

OpenBSD Internals: What makes it different

# OpenBSD FileSystem

---

- OpenBSD FileSystem: teoria && implementazione && utilizzo

Autore

Michele "pirroH" Catasta  
email: [pirroh<at>gmail.com](mailto:pirroh@gmail.com)

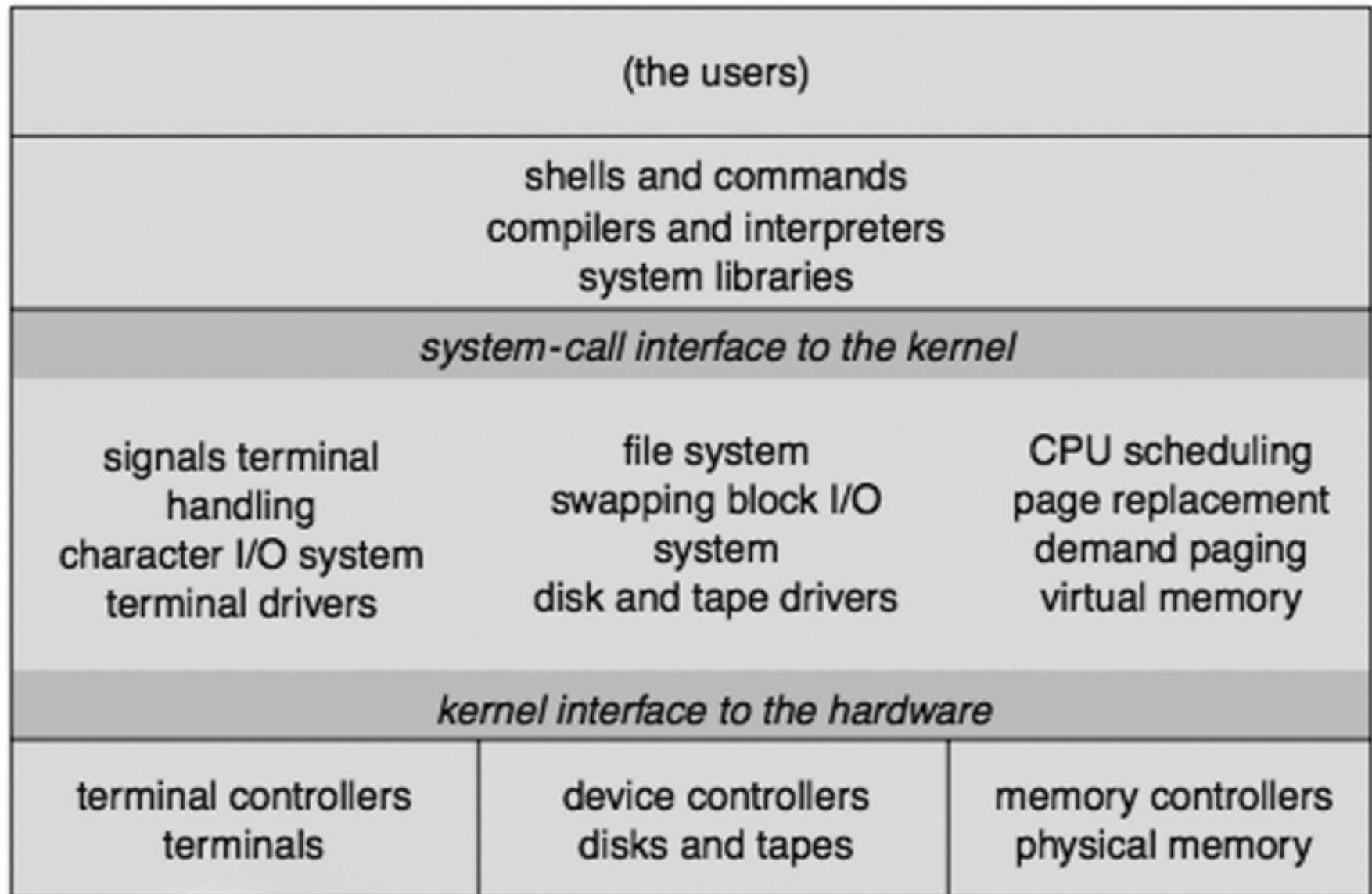
Oh, my head hurts bad.  
Rings of ones and zeros, ouch!  
File systems hide them.



OpenBSD Internals: What makes it different

# BSD Layer Structure

---



4.4BSD layer structure

## Intro FileSystem (1)

---

- Insieme dei tipi di dati astratti necessari per la memorizzazione, l'organizzazione gerarchica, la manipolazione, la navigazione, l'accesso e la lettura dei dati.
  
- Si appoggia solitamente a dispositivi di archiviazione che offrono l'accesso ad un array di blocchi di dimensione fissa (settori), tipicamente di 512 byte l'uno.
  
- Organizza i settori in file e directory, tiene traccia di quali settori appartengono a quali file, e quali invece non sono utilizzati.

# OpenBSD Internals: What makes it different

## Intro FileSystem (2)

---

### □ Varie tipologie disponibili:

- Disk FileSystem: progettato per memorizzare dei file su un'unità a disco, che può essere collegata direttamente o indirettamente al computer (FFS/UFS, Ext2/3, HFS(+), ReiserFS, XFS...).
- Network FileSystem: permette di accedere ai file contenuti su un computer remoto tramite rete, anche in simultanea da diversi computer (NFS, CIFS, AFS, Coda, GFS...).
- Database FileSystem: permette di identificare i file attraverso i loro metadati e di ricercarli attraverso query SQL o in linguaggio naturale (GNOME Storage, WinFS, Spotlight...).
- Special Purpose FileSystem: solitamente generati dinamicamente tramite software (DEVFS, PROCFS, TMPFS...).

### □ Offre solitamente criteri di sicurezza nell'accesso alle operazioni, in genere basati su ACL (Access Control List).

- Una ACL permette di definire per ciascun elemento del file system di quali permessi (lettura, scrittura, modifica ecc.) dispone ciascun utente.

OpenBSD Internals: What makes it different

# Intro Unix FileSystem

---

- Unix utilizza un file system virtuale, dove tutti i file di ogni device appartengono ad una sola gerarchia. Ossia, data una directory root, qualsiasi file esistente puo' essere localizzato nell'albero sotto forma di foglia (path name).
- La directory root puo' non essere relativa ad un device fisico del proprio computer (ad esempio una risorsa condivisa in rete).

OpenBSD Internals: What makes it different

# File Operations

---

- Un file in Unix e' una sequenza di byte non necessariamente strutturata (sebbene molti programmi organizzino i loro file come sequenze di record).
  
- Operazioni standard Unix/POSIX su files:
  - o `open()`: apre un file esistente o ne crea uno nuovo
  - o `creat()`: crea un nuovo file
  - o `close()`: chiude un file gia' aperto
  - o `lseek()`: posizionamento in un file
  - o `read()`: legge dati dalla posizione corrente
  - o `write()`: scrive dati partendo dalla posizione corrente

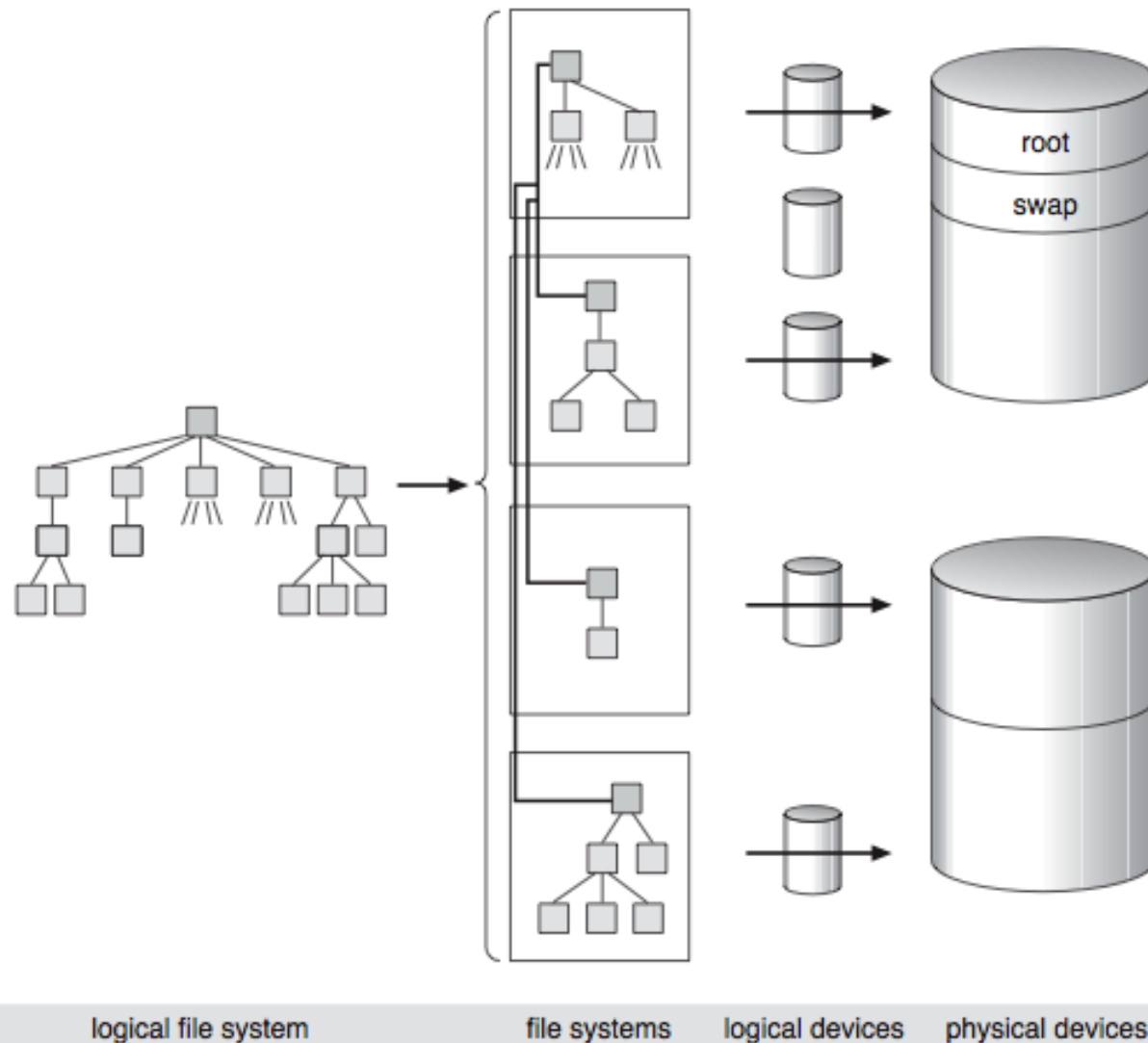
## Hard and Soft Link

---

- Un file e' individuabile attraverso uno o piu' nomi (hard link). Il numero degli hard link di un file rappresenta il reference count. Il file viene effettivamente cancellato solo quando `reference count==0`.
  
- "." e ".." sono hard link rispettivamente alla directory corrente e alla directory superiore.
  
- \*BSD supporta anche i symbolic link (file contententi il path name di un altro file). Possono, a differenza degli hard link, puntare a directory e a risorse esterne al filesystem stesso.

# OpenBSD Internals: What makes it different

## Logical FileSystems & Physical Devices



## Device Nodes

---

- Interfaccia logica (attraverso un "file") verso un elemento hardware. Si distinguono in Raw e Block devices (con scopi mutuamente esclusivi).
  
- Raw devices: detti anche character device (permettono l'accesso "un char alla volta"). Strumenti di indiscussa potenzialita' e pericolosita'!
  
- Block devices: bufferizzano il flusso dei dati (con conseguente miglioramento delle prestazioni).

OpenBSD Internals: What makes it different

# OpenBSD Disk Management

---

- IDE disk: /dev/wd?? SCSI disk: /dev/sd?? -> ? = numero (dmesg), ? = lettera (partizione [c = intero disco])
- Numerazione dipendente dall'ordine di identificazione (non dalla posizione nel bus).
- Disponibili due potenti strumenti per la costruzione di filesystem flessibili: CCD (Concatenated Disk Driver) e RAIDFrame.

OpenBSD Internals: What makes it different

## CCD & RAIDFrame

---

- CCD: permette la creazione di un filesystem distribuito su piu' partizioni (non necessariamente delle stesse dimensioni e/o appartenenti allo stesso disco). Adatto per lo storing di grandi moli di dati (senza alcuna assicurazione sull'integrita').
- RAIDFrame: permette la creazione di RAID 0, 1, 4 e 5 via software (piu' flessibile di CCD, supporto RAID anche per il root filesystem). Come CCD, non e' attivo di default nel kernel!

## Disk Images

---

- E' possibile mountare una disk image come se fosse una partizione (ad esempio ISO cdrom image o floppy image). Prima del mount, occorre associare l'immagine ad un device node. OpenBSD fornisce vnconfig(8) per tale associazione.
- Un vnode e' un layer di astrazione tra il kernel e il file system. Il kernel GENERIC ha 4 vnode device (/dev/vnd\* e /dev/svnd\* -> safe vnode buffered).

## VFS

---

- Rende netta la separazione tra un filesystem indipendente e le operazioni/data structure dipendenti da esso tramite l'utilizzo di due layer: uno dipendente e uno indipendente dal filesystem stesso.
  
- Il vantaggio risiede nel fatto che alcune complicate funzioni (namei, lookup, etc.) non debbono essere reimplementate ogni volta, poiche' sono contenute nel layer indipendente.
  
- OpenBSD Filesystem HowTo: <http://glot.net/openbsd-filesystem-howto/filesystem-howto.pdf>
-

OpenBSD Internals: What makes it different

## Supported FileSystems

---

```
opencon:/usr/src/sys pirroH$ echo 'ls|grep fs'
adosfs isofs miscfs msdosfs nfs ntfs ufs xfs
opencon:/usr/src/sys/miscfs pirroH$ echo 'ls|grep
fs'
deadfs fifofs genfs kernfs nullfs procfs specfs
umapfs
opencon:/usr/src/sys/ufs pirroH$ echo 'ls|grep fs'
ext2fs ffs lfs mfs ufs
```

- Supporto FFS/UFS, NFS, ext2fs, FAT(32), iso9660...
- MFS: memory file system. Permette di creare virtual ram disk molto performanti (partizioni per compilare, comprimere, etc.).

# OpenBSD Internals: What makes it different

## UFS (1)

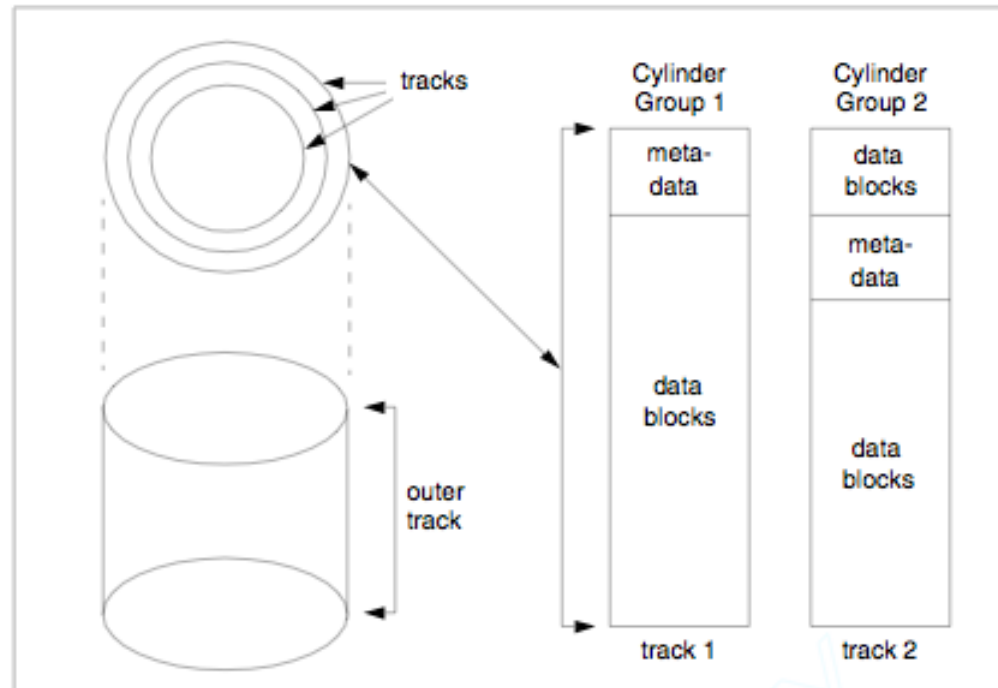
---

- Unix File System, derivato da Fast File System. Ha subito un processo di inesorabile evoluzione, paralla a quella dell'hardware sui cui si appoggia (specialmente per quanto riguarda le policy di allocazione).
  
- Composto dai seguenti componenti:
  - Alcuni blocchi all'inizio della partizione riservati per i boot block
  - Un superblocco, contenente un magic number che lo identifica come UFS, e alcuni dati riguardo la geometria del filesystem, parametri di tuning, etc.
  - Un insieme di cylinder group composto da:
    - ▷ Un copia di backup del superblock
    - ▷ L'header del cylinder group con dati statistici, lista dei blocchi liberi
    - ▷ Un array di inode, ognuno dei quali contiene gli attributi dei file (metadata)
    - ▷ Blocchi dati

# OpenBSD Internals: What makes it different

## UFS (2)

---



Mapping the UFS filesystem to underlying disk geometries.

- Ridondanza del superblock, offset nel posizionamento (problema del primo piatto), etc. etc.

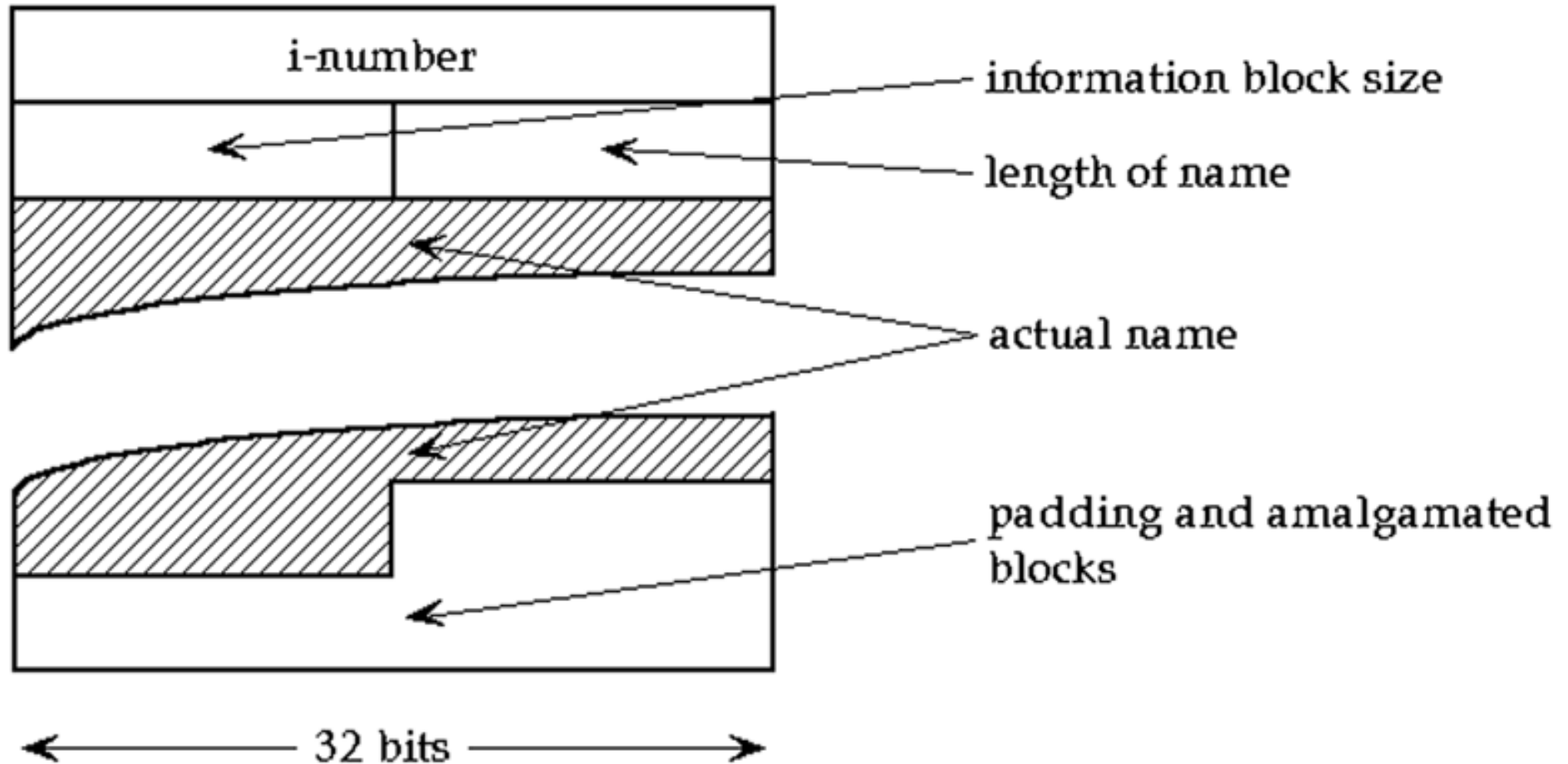
OpenBSD Internals: What makes it different

## Blocks & Fragments

---

- La dimensione di un blocco e' di 8kB nei recenti UFS (valore standard per OpenBSD). Rappresenta un buon compromesso tra miglioramento delle prestazioni grazie al read-ahead e incremento della frammentazione interna.
- Molti FS usano il concetto di frammento (divisione logica di un blocco) per minimizzare la frammentazione interna. (1kB valore standard OpenBSD).

# OpenBSD Internals: What makes it different Directory



- Associa il nome del file con il suo i-number (attraverso una struttura dinamica).

# OpenBSD Internals: What makes it different

## Inode (1)

---

Type/Mode	Link Count
User Id	Group Id
File Size (in bytes)	
Last Accessed Time	
Last Modified Time	
Created Time	

12 data block addresses

First level index block address
Second level index block address
Third level index block address

Extensions (7 4 byte words)

# OpenBSD Internals: What makes it different

## Inode (2)

---

□ /usr/src/sys/ufs/ufs/dinode.h

```
/*
 * A dinode contains all the meta-data associated with a UFS file.
 * This structure defines the on-disk format of a dinode. Since
 * this structure describes an on-disk structure, all its fields
 * are defined by types with precise widths.
 */
```

□ /usr/src/sys/ufs/ufs/inode.h

```
/*
 * The inode is used to describe each active (or recently active) file in the
 * UFS filesystem. It is composed of two types of information. The first part
 * is the information that is needed only while the file is active (such as
 * the identity of the file and linkage to speed its lookup). The second part
 * is * the permanent meta-data associated with the file which is read in
 * from the permanent dinode from long term storage when the file becomes
 * active, and is put back when the file is no longer being used.
 */
```

OpenBSD Internals: What makes it different

# Dirpref (1)

---

□ /usr/src/sys/ufs/ffs/ffs\_alloc.c

```
/*
 * Find a cylinder group to place a directory.
 *
 * The policy implemented by this algorithm is to allocate a
 * directory inode in the same cylinder group as its parent
 * directory, but also to reserve space for its files inodes
 * and data. Restrict the number of directories which may be
 * allocated one after another in the same cylinder group
 * without intervening allocation of files.
 *
 * If we allocate a first level directory then force allocation
 * in another cylinder group.
 */
```

```
static ino_t    ffs_dirpref(struct inode *);
```

□ <http://www.ptci.ru/gluk/dirpref/old/dirpref.html>

## Dirpref (2)

---

- Patch di Grigoriy Orlov, testata su OBSBSD e adottata ufficialmente dalla versione 2.9 (Apr '01). Diminuisce in modo tangibile il seek time (principio di localita').
  
- Sostituisce la precedente policy, che si limitava a scegliere il cylinder group con meno directory allocate (non ottimalita' causata dalla distribuzione indiscriminata di directory su tutto il FS).
  
- Il rischio ovvio di riempire un CG con sole directory, non lasciando spazio per i file, puo' essere evitato impostando opportuni parametri tramite tunefs (sebbene quelli di default siano piu' che soddisfacenti).

# OpenBSD Internals: What makes it different

## Dirhash (1)

---

□ /usr/src/sys/ufs/ufs/dirhas.h

```
/*  
 * For fast operations on large directories, we maintain a hash  
 * that maps the file name to the offset of the directory entry within  
 * the directory file.  
 */
```

```
/*  
 * Dirhash uses a score mechanism to achieve a hybrid between a  
 * least-recently-used and a least-often-used algorithm for entry  
 * recycling. The score is incremented when a directory is used, and  
 * decremented when the directory is a candidate for recycling. When  
 * the score reaches zero, the hash is recycled. Hashes are linked  
 * together on a TAILQ list, and hashes with higher scores filter  
 * towards the tail (most recently used) end of the list.  
 */
```

## Dirhash (2)

---

- Ottimizzazione introdotta da Ian Dowse.  
[http://www.maths.tcd.ie/report\\_series/tcdmath/tcdm0206.pdf](http://www.maths.tcd.ie/report_series/tcdmath/tcdm0206.pdf)
- Anche se evita ripetute ricerche lineari, la creazione della hash table ha comunque un certo peso computazionale. Per questo motivo, e' possibile impostare la quantita' di memoria dedicabile alle hash table e la dimensione minima di una directory affinche' venga hashata (default 2MB e 2.5kB).

## Soft Updates (1)

---

- Una delle soluzioni possibili al problema del recupero della consistenza di un file system su disco. Alternativa molto piu' performante delle synchronous write.
- Le principali alternative sono rappresentate dai Journaled File System, o dall'utilizzo di NVRAM (soluzione di relativo successo).
- Riordina (con opportuni check) e scrive sequenzialmente (async con consistenza).

## Soft Updates (2)

---

- Ideato da Ganger e Patt. Implementato da McKusick su FreeBSD (addottato da OBSD 2.3 Nov '97).
  
- [http://www.usenix.org/publications/library/proceedings/usenix99/full\\_papers/mckusick/mckusick.pdf](http://www.usenix.org/publications/library/proceedings/usenix99/full_papers/mckusick/mckusick.pdf)
  
- Sebbene debbano essere controllate le dipendenze durante gli update della cache (spesso cicliche), il suo utilizzo apporta sempre ottimi livelli di miglioramento (fino ad un fattore 20 per grandi carichi di lavoro sull'update dei metadati).

# OpenBSD Internals: What makes it different

## fstab

---

```
struct fstab {
    char    *fs_spec;        /* block special device name */
    char    *fs_file;       /* filesystem path prefix */
    char    *fs_vfstype;    /* type of filesystem */
    char    *fs_mntops;     /* comma separated mount options */
    char    *fs_type;       /* rw, ro, sw, or xx */
    int     fs_freq;        /* dump frequency, in days */
    int     fs_passno;      /* pass number on parallel fsck */
};
```

man fstab && vim /etc/fstab

man mount && man umount

## Mount options (1)

---

- Read-only: ("rdonly"). Opzione sicura quanto spesso inutilizzabile (se non per / e /usr, o per il recupero di file system danneggiati).
  
- Read-Write: ("rw"). Opzione standard (ma sempre piu' rimpiazzata da softdep).
  
- No Access Time: ("noatime"). FFS registra gli accessi ai file (ultima esecuzione, ultima lettura, etc.). Questa opzione permette di risparmiare il tempo dedicato alla registrazione di tali accessi.

## Mount options (2)

---

□ Synchronous: ("sync"). Con questa opzione, la velocita' di lettura dipende solamente dal transfer rate del disco. Al contrario, le operazioni di scrittura sono rallentate dalla politica di sync -> Fasi di scrittura:

- o1) Il kernel scrive un chunk di dati sul disco.
- o2) Il kernel aspetta dal disco informazioni sulla scrittura avvenuta.
- o3) In caso di scrittura avvenuta, il kernel avvisa il programma che ha effettuato le call creat(), write(), etc.

Sebbene assicuri piu' di altre opzioni l'integrita' dei dati in caso di crash, puo' risultare nella maggior parte dei casi una soluzione sconveniente.

□ Asynchronous: ("async"). Antitetica alla precedente opzione (con conseguente rischio di perdita di dati).

## Mount options (3)

---

- **Soft Update:** ("softdep"). Supervisiona e organizza le operazioni di scrittura dei metadati sul FS, affinché rimanga in stato consistente. Offre le performance di "async" assicurando l'integrità di "sync". Non previene la perdita di dati in caso di power failure!
  
- **Noexec:** ("noexec"). Nega la possibilità di eseguire file contenuti nella partizione (utile ad esempio per /home o per spazi condivisi come /tmp).
  
- **Nosuid:** ("nosuid"). Disabilita il comportamento caratteristico di un programma suidato.
-

## Disk Quota

---

- Supportata solo da FFS, attivabile modificando le mount options in /etc/fstab (/dev/wd0g /home ffs rw,userquota,groupquota,...). In questo caso, i file /home/quota.user e /home/quota.group contengono i limiti (editabili tramite edquota).
- 4 sono i limiti impostabili sia per user che per group (rispettivamente, 2 soft [alert policy] e 2 hard [block policy]) -> blocks in use && inodes in use.

# When everything seems to be lost... (1)

---

- A causa di un system failure, alcuni inode potrebbero essere stati editati solo in parte. Queste inconsistenze devono essere risolte prima di montare nuovamente il FS in "rw" mode.
- fsck: FileSystem consistency Check (fondamentale dopo un unmount forzato) -> flag utili: -f forza l'esecuzione anche in caso di filesystem montato, -n/-y rende fsck non interattivo e impone una politica di default.

# When everything seems to be lost... (2)

---

- `scan_ffs`: restituisce la lista delle partizioni individuate durante lo scan di un intero device. Individuando `disk size`, `filesystem`, `fsize`, `bsize`, `last mount point`, etc., e' utilizzabile, con l'ausilio di `disklabel`, per la ricostruzione di una `partition table` corrotta.
  
- `man dump && man fsdb && man clri`
  
- `Background fsck`: non ancora implementato in OpenBSD. <http://www.usenix.org/publications/library/proceedings/bsdcon02/mckusick.html>

OpenBSD Internals: What makes it different

# Encrypted Partition Demo + Q&A

---

- Encrypted Partition Demo:  
[www.backwatcher.org/writing/howtos/  
obsd-encrypted-filesystem.html](http://www.backwatcher.org/writing/howtos/obsd-encrypted-filesystem.html)
  
- Link di riferimento:  
<http://www.forensics.nl/filesystems>
  
- Q&A :)

Thanks for all the attention!

pirroH  
[Intel Inside, Idiot Outside]

