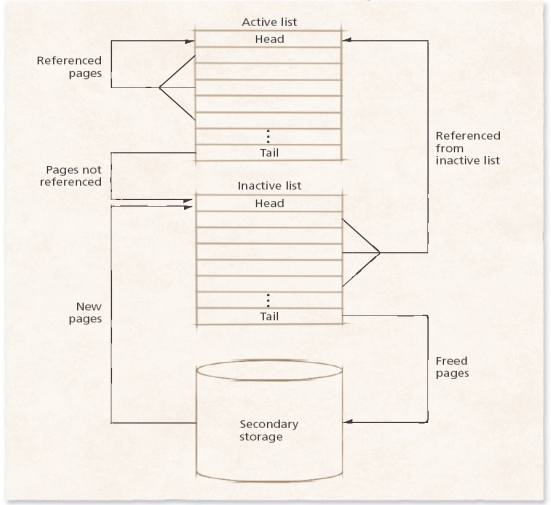
## Sostituzione di Pagine

- La sostituzione di pagina è eseguita indipendentemente per ogni zona
- L'algoritmo è una variante dell'algoritmo di sostituzione di pagina a orologio
  - **Due liste** collegate per ogni zona (contenenti le strutture page)
    - La **Lista attiva**: pagine che sono state riferite di recente (*working set*)
    - La **Lista Inattiva**: pagine che sono state utilizzate meno di recente
  - Una pagina entra nel sistema in testa alla lista inattiva, con il bit di riferimento on
  - Se la pagina è attiva o inattiva e il suo bit di riferimento è off, il bit è attivato
    - Assicura che le pagine alle quali è stato fatto riferimento di recente non siano selezionate per la sostituzione
  - Se la pagina è inattiva ed è stato fatto riferimento per la seconda volta (bit di riferimento è on), la pagina viene spostata in testa alla lista attiva, e il bit di riferimento è azzerato
    - Permette al nucleo di distinguere tra le pagine di riferimento che sono state riferite una volta e quelle che sono stati riferite più di una volta di recente
    - Queste ultime sono inserite nella lista attiva in modo che non siano selezionate per la sostituzione

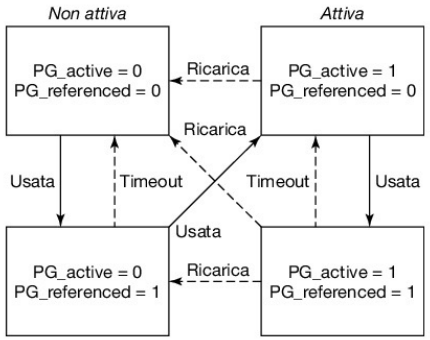




## Sostituzione di Pagine



## Allocazione e deallocazione della memoria fisica




Stati della pagina usati dall'algoritmo di sostituzione delle pagine PFRA (*Page Frame Reclaiming Algorithm)*



48

## Swapping

- *kswapd* (il demone *swap* del nucleo)
  - Libera periodicamente page frame tramite scaricamento di pagine sporche su disco (*swap out*)
  - Scambia pagine dalla coda della lista inattiva
    - Prima determina se la pagina ha una riga valida nella **cache di swap** (righe della tabella delle pagine per cui esiste un file *swap*)
      - Consente di liberare immediatamente le pagine non modificate
    - Non può liberare pagine libere se
      - La pagina è condivisa
        - kswapd deve eliminare il mapping di riferimenti multipli alla pagina
        - La mappatura inversa (contiene le righe della tabella che riferiscono la pagina) migliora l'efficienza
      - La pagina è modificata (*dirty*)
        - kswapd deve scaricarla su disco
        - Eseguita in modo asincrono da *pdflush*
      - La pagina è *locked* (es.: attualmente in fase di I /O)
        - kswapd deve aspettare finché la pagina è sbloccata



49

## File System

- Ogni particolare file system determina come memorizzare e accedere ai suoi dati
- Un file si riferisce ad un insieme di bit in memoria secondaria ed è un punto di accesso ai dati
  - che possono trovarsi su un disco locale, in rete, o anche generati dal nucleo stesso
- Consente al nucleo di accedere utilizzando una singola interfaccia di sistema di file generici a
  - dispositivi hardware
  - meccanismi di comunicazione tra processi,
  - dati memorizzati su disco
  - diverse altre fonti di dati
- Il nucleo supporta numerosi file system caricabili come moduli
  - *Es. in Linux 2.6 oltre 40 file systems integrabili*
  - *Es. ext2, FAT, UFD, NFS, Coda, procf…*



50

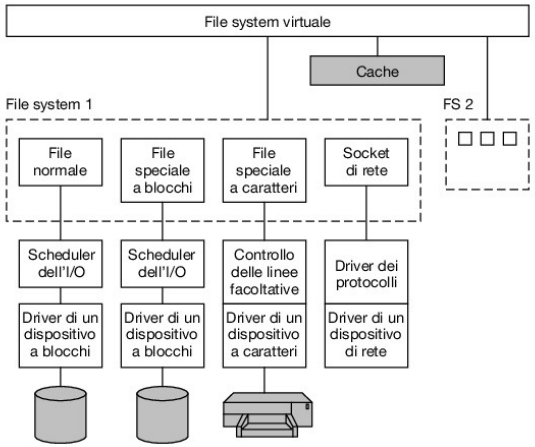
## Virtual File System

- VFS – Strato per supportare diversi file systems
  - Astrazione dai dettagli di accesso ai file, consentendo agli utenti di visualizzare tutti i file e le directory nel sistema in un unico albero di directory
  - Tutte le richieste relative ai file vengono inizialmente inviati al livello VFS, che fornisce **un'interfaccia** per l'accesso dati dei file su qualsiasi file di sistema disponibile
  - I processi effettuano chiamate di sistema come read, write e open, che vengono passate al file system virtuale
    - VFS determina il file system al quale corrisponde la richiesta e
    - richiama la routine corrispondente nel driver del file system, che esegue le operazioni richieste
  - Trasparenza, flessibilità, espandibilità



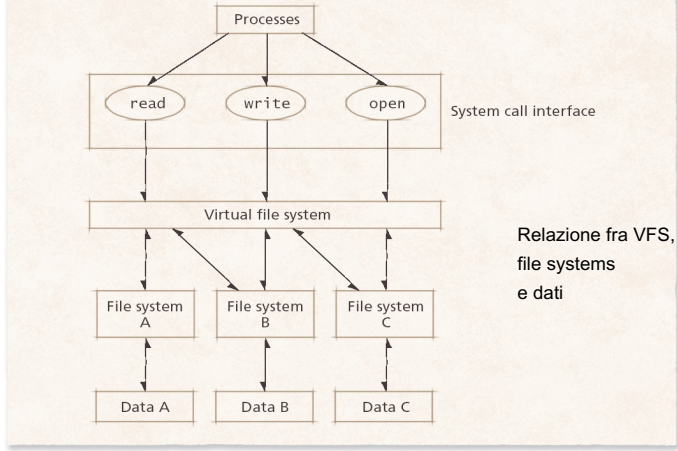
51

## Virtual File System






## Virtual File System




Relazione fra VFS, file systems e dati



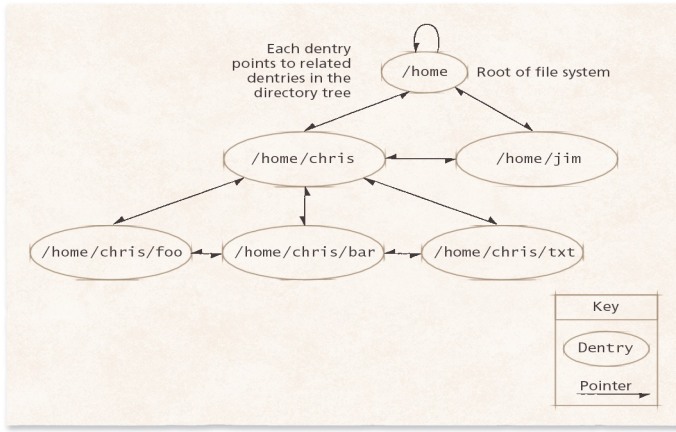
## Virtual File System

- VFS *inode*
  - Descrive la locazione di ogni file, directory o un link all'interno di ogni file system disponibile
  - Riferimento ad ogni file da un numero di *inode* e numero di file system
- Descrittore di file
  - contiene:
    - Informazioni sul *inode* a cui si accede
    - Informazioni sulla posizione del file a cui si accede
    - Flag che descrivono come accedere ai dati (es.: lettura/scrittura, append-only)
- Dentry (*Directory* entry - riga o voce di directory)
  - Mappa i descrittori di file negli *inode*
  - Contiene il nome del file o directory che un *inode* rappresenta
  - Puntatori alle *Dentry* dei genitori e figli



## Virtual File System

Organizzazione delle Dentry per una particolare `/home` directory



## Directories

Alcune directory nella maggior parte dei sistemi Linux

| Directory | Contenuti |
|---|---|
| bin | Programmi binari (eseguibili) |
| dev | File speciali per i dispositivi di I/O |
| etc | File di sistema vari |
| lib | Librerie |
| usr | Directory degli utenti |



## Directories

Alcune chiamate di sistema relative a directory in sistemi Linux

| Chiamata di sistema | Descrizione |
|---|---|
| s = mkdir(path, mode) | Crea una nuova directory |
| s = rmdir(path) | Rimuove una directory |
| s = link(oldpath, newpath) | Crea un link a un file esistente |
| s = unlink(path) | Toglie il link al file |
| s = chdir(path) | Cambia la directory di lavoro |
| dir = opendir(path) | Apre una directory in lettura |
| s = closedir(dir) | Chiude una directory |
| dirent = readdir(dir) | Legge una sola voce della directory |
| rewinddir(dir) | Riavvolge una directory così che possa essere riletta |



## Virtual File System

- Montaggio di file system
- **Superblocco** VFS
  - Contiene informazioni su un file system montato, p.es.:
    - Il tipo di file system
    - La posizione del suo *inode* radice sul disco
    - Informazioni che proteggono l'integrità del file system
  - Memorizzato solo in memoria principale, creato quando FS è montato
- Il VFS definisce le operazioni generiche del file system
  - Richiede che ogni file system fornisca un'implementazione per ogni operazione che supporta
  - Ad esempio, il VFS definisce una funzione read , ma non la implementa



## Virtual File System

Astrazioni del file system supportate da VFS

| Oggetto | Descrizione | Operazione |
|---|---|---|
| Superblock | File system specifico | read_inode, sync_fs |
| Dentry | Voce della directory, singola componente di un percorso | create, link |
| I-node | File specifico | d_compare, d_delete |
| File | Apri il file associato a un processo | read, write |

## Virtual File System

Operazioni del VFS su file e inode

| VFS operation | Intended use |
|---|---|
| read | Copy data from a file to a location in memory. |
| write | Write data from a location in memory to a file. |
| open | Locate the inode corresponding to a file. |
| release | Release the inode associated with a file. This can be performed only when all open file descriptors for that inode are closed. |
| ioctl | Perform a device-specific operation on a device (represented by an inode and file). |
| lookup | Resolve a pathname to a file system inode and return a dentry corresponding to it. |




60

## Secondo File System esteso (ext2fs)

- caratteristiche di Ext2
  - Obbiettivo: elevate prestazioni, file system robusto con il supporto alle funzioni avanzate
  - Tipiche dimensioni dei blocchi: 1.024, 2.048, 4.096 o 8.192 byte
  - Per default, 5% dei blocchi sono riservati esclusivamente agli utenti con privilegi di root quando il disco è formattato
    - previsto un meccanismo di sicurezza per consentire ai processi di root di continuare l'esecuzione se un processo utente malintenzionato o in errore consuma tutti gli altri blocchi disponibili nel file system



61

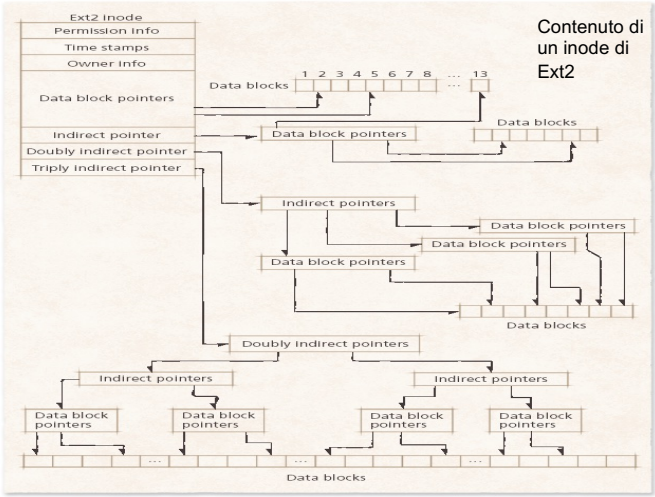
## Secondo File System esteso (ext2fs)

- ext2 i-node
  - Rappresenta file e directory in un file system ext2
  - Memorizza le informazioni rilevanti per un singolo file o directory, es: data e ora, autorizzazioni, identità del proprietario e puntatori ai blocchi di dati
    - I primi 12 puntatori individuano direttamente i primi 12 blocchi di dati
    - 13° puntatore è un puntatore indiretto che individua un blocco che contiene i puntatori ai blocchi di dati
    - 14° è un puntatore doppiamente indiretto e individua un blocco di puntatori indiretti
    - 15° puntatore è un puntatore a triplo indirizzamento indiretto individua un blocco di puntatori doppiamente indiretti
  - Fornisce un accesso rapido ai file piccoli, pur supportando file di dimensioni maggiori



62

## Secondo File System esteso (ext2fs)







63

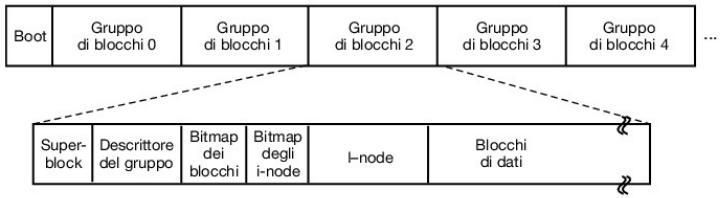
## Secondo File System esteso (ext2fs)

- Gruppi di blocchi
  - Clusters di blocchi contigui
  - Il F.S. tenta di memorizzare i dati correlati nello stesso gruppo di blocchi
  - Riduce il tempo di ricerca per l'accesso ai grandi gruppi di dati correlati
  - contiene
    - il superblocco
      - Le informazioni critiche circa l'intero FS, non solo un particolare gruppo di blocchi
        - Include il n. totale di blocchi e inode nel file system, la dimensione dei gruppi di blocchi, il tempo in cui il file system è stato montato e altri dati
        - Una copia ridondante del superblocco è mantenuta in alcuni gruppi di blocchi
    - Tabella degli inode
      - Contiene una riga per ogni inode nel gruppo di blocco
    - allocazione bitmap degli Inode
      - Traccia l'uso degli inode all'interno di un gruppo di blocchi





## Secondo File System esteso (ext2fs)




Struttura su disco del file system ext2 in Linux

A. Tanenbaum – Modern Operating Systems





## Secondo File System esteso (ext2fs)

- Gruppi di blocchi
  - Contiene
    - bitmap di allocazione del blocco
      - Traccia l'uso dei blocchi di ogni gruppo
    - Descrittore di Gruppo
      - Contiene i numeri di blocco corrispondenti alla posizione della bitmap di allocazione di i-node, bitmap di allocazione di blocco e i-node, informazioni di accounting





## Secondo File System esteso (ext2fs)

- Gruppi di blocchi
  - Contiene
    - I blocchi rimanenti in ogni gruppo di blocchi memorizzano i dati di file/directory
      - Le informazioni delle directory sono memorizzate in righe della directory
        - Ogni riga della directory è composta da un numero di i-node, dalla lunghezza della riga della directory, lunghezza del nome del file, tipo di file e il nome del file
- Sicurezza del File
  - Permessi del File
    - Specificano i privilegi read, write execute per le tre categorie di utente: Owner, group, other
  - Attributi del File
    - Controllo di come si può modificare il file
    - P.es.: append-only

## Proc File System

- Procfs
  - Creato per fornire informazioni in tempo reale sullo stato del nucleo e i processi di sistema
  - Consente agli utenti di ottenere informazioni dettagliate che descrivono il sistema, dalle informazioni di stato hardware per i dati che descrivono il traffico di rete
  - Esiste solo nella memoria principale
    - I dati del file proc sono creati su richiesta
    - Le chiamate procfs read e write possono accedere ai dati del nucleo
      - Permette agli utenti di inviare i dati al nucleo





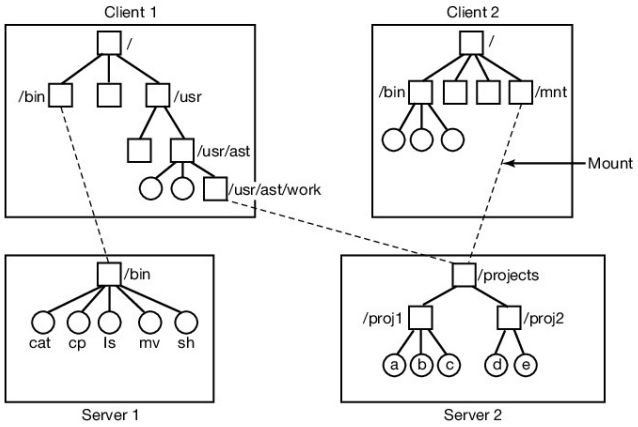
## Network File System - NFS

- Network File System – NFS
  - Introdotto da Sun Microsystem
  - diverse versioni
  - NFS 3 1994 - diffusa
  - NFS4 2000
  - Caratteristiche
    - Architettura client server
    - Protocollo
    - Implementazione
  - Directory esportabili */etc/export*
  - mount di directory remote





## Network File System - NFS



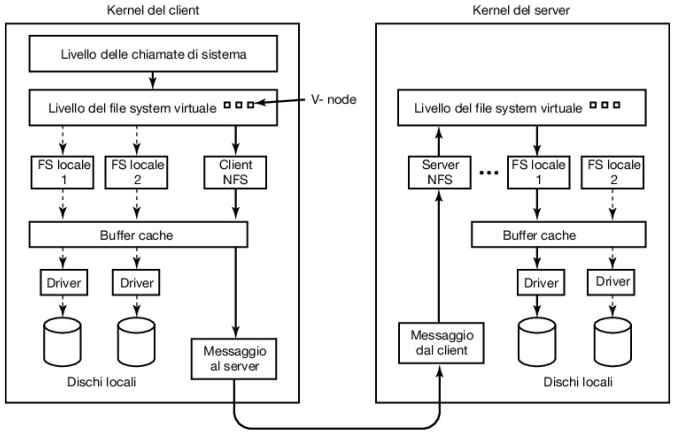

Esempio di mount di file system remoti







## Network File System - NFS




Struttura a livelli di NFS

## Gestione Input/Output

- I nucleo fornisce una interfaccia comune per le chiamate di sistema di I/O
- Le periferiche sono raggruppate in classi
  - I membri di ciascuna classe di dispositivi svolgono funzioni simili
  - Permette al nucleo di soddisfare le esigenze di prestazioni di alcuni dispositivi (o classi di dispositivi) singolarmente





## Drivers dei dispositivi

- Device driver: interfaccia sw tra chiamate di sistema e un dispositivo hardware
  - La maggior parte sono stati scritti da sviluppatori indipendenti
  - In genere implementato come moduli caricabili del nucleo
- File speciali di dispositivo
  - La maggior parte dei dispositivi sono rappresentati da file speciali di dispositivo
  - Le righe della directory /dev forniscono l'accesso ai dispositivi
  - La lista dei dispositivi del sistema può essere ottenuta leggendo il contenuto di /proc/devices





## Drivers dei dispositivi

- Classi di dispositivi
  - Gruppi di dispositivi che eseguono funzioni simili
- Numeri di identificazione principali e secondari
  - Usati dai driver di periferica per identificare i loro dispositivi
  - I dispositivi assegnati lo stesso numero di identificazione principali sono controllati dallo stesso driver
  - numeri di identificazione secondari consentono al sistema di distinguere tra i dispositivi della stessa classe



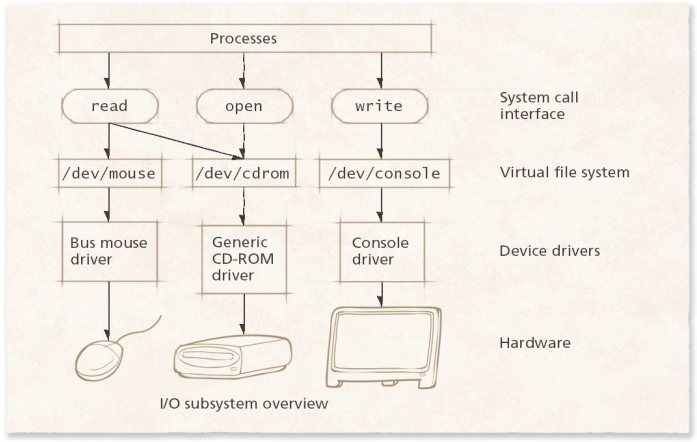


## Drivers dei dispositivi

- File speciali di dispositivi sono accessibili tramite il virtual file system
  - Le chiamate di sistema passano al VFS, che a sua volta chiama il driver di periferica
  - La maggior parte dei driver implementano operazioni di file comuni, come read, write e seek
  - Per sostenere le attività come ad esempio l'espulsione di un CD-ROM o il recupero di informazioni sullo stato di una stampante, Linux fornisce la chiamata di sistema ioctl

## Drivers dei dispositivi

Livelli di interfaccia di I/O








76

## Network Device I/O

- Network I/O
  - Si può accedere all'interfaccia di rete Network solo indirettamente da un processo utente attraverso IPC e l'interfaccia socket
  - Il traffico di rete può arrivare in qualsiasi momento
    - Le operazioni read e write di un file speciale di dispositivo non sono sufficienti per accedere ai dati da dispositivi di rete
    - Il nucleo usa strutture net_device per descrivere i dispositivi di rete
    - Nessuna struttura file_operations
- Elaborazione dei Pacchetti
  - Una volta che il nucleo ha preparato pacchetti da trasmettere a un altro host, li passa al driver di periferica per la appropriata scheda di interfaccia di rete (NIC)



77

## Network Device I/O

- Il nucleo esamina una tabella di routing interna per abbinare l'indirizzo di destinazione del pacchetto all'interfaccia appropriata nella tabella di routing
- Poi il nucleo passa il pacchetto al driver di periferica
  - Ciascun driver elabora pacchetti secondo una disciplina di accodamento, che specifica l'ordine sul suo dispositivo
- Il nucleo si sveglia il dispositivo per inviare i pacchetti
- Quando i pacchetti arrivano, il dispositivo di rete lancia un interrupt
  - Il nucleo copia il pacchetto e lo passa al sottosistema di rete



78