

Today
 Status
 Nachos File System
 The True NFS Insight
 Assignment 4: File Systems
 CS 439: Operating Systems
 Stanford University

Today {}

- Status
- Nachos file system
- Assignment 4: File Systems

Today
 Status
 Nachos File System
 The True NFS Insight
 Assignment 4: File Systems
 CS 439: Operating Systems
 Stanford University

Status {}

- Lab 1: 12/12
- Lab 2: 6/12
 - No major problems
 - Good problem understanding
- Lab 3: 1/12
- Currently, you should be working on the 3rd lab.

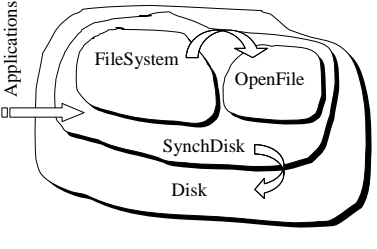
Today
 Status
 Nachos File System
 The True NFS Insight
 Assignment 4: File Systems
 CS 439: Operating Systems
 Stanford University

Overview {Nachos File System}

- Two different FS versions:
 - a **front-end** to UNIX
 - a very **simple FS based on a raw disk emulation** (the True NFS)
- Both have the same interface and services
- Nachos uses one or the other depending on the -DFLESYS_STUB compile flag

Today
 Status
 Nachos File System
 The True NFS Insight
 Assignment 4: File Systems
 CS 439: Operating Systems
 Stanford University

The True NFS Insight



Today
 Status
 Nachos File System
 The True NFS Insight
 Assignment 4: File Systems
 CS 439: Operating Systems
 Stanford University

Disk

- Emulates a raw disk
 - single platter -> every block is uniquely identified by a **sector number**
 - can serve one request at one time, no mutual exclusion
 - hardware: **DO NOT CHANGE!**
- More in "A Road Map Through Nachos", 2.6

Today
 Status
 Nachos File System
 The True NFS Insight
 Assignment 4: File Systems
 CS 439: Operating Systems
 Stanford University

SynchDisk

- Uses the raw disk
- Provides sector level mutual exclusion
- As Disk, can only read/write full sectors
- Works fine as it is! **DO NOT CHANGE!**
- More in "A Road Map Through Nachos", 5.1

Tr 407
Scribe
Nachos File System
The True NFS Representation
Assignment 4: File System
TODAY: October, 2003, 4, 10:23:00
University of Berkeley

FileSystem

- Memory resident image of NFS
- Typical methods:
 - Create
 - Open
 - Remove
- This is the interface to the file system, thus NO MAJOR CHANGES NEEDED!
- More in "A Road Map Through Nachos", 5.2

Tr 407
Scribe
Nachos File System
OpenFile
Assignment 4: File System
TODAY: October, 2003, 4, 10:23:00
University of Berkeley

OpenFile

- Memory resident structure associated with a file
- Typical Methods:
 - OpenFile(int sector) - sector identifies the file
 - Read, ReadAt - use Synchron Disk Read
 - Write, WriteAt - use Synchron Disk Read/Write
 - Length
- More in "A Road Map Through Nachos", 5.3

Tr 407
Scribe
Nachos File System
The True NFS Representation
Assignment 4: File System
TODAY: October, 2003, 4, 10:23:00
University of Berkeley

The True NFS Representation

- |or: "What's on the (Synch)Disk?"

Sector 0	Sector 1	Sector 2	Sector k
freeMapFile	directoryFile	empty	Beginning of a file

BitMap Object

Keeps the sector allocation map

Directory Object

Table of pairs:
• filename
• start sector

FileHeader Object

Information about the file:
• size in bytes and sectors
• table of assigned sectors

Tr 407
Scribe
Nachos File System
BitMap, Directory, FileHeader
Assignment 4: File System
TODAY: October, 2003, 4, 10:23:00
University of Berkeley

BitMap, Directory, FileHeader

- BitMap, Directory, FileHeader
- Disk resident images - access methods:
 - FetchFrom(int sector)
 - WriteBack(int to_sector)
- Common resources
 - several processes use them maybe at the same time
- More in "A Road Map Through Nachos", 5.4.1 and 5.4.2

Tr 407
Scribe
Nachos File System
Drawbacks of Today's NFS
Assignment 4: File System
TODAY: October, 2003, 4, 10:23:00
University of Berkeley

Drawbacks of Today's NFS

- No synchronization at file system level
- Small file size (4kb), see FileHeader
- Fixed file size, no dynamic allocation
- No hierarchical directory structure
- No delete system call
- and more

Tr 407
Scribe
Nachos File System
What to do ? - Part I (Assignment 4: File System)
Assignment 4: File System
TODAY: October, 2003, 4, 10:23:00
University of Berkeley

What to do ? - Part I (Assignment 4: File System)

Develop the file system

- each process **closes** all [self opened] files at exit
- add **synchronisation** [assignment 1]
- allow **many read/single write**
- file system operations have to be **atomic**
- correct **delete management**
 - handle cases when a file is deleted while still used by other processes

Title: **Hints & Requirements - Part I**
 Author: **Stefan**
 Reviewer:
 The File System Design
 CS63C, Stanford
 Copyright © 2004
 The File System Design is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike license.
 Attribution: This document is part of the File System Design, a book by the author.
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)

Hints & Requirements - Part I

- DO NOT**
 - Lock the whole FS
 - Modify Disk nor Synch Disk
- DO**
 - Modify Open File and File System
 - Use locks wisely: not too many, not too few
 - WriteBack all or nothing

Suggested Structure

Title: **What to do ? - Part II (Optional)**
 Author: **Stefan**
 Reviewer:
 The File System Design
 CS63C, Stanford
 Copyright © 2004
 The File System Design is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike license.
 Attribution: This document is part of the File System Design, a book by the author.
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)

What to do ? - Part II (Optional)

Add support for **large files** (disk size)

- might want to use indirection levels
 - easiest: entry in FileHeader pointing to new FileHeaders (new level)
 - Modify FileHeader::ByteToSector to get the right sector
 - More in the course book [subsection 11.2.3, page 381, Fig. 11.7]

Title: **What to do ? - Part III (Optional)**
 Author: **Stefan**
 Reviewer:
 The File System Design
 CS63C, Stanford
 Copyright © 2004
 The File System Design is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike license.
 Attribution: This document is part of the File System Design, a book by the author.
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)

What to do ? - Part III (Optional)

Allow **growing file sizes**

- Allocate more sectors if Write exceeds the current file length

Title: **Useful**
 Author: **Stefan**
 Reviewer:
 The File System Design
 CS63C, Stanford
 Copyright © 2004
 The File System Design is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike license.
 Attribution: This document is part of the File System Design, a book by the author.
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)
 What is it? - Part III (Optional)

Useful

- Work in `$HOME/nachos-3.4/code/filesys`
 - check Nachos run parameters used for FS
 Example: `nachos -E -cp test/E1 E1`
 [formats nachos drive and copies fl from UNIX dir to Nachos FS]
- Course book, chapter 21.7
- Man pages for UNIX file access functions [for comparison]

Good Luck!