CS 1550 – Chapter 6 File Systems

Jonathan Misurda
jmisurda@cs.pitt.edu

Files

File Naming

- Case Sensitive
  - Linux/UNIX
- Case Insensitive
  - DOS
- Case Insensitive, Case Preserving
  - Windows
  - Mac

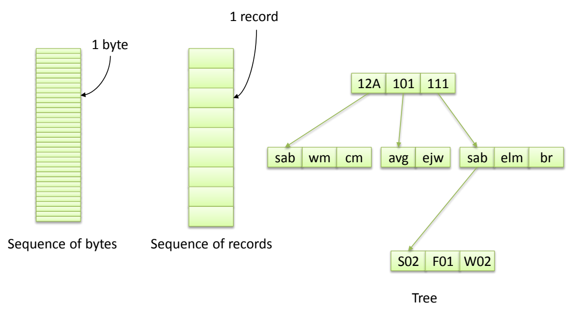
File Extensions

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

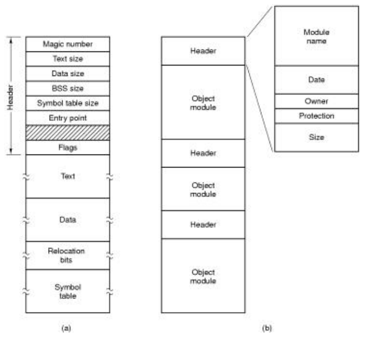
Metadata

- Data that describes data
  - Type of file
  - Creator
  - Structure of the data
- *Is an extension a good place to record metadata?*

File Structure



File Types



File Attributes

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

File Operations

- Create
- Delete
- Open
- Close
- Read
- Write
- Append
- Seek
- Get attributes
- Set attributes
- Rename

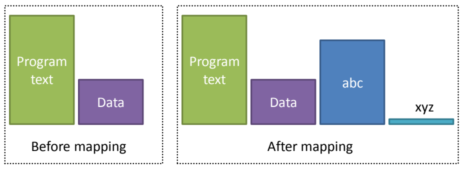
Using System Calls

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                    /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);         /* ANSI prototype */

#define BUF_SIZE 4096                     /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                  /* protection bits for output file */

int main(int argc, char *argv[])
{
     int in_fd, out_fd, rd_count, wt_count;
     char buffer[BUF_SIZE];

     if (argc != 3) exit(1);              /* syntax error if argc is not 3 */
```

Using System Calls (2)

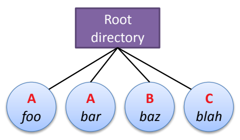
```
     /* Open the input file and create the output file */
     in_fd = open(argv[1], O_RDONLY);    /* open the source file */
     if (in_fd < 0) exit(2);             /* if it cannot be opened, exit */
     out_fd = creat(argv[2], OUTPUT_MODE); /* create the destination file */
     if (out_fd < 0) exit(3);            /* if it cannot be created, exit */

     /* Copy loop */
     while (TRUE) {
          rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
     if (rd_count <= 0) break;           /* if end of file or error, exit loop */
          wt_count = write(out_fd, buffer, rd_count); /* write data */
          if (wt_count <= 0) exit(4);    /* wt_count <= 0 is an error */
     }

     /* Close the files */
     close(in_fd);
     close(out_fd);
     if (rd_count == 0)                  /* no error on last read */
          exit(0);
     else
          exit(5);                       /* error on last read */
}
```

Memory-mapped Files

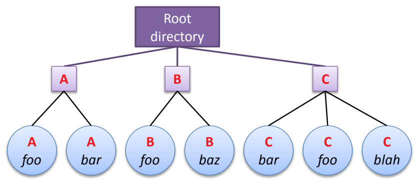


Directories

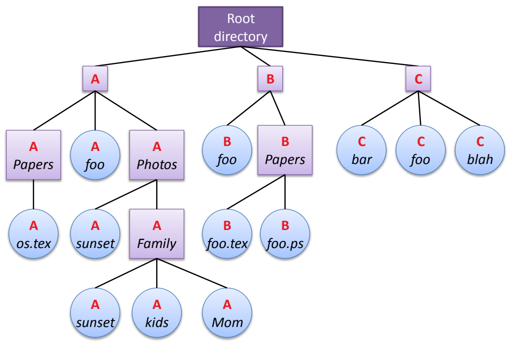
Single Level Directory



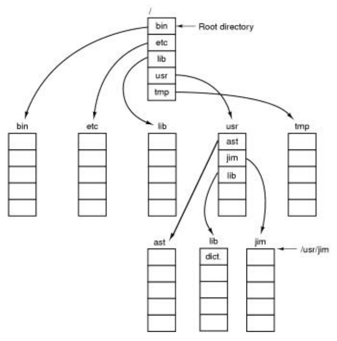
Two-Level Directory



Hierarchical File System



UNIX Directory Structure

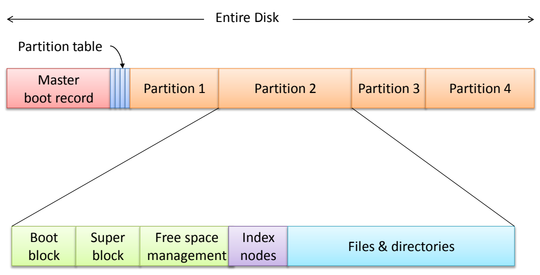


Directory Operations

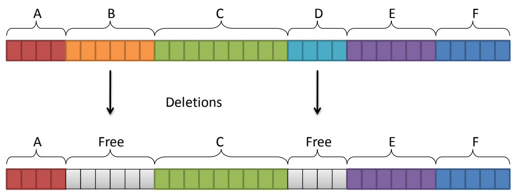
- Create
- Delete
- Opendir
- Closedir
- Readdir
- Rename
- Link
- Unlink

File System Implementation

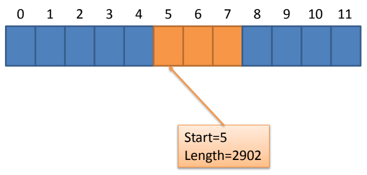
Disk Layout



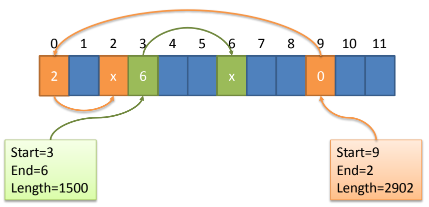
Contiguous Allocation



Contiguous Allocation



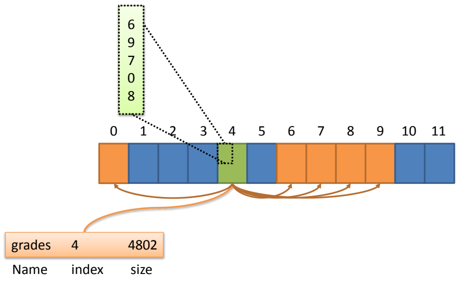
Linked Allocation



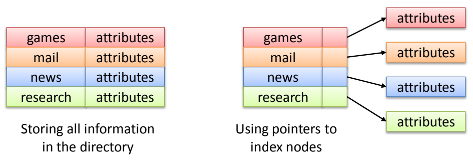
File Allocation Table (FAT)

0	2	
1	-1	
2	-2	
3	-2	
4	-1	
5	-1	
6	3	B
7	-1	
8	-1	
9	0	A
10	-1	
11	-1	
12	-1	
13	-1	
14	-1	
15	-1	

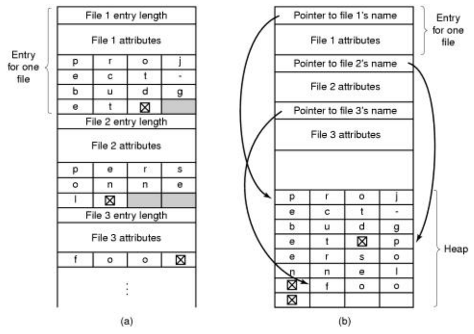
Index Nodes (i-nodes)



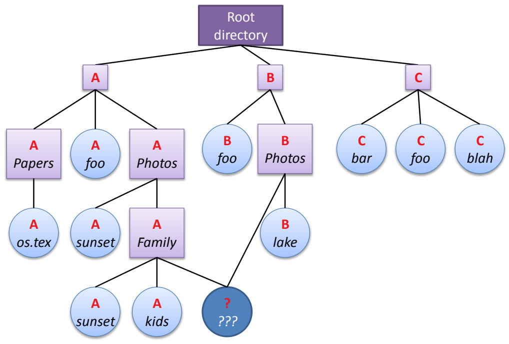
Directories and Attributes



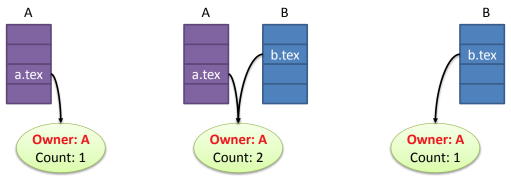
Long Filenames



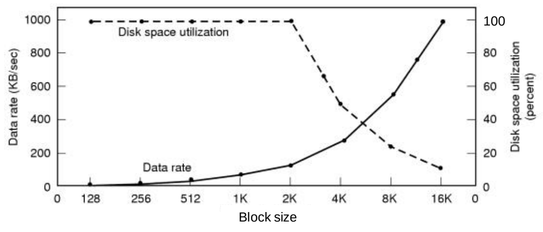
Sharing Files



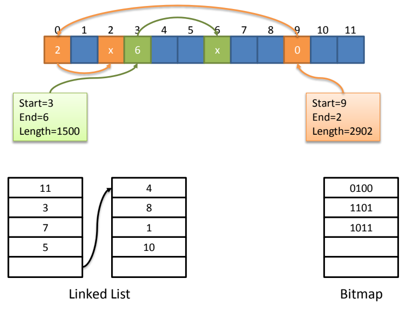
Links



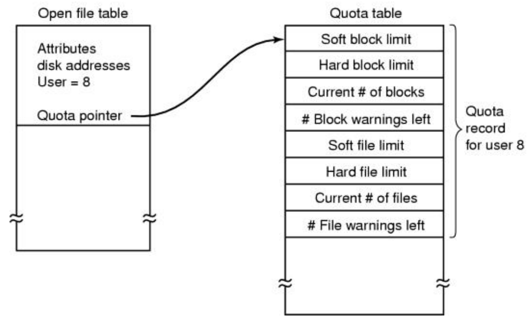
Disk Space Management



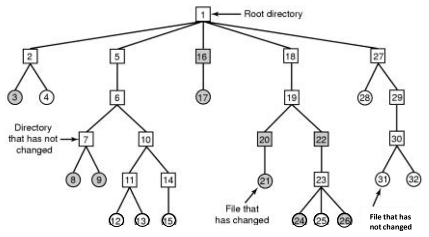
Free Block Tracking



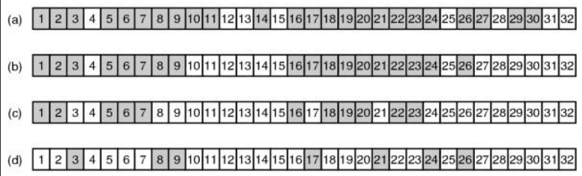
Disk Quotas



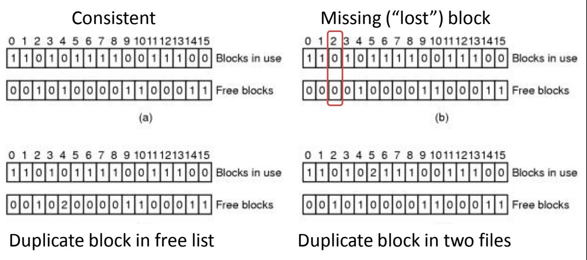
Backing Up



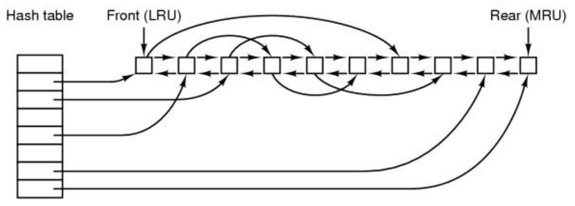
Bitmap From Dump



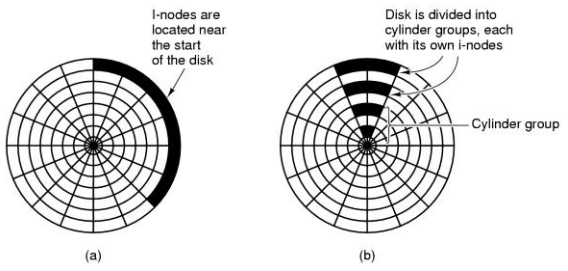
Consistency Checking


Consistent
Missing (“lost”) block
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 Blocks in use
0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 Free blocks
(a)
1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 Blocks in use
0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 Free blocks
(b)
1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 Blocks in use
0 0 1 0 2 0 0 0 0 1 1 0 0 0 1 1 Free blocks
1 1 0 1 0 2 1 1 1 0 0 1 1 1 0 0 Blocks in use
0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 Free blocks
Duplicate block in free list
Duplicate block in two files

File Block Cache

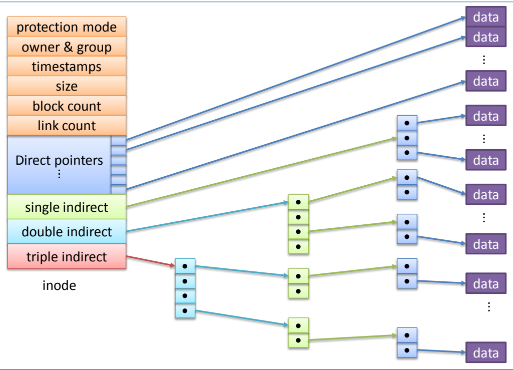


Grouping On-Disk Data

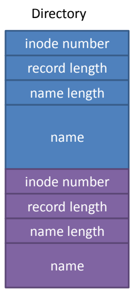

I-nodes are located near the start of the disk
Disk is divided into cylinder groups, each with its own i-nodes
Cylinder group
(a)
(b)

Log Structured File Systems

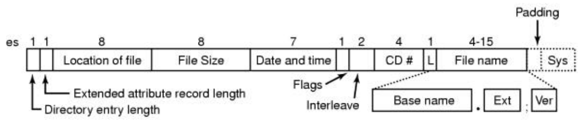
UNIX Fast File System



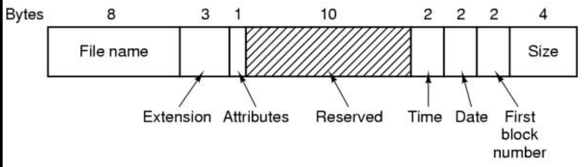
FFS Directory



CDFS



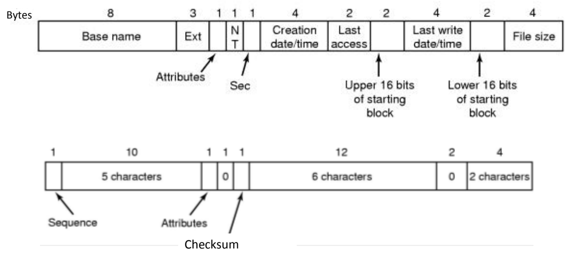
FAT Directory



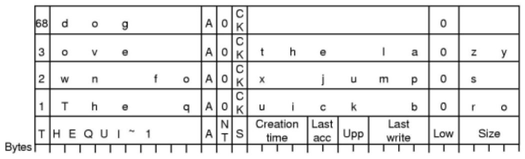
MS FAT

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

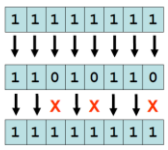
FAT32 Directory and Filename



Long Filename in FAT32


68 d o g A 0 CK 0
3 o v e A 0 CK t h e l a 0 z y
2 w n f o A 0 CK x j u m p 0 s
1 T h e q A 0 CK u i c k b 0 r o
T H E Q U I ~ 1 A NT S Creation time Last acc Upp Last write Low Size
Bytes

Flash File Systems



Start off with all 1's

Programming from 1's to 0's is allowed

Programming from 0's to 1's is not allowed

Wear Leveling

Count total writes per flash sector and attempt to balance across the whole disk